МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ВИПУСКНА РОБОТА

на тему:

«Інформаційний чат-бот месенджеру Telegram для вирішення задач чисельними методами»

Завідувач випускаючої кафедри — **Довбиш А.С.**

Керівник роботи — **Шовкопляс О.А.**

Студента групи ІН – 71 — **Видриган В.О.**

СУМИ 2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних наук

Затверджую_____

Зав. кафедрою Довбиш А.С.

"_____" _____2021 р.

ЗАВДАННЯ
до випускної роботи

Студента четвертого курсу, групи ІН-71 спеціальності "Комп'ютерних наук " денної форми навчання Видригана Владислава Олеговича.

Тема: " Інформаційний чат-бот месенджеру Telegram для вирішення задач чисельними методами "

Затверджена наказом по СумДУ

№_____ від _____2021 р.

Зміст пояснювальної записки: 1) Інформаційний огляд. Огляд проблемної області. Особливості використання Telegram бот. Постановка задачі. 2) Вибір методу рішення. Вибір програмної реалізації. СУБД. Інші бібліотеки та інструменти. Середовище програмування. Вибір бібліотеки для взаємодії з Telegram. 3) Практична реалізація. Інформаційна модель. Створення бота. Строврення бази даних. Опис викинаної роботи.

Дата видачі завдання "_____"_____2021 р.

Керівник випускної роботи _____ Шовкопляс О.А.

Завдання приняв до виконання _____ Видриган В.О.

РЕФЕРАТ

Записка: 59 стор., 29 рис.,1 таблиця, 1 додаток, 15 джерел.

Об’єкт дослідження – проблема відсутності засобів для вирішення задач чисельними методами.

Мета роботи – розробка коду на базі кросплатформного засобу доступного широкому колу осіб.

Методи дослідження – методи дослідження пристроїв та найбільш використаних програмних додатків.

Результати – розроблено чат-бот месенджеру Telegram для вирішення алгебраїчних задач чисельними методами.

НАВЧАЛЬНИЙ ПРОЦЕС,ЧАТ-БОТ, PYTHON,БАЗА ДАНИХ,ЧИСЕЛЬНІ МЕТОДИ.

ЗМІСТ

ВСТУП

З розвитком сучасних технологій та їх доступністю кількість нових користувачів Інтернету неодмінно зростає. За останній рік це число збільшилось на 316 мільйонів з них 93 мільйона це нові користувачі, що підключаються через мобільні пристрої.[1]

Мобільні пристрої в даний час є найбільш широко використовуваними для підключення до Інтернету і тенденції говорять про мимовільне витіснення інших пристроїв таких як персональні комп'ютери, портативні консолі, хоч ними користується ¾ користувачів.[1][2]

Основним видом додатків для мобільних пристроїв з доступом в Інтернет є соціальні мережі, які дозволяють поширювати і отримувати інформацію як від авторитетних ЗМІ так і від своїх знайомих.[3] Так за результатами дослідження CMeter Mobile на початку 2021 року найпопулярнішими мобільними додатками серед користувачів України були Viber(96,2%), Facebook(85,7%), Telegram(76,8%), Fb Messenger(72,3%), Instagram(69%).[4] Їх неймовірної популярності за останнє десятиліття і рекорди одночасних користувачів не зникають з новин. Лідером серед них по приросту нових користувачів є месенджер – Telegram який й далі стабільно нарощує свою аудиторію, випередивши в січні цього року Fb Messenger.

Telegram має обширну аудиторію не лише через свою захищеність даних, спілкування з одним співрозмовником чи з 100 000 одночасними, а й за різноманітні додаткові функції. Так потрібно згадати про його кросплатформність – доступний на всіх платформах. В Telegram є додатки під всі платформи (Android, IOS, Windows, Mac OS,), веб версія в який можна потрапити з всіх сучасних браузерів, також є підтримка для смарт годинників.

Для користувачів месенджеру, які вміють та мають бажання писати код Telegram надає можливість безкоштовного створення чат-ботів за допомогою API для своєї платформи під різні цілі.

Метою кваліфікаційної роботи бакалавра є створення інформаційної системи для вирішення задач чисельними методами у вигляді чат-боту для месенджері Telegram, яким зможуть користуватися як студенти, викладачі так і пересічні користувачі платформи для знаходження розв’язку лінійних та нелінійних рівнянь та систем рівнянь.

1 ІНФОРМАЦІЙНІЙ ОГЛЯД

1.1 Огляд проблемної області

На сьогоднішній день боти зайняли важливе місце в інформаційній системі людства. Вони використовуються в більшості сфер життєдіяльності людини, але що це таке?

Чат-бот - це сторонній код, який отримує від користувача повідомлення, команди і вбудовані запити та надсилає відповіді на поставлені задачі.

За допомогою чат-ботів можна спрощувати комунікацію з користувачем шляхом завчасного написання сценарію поведінки чат-бота, що дасть можливість полегшити діалог користувача з чат-ботом.[6]

На інтерес аудиторії велика кількість організацій почала створювати власних чат-ботів, щоб залучити увагу користувача чи просто надати доступ до свого функціоналу. Чат-бот став новим каналом просування чи просто рекламним ходом для любого роду бізнеса чи то малого, середнього чи навіть міжнародного.[7]

Чат-боти відрізняються між собою в залежності від поставленої задачі. Так їх розділити можна на використання для спілкування між користувачами, розважальних, інформаційних цілях, а саме деякі можливості чат-ботів:

- повідомити погоду;
- конвертація повідомлення в голос та навпаки;
- нагадування;
- розпізнання треків;
- запис до лікаря;
- переклад тексту;
- управління персоналом;
- розсилка акцій;
- бронювання білетів;
- онлайн консультація;

- вивчення іноземних мов;
- інтеграція з різними магазинами;
- пошук вакансій;
- оплата товарів;
- пошук курсів;
- і тд..

Хоч чат-боти і набирають популярності в різних сферах життєдіяльності людини, але на даний момент не існує жодного бота що зміг надати інформацію й вирішити задачі чисельними методами.

Тому було прийнято рішення створити власного чат-бота, який мав би можливість вирішувати поставлені задачі й допомагати іншим в засвоєнні методів вирішення цих задач.

1.2 Особливості використання Telegram бот

Боти в Telegram це дещо змінені облікові засоби, які не потребують номера мобільного телефону для свого створення, роль яких автоматично отримувати, обробляти та відправляти результат за допомогою HTTPS запитів. Для цього в Telegram існують безкоштовна бібліотеки для взаємодії з Telegram Bot API.[5]

Користувачі можуть взаємодіяти з ботами двома способами:

- надсилати повідомлення та або команди ботам;
- надсилати запити безпосередньо з поля введення, ввівши @username бота та запит.

Також облікові засоби ботів мають ряд змін, що допомагають відрізнити їх від решти, а саме:

- відсутність статусів;
- ім’я бота завжди закінчується на bot:
- користувач сам шукає бота і починає розмову з ним.

1.3 Постановка задачі

Головною метою роботи є створення Інформаційного чат-боту месенджеру Telegram для вирішення задач числовими методами. Даний бот повинен шукати розв’язки:

- нелінійні рівняння;
- системи лінійних рівнянь;
- системи не лінійних рівнянь;

Для інформаційної допомоги у вирішенні задач було обрано такі методи:

- для нелінійних рівняння:
  - метод Ньютона;
  - метод Дихотомії;
  - метод Хорд.
- для системи лінійних рівнянь:
  - метод Зейделя;
  - метод Простої ітерації.

Для системи нелінійних рівнянь вивести лиш відповідь.

У разі допущення помилки з боку користувача довести його до відома.

Задля полегшення в управлінні ботом, користувач повинен мати можливість налаштувати деякі параметри пошуку під себе, такі як:

- точність пошуку відповідей;
- відображення графіка(для збереження трафіка)

Графічну структуру меню можна побачити на рисунку 1.1.

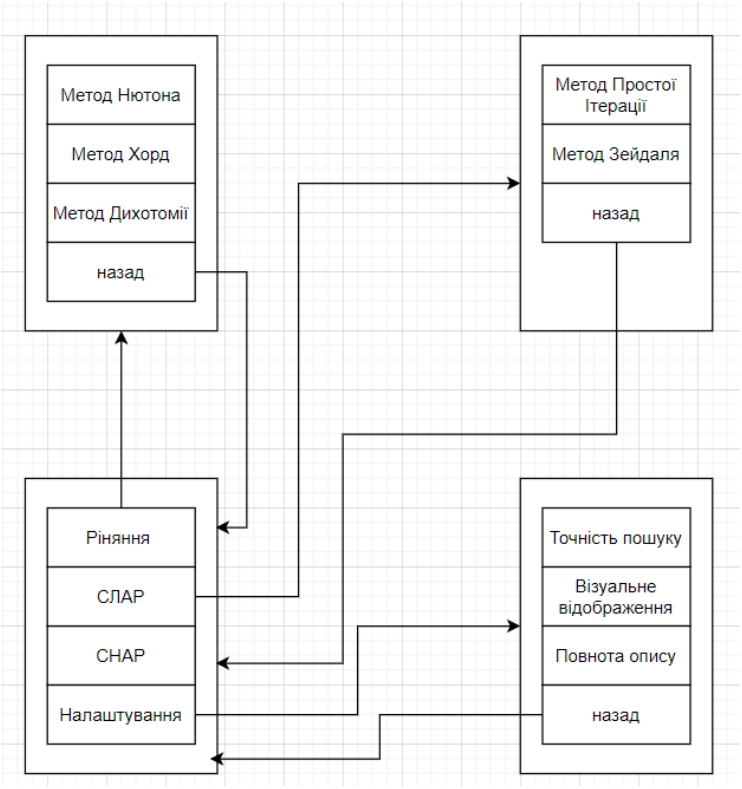

Рисунок 1.1 — Графічне зображення меню

Також при натисканні на метод потрібно добавити можливість повернутися на попередній крок. Це обумовлено оптимізацією діалогу з користувачем

Для зменшення навантаження на серверну частину бота, було вирішено зберігати створені графіки користувачів для подальшого використання.

2 ВИБІР МЕТОДУ РІШЕННЯ

2.1 Вибір програмної реалізації

Для створення чат-бота було обрано мову програмування Python.

Python – об’єктно орієнтована мова програмування, яка дозволяє створювати читабельний код. Дана мова має рекомендації в написанні коду PEP 8, для полегшення форматування коду. Python має вбудовані структури даних списків та словників, що зменшує час для написання коду. Кросплатформність мови Python дозволяє запускати її код на різних операційних системах , що допоможе в майбутньому перенести чат-бота на сервер. На цій мові програмування велика кількість навчальних матеріалів, що допоможе написати код.[8]

Python має стандартну бібліотеку інструментів, які має велике значення в написанні читабельного коду.

Модуль os надає безліч функцій для роботи з ОС, але в нашому випадки, ми обійдемося лиш пошуком створених графіків, щоб в разі повторного введення певного рівняння не створювати новий графік, а лиш знайти старий та відправити його користувачеві.[9]

2.2 СУБД

Для збереження даних в базі даних було обрано модуль sqlite. Це доволі компактна реляційна система керування базами даних. Дана база даних є суспільним надбання, може бути використана без обмежень та безкоштовно з будь якою метою.[10] Основним плюсом є не використання парадигми клієнт-сервер – що значить база компілюється разом з іншим кодом, тобто це не окремий процес, з яким взаємодіє інша частина коду. Завдяки цьому збільшується швидкість відгуку. Також плюсом є те що sqlite зберігається в єдиному файлі, що дає змогу робити бекап бази даних дещо комфортнішим.

Щоб створити базу даних та для подальшої візуалізації процесу було прийняте рішення використати DB Browser for SQLite. Дана програма -

високоякісний візуальний інструмент з відкритим вихідним кодом для створення, проектування і редагування файлів баз даних, сумісних з SQLite.[11]

2.3 Інші бібліотеки та інструменти

Також буде використана бібліотека matplotlib. Дана бібліотека використовується для візуалізації даних двовимірними та трьохвимірними графіками. Matplotlib поширюється на умовах BSD. Саме нею будуть побудовані графіки для візуалізації даних.[12]

Для збереження нових користувацьких налаштування та виконаних задач в рамках оптимізації щоденного в 3 ранку відбуватиметься очищення тимчасових файлів.

2.4 Середовище програмування

Середовищем для написання чат-бота мовою програмування Python було обрано продукт компанії JetBrains PyCharm Community Edition 2020.

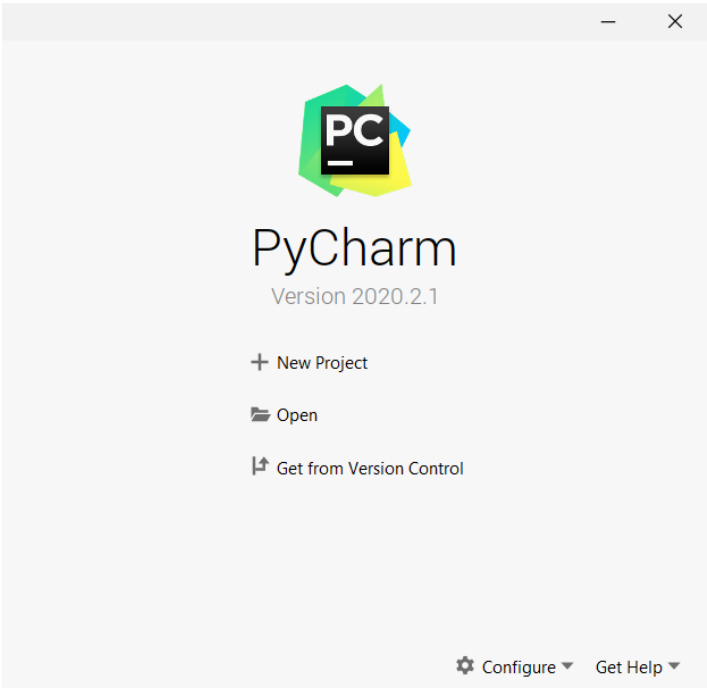

Рисунок 2.1 — Початкове меню PyCharm

Дане середовище – має засоби для аналізу коду, а також допоможе написати гарний читабельний код за допомогою вбудованих рекомендацій написання коду PEP8, що належить мові програмування Python. Важливу роль

грає і те, що всі бібліотеки, використані під час створення проекту знаходяться в одній папці venv. Це полегшить в майбутньому перенесення чат-бота на інший пристрій чи сервер.[13]

2.5 Вибір бібліотеки для взаємодії з Telegram

Для роботи з чат-ботом месенджера Telegram потрібно використовувати бібліотеку написана на мові програмуванні Python та дозволить виконувати запити до Telegram Bot API простими конструкціями. Таку бібліотеку має сам Telegram, вона має назву pyTelegramBotAPI.[14] Даний вибір обумовлено відсутністю створення інтерфейсу взаємодії користувача з ботом, а також можливістю створенням прогресивного діалогу за допомогою використання користувачем клавіатури, кнопок, і так далі. Використання цієї бібліотеки дозволяє зменшити час на розробку чат-бот, розділити написання коду на окремі частини, що зробить код більш читабельним.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ

3.1 Інформаційна модель

Для опису принципу роботи коду та її складових була створена схема:

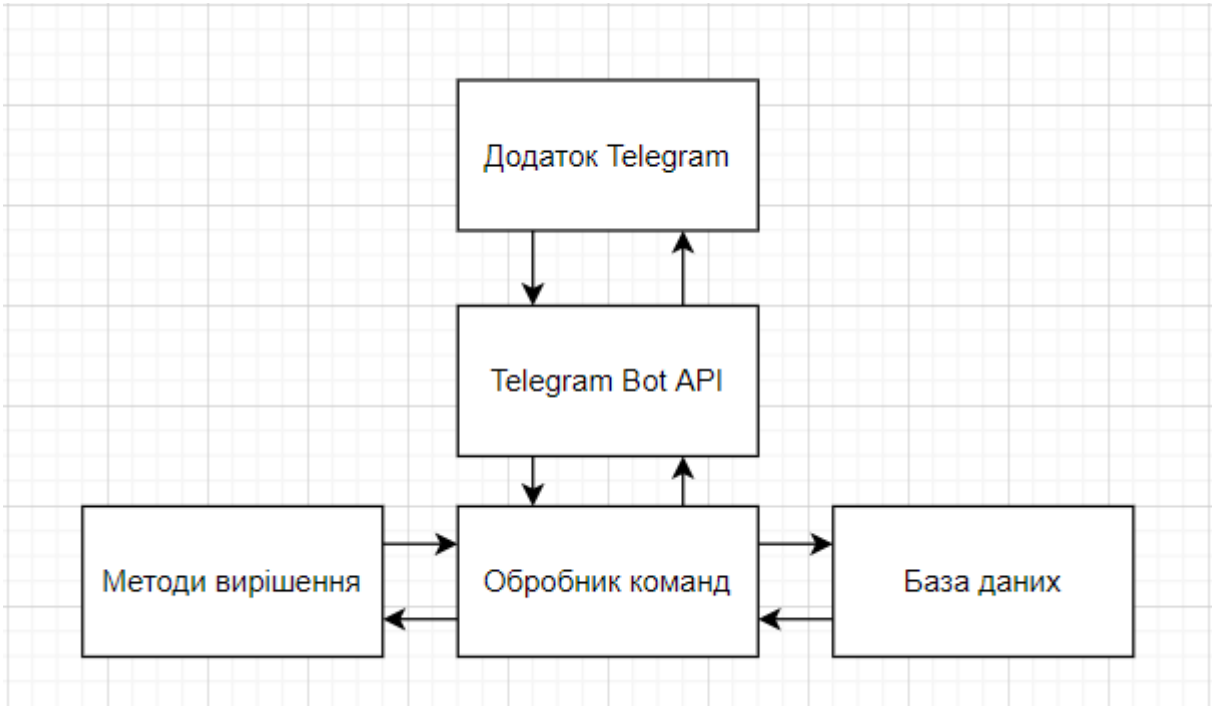

Рисунок 3.1 — Структура бота

Звідси видно, що:

- месенджер Telegram виконує роль платформи для розробки боту.
- Telegram Bot API виконує роль посередника між месенджером Telegram та Обробником команд. Так він отримує повідомлення від користувача (повідомлення @my_bot) і в разі необхідності відправляє відповідь користувачу з id користувачем
- Обробник команд отримує команду від Telegram Bot API, знаходить відповідний сценарій виконує його за допомогою Бази даних та Методів вирішення та отриманий результат відправляє назад Telegram Bot API.

Також було створено діаграма варіантів використання.

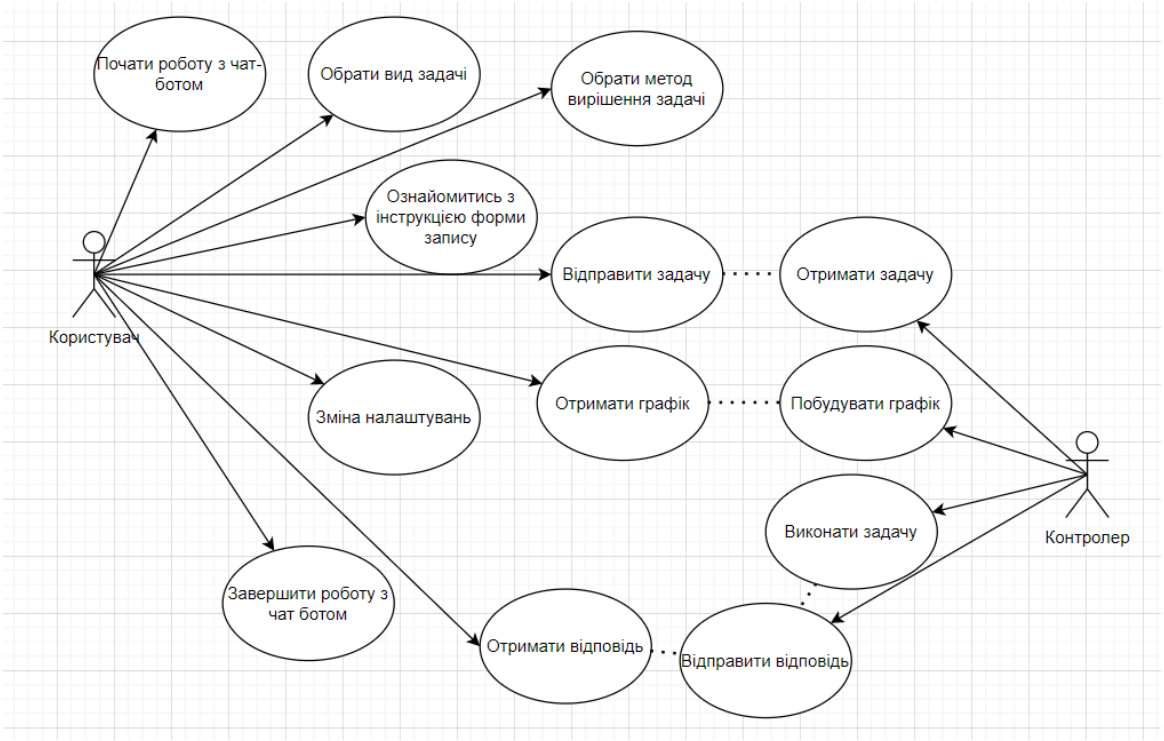

Рисунок 3.2 — Use Case діаграма

3.2 Створення бота

Перед початком роботи, потрібно створити чат-бот. Для цього потрібно в месенджері Telegram знайти бот по його username @BotFather.[14]

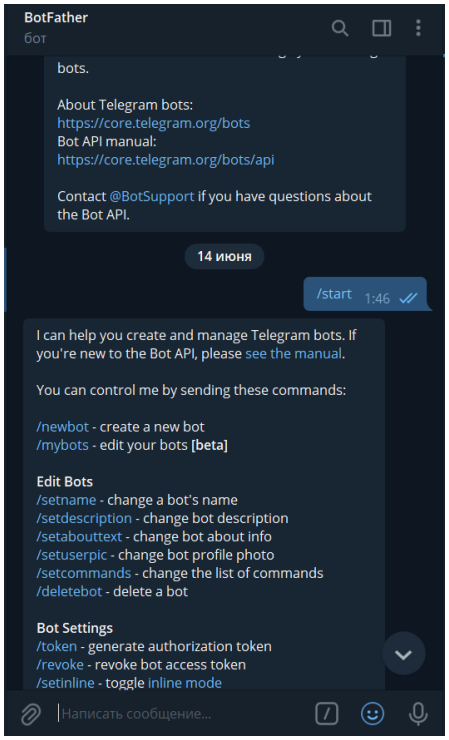

Рисунок 3.3 — Початкова сторінка бота @BotFather

Йому написати команду «/newbot», що дозволить зареєструвати нового бота. Далі вказати ім’я та username бота.

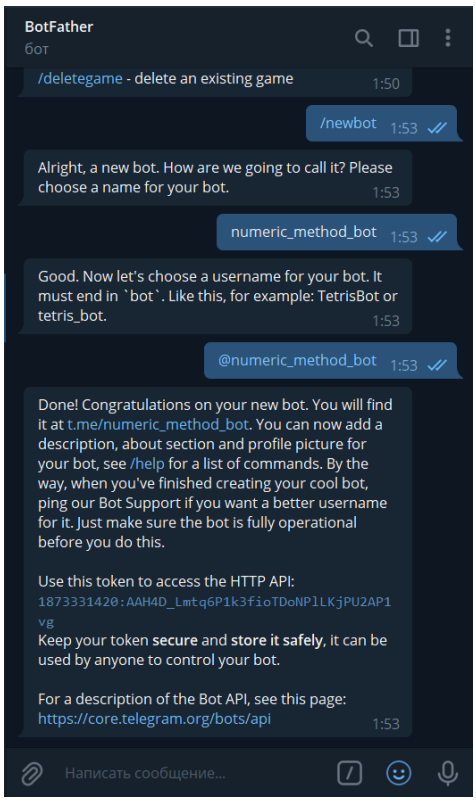

Рисунок 3.4 — Мінімальний набір команд для створення бота

Все Telegram Token отримано, можна приступати до писання коду.

3.3 Створення бази даних

Щоб дані користувачів у разі виникнення аварійної ситуації не було втрачено ці дані будуть зберігатися у базі даних. В одній буде міститься інформація про налаштування користувача, в іншій інформацію про створені графіки, що дасть смогу скоротити час на взаємодію з ботом. Це в свою чергу зменшить навантаження на серверну частину бота.

Код таблиці для налаштування користувачів:

```
CREATE TABLE "user" (
"id"   INTEGER NOT NULL UNIQUE,
"graf" TEXT,
"details"    TEXT,
"accuracy"   INTEGER,
PRIMARY KEY("id")
)
```

Код таблиці для збереження графіків:

```
CREATE TABLE "formula" (
"id"   INTEGER NOT NULL UNIQUE,
"name" TEXT NOT NULL UNIQUE,
"x"    TEXT,
PRIMARY KEY("id" AUTOINCREMENT)
)
```

Для цього в програмі DB Browser for SQLite я створив базу даних та ввів код, що знаходиться вище.

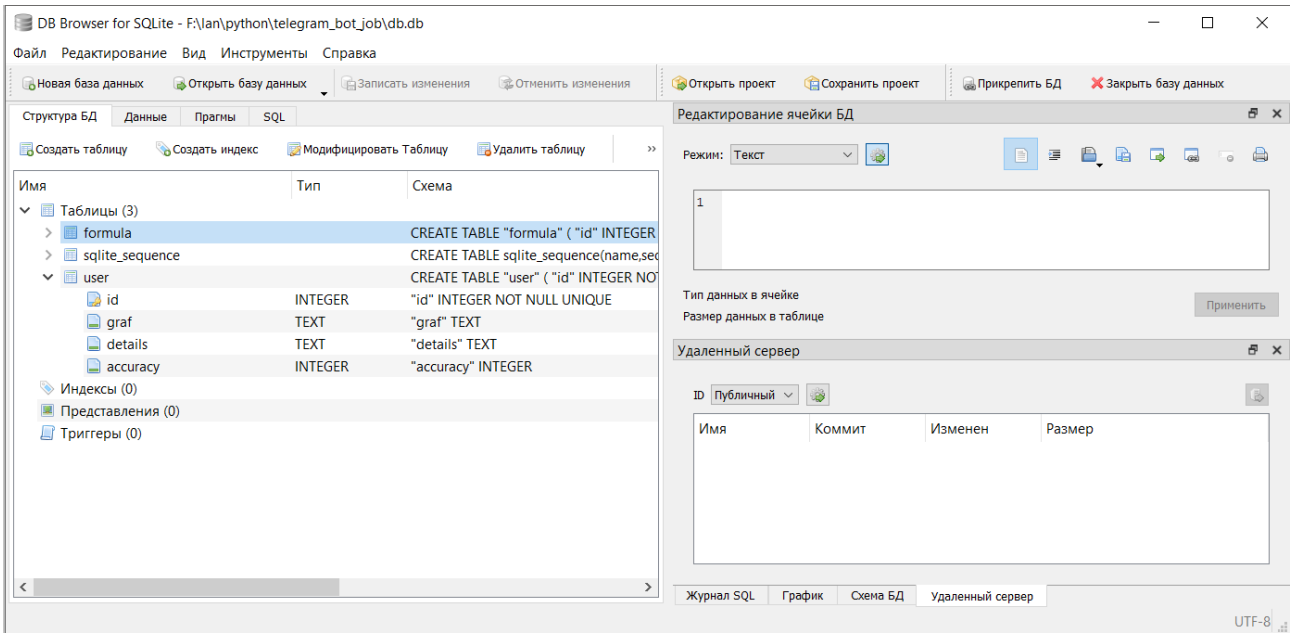

Рисунок 3.5 — Головна сторінка СУБД

3.4 Програмна реалізація

Отримавши Telegram Token з минулого розділу можна приступити до написання код.

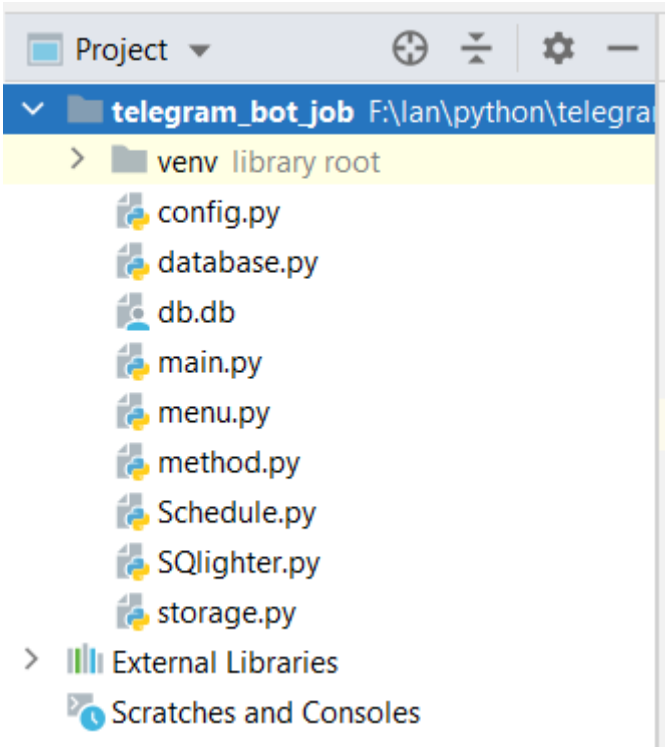

Рисунок 3.6 — Структура проекту

Папка venv зберігає всі використані бібліотеки під час створення чат-бота.

Файл config.py зберігає назви бази даних, Telegram Token, та початкові налаштування.

Основна частина чат-бота знаходиться в main.py. Саме тут відбувається ініціалізація бота:

```
  bot = TeleBot(config.TOKEN)
```

Також, тут знаходиться перше привітання користувача:

```
def info():
    return """Радий Вас вітати. Даний бот допоможе Вам знайти розв'язки задач чисельними методами"""


@bot.message_handler(commands=['start'])
def birth(message):
    bot.send_message(message.chat.id,
                                text=info())
    write_new_user(message.from_user.id)
    menu_main(message.from_user.id)
```

І решта трігерів на дії користувача таких як на натиск кнопки:

```
@bot.callback_query_handler(func=lambda call: True)
def callback_worker(call):
    if call.data == "menu":
        bot.delete_message(call.message.chat.id, call.message.message_id)
        menu_main(call.message.chat.id)
```

Та відправлене повідомлення:

```
@bot.message_handler(content_types=['text'])
def write(message):
    if not message.chat.id in user_stage.keys():
        menu_main(message.chat.id)
        return
    if not message.chat.id in action_user.keys():
        menu_main(message.chat.id)
        return

    if action_user[message.chat.id] == 'accuracy':
        try:
```

Файл menu.py відповідає за відповіді користувачу, коли той натискає певні кнопки:

```
def menu_main(user_id):
    if user_id in user_stage.keys():
        user_stage[user_id] = [None,None]
    keyboard = types.InlineKeyboardMarkup()
    key_fun = types.InlineKeyboardButton(text='Рівняння', callback_data='fun')
    keyboard.add(key_fun)
    key_slar = types.InlineKeyboardButton(text='СЛАР', callback_data='slar')
    keyboard.add(key_slar)
    key_snar = types.InlineKeyboardButton(text='СНАР', callback_data='snar')
    keyboard.add(key_snar)
    key_setting = types.InlineKeyboardButton(text='Налаштування',
callback_data='setting')
    keyboard.add(key_setting)
    bot.send_message(user_id,
                     text='Оберіть тип',
                     reply_markup=keyboard)
```

Файл storage.py має лиш 3 рядка, що відповідають за тимчасові файли, такі як налаштування користувача на сервері, етап його проходження метода і активних користувачів.

Файл Shedule.py відповідає за щоденне очищення тимчасових даних з попереднього файлу.

Файл Sqlighter.py має лиш клас, що відповідає за запити до бази даних.

```
import sqlite3
i
class SQLighter:

    def __init__(self, database):
        self.connection = sqlite3.connect(database)
        self.cursor = self.connection.cursor()

    def write_new_user(self, table_name, id, graf, details, accuracy):
        with self.connection:
            self.cursor.execute(f"""INSERT INTO {table_name} ('id', 'graf',
'details', 'accuracy')
                                    VALUES ({id}, {graf}, {details},
{accuracy})""")
            self.connection.commit()
```

Файл database.py ініціалізує клас Sqlighter, тому є зв’язком між базою даних, де зберігаються налаштування користувачів та назви і відповіді рівнянь

Файл method.py має в собі чисельне вирішення поставлених задач, а також побудову графіків.

3.5 Опис виконаної роботи

Першочергово потрібно знайти бота та запустити його.

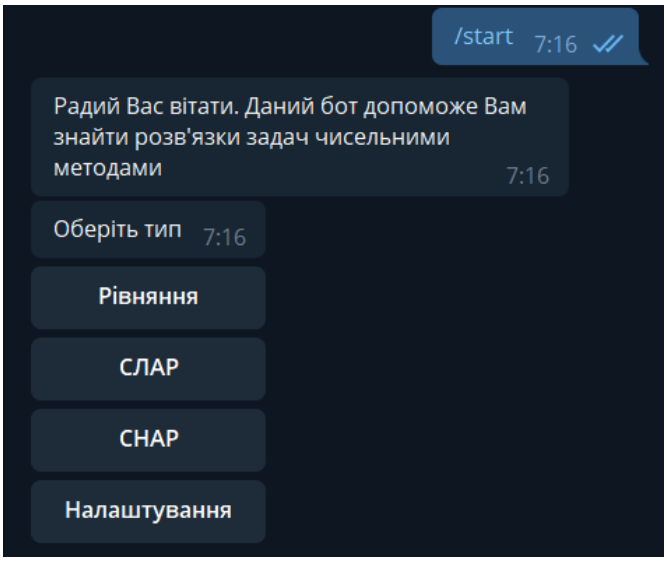

Рисунок 3.7 — Початкова сторінка бота

Користувач отримав доступ до функціоналу бота. Він може обрати вид задачі чи перейти в меню «Налаштування». Перейдем в «Налаштування»

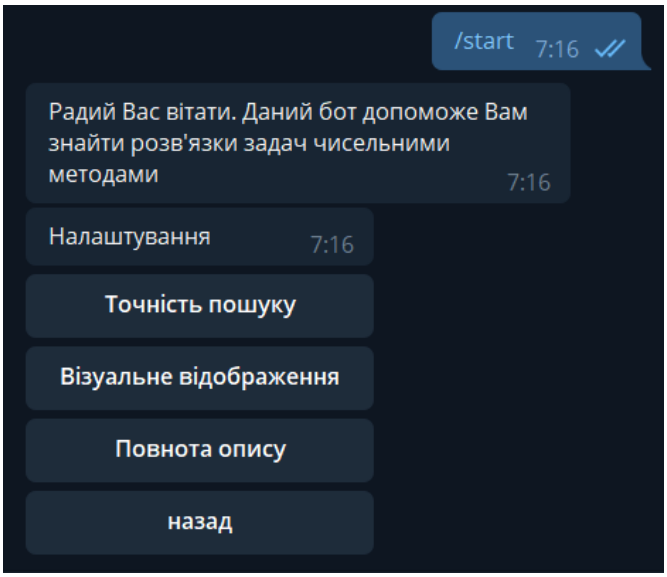

Рисунок 3.8 — Меню «Налаштування»

В цьому меню користувач може обрати чи потрібно йому відображати графіки, чи потрібно розписана відповідь, яка точність використовується для виконання методів. Для точності користувач повинен вести число, кількість знаків після коми. Користувач не може вести точність менше 1.

Так як користувач, щойно підключив бота, то має стандартні налаштування вказані адміністратором:

Таблиця 3.1—Стандартні налаштування користувача

Точність	5
Будувати Графік	Так
Повнота опису	Так

Зараз ми змінимо ці налаштування на точність 3, не будувати графік.

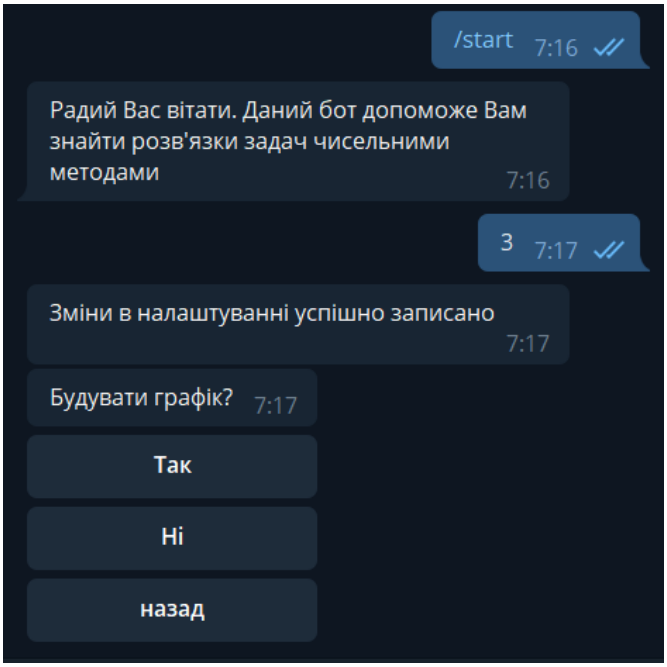

Рисунок 3.9 — Демонстрація змін налаштувань точності

Натискаємо «Ні».

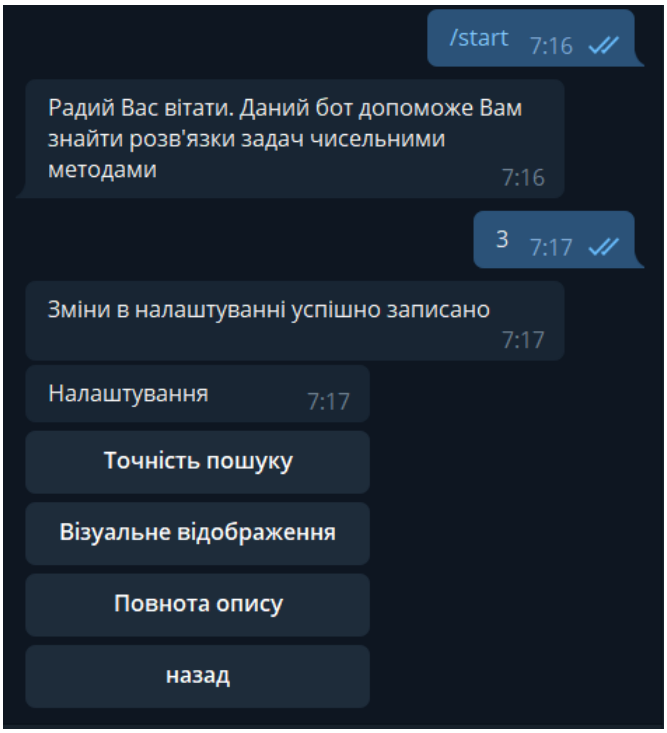

Рисунок 3.10 — Демонстрація змін налаштувань візуального відображення

Після успішного запису нових налаштувань ці дані зберігають в базі даних. Перейдемо до вирішення задачі з 1 рівнянням. Натиснемо на «Рівняння».

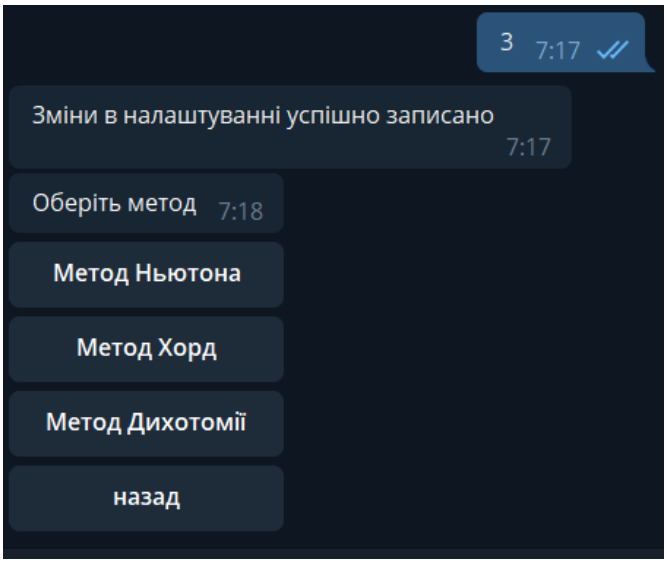

Рисунок 3.11 — Меню «Рівняння»

Перед нами 3 методи вирішення рівняння :

- «Метод Ньютона»
- «Метод Хорд»
- «Метод Дихотомії»

На початку спробуємо «Метод Ньютона». Натискаємо на цю кнопку.

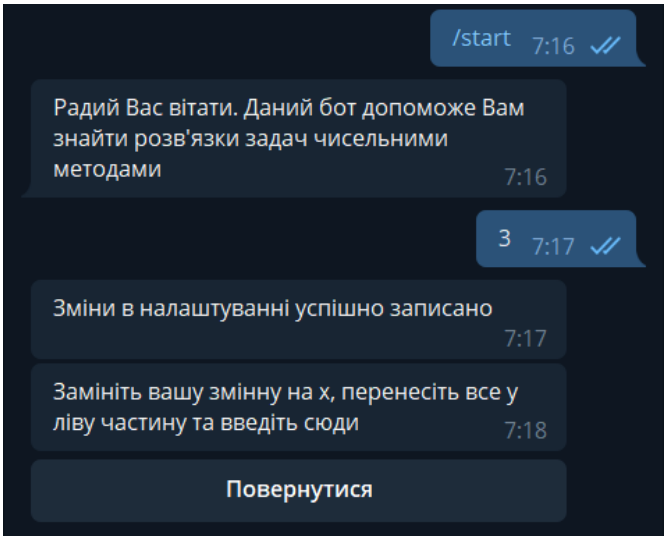

Рисунок 3.12 — Демонстрація впливу налаштувань

Введемо нелінійне рівняння додержуючись інструкції. Наприклад log(x+1) - 4 + x, але припустимо помилку.

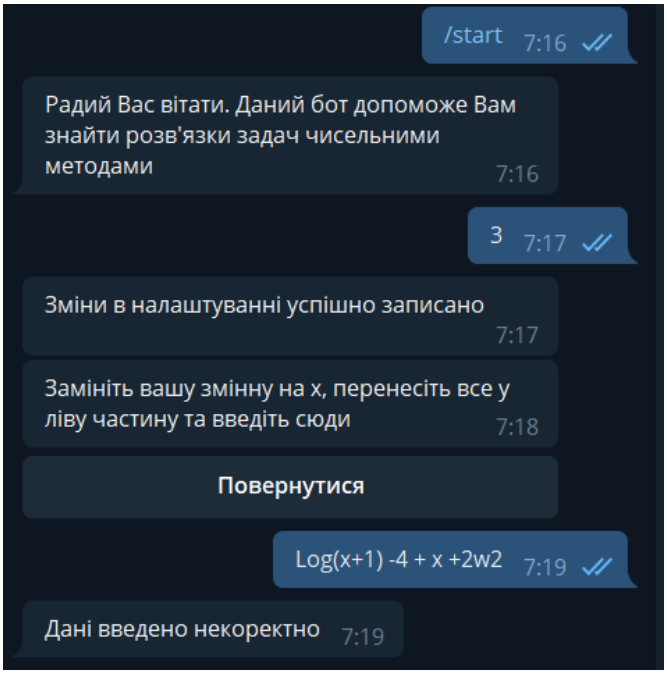

Рисунок 3.13 — Демонстрація попередження користувача

Бот попередить якщо не зможе отримати коректні данні. На цей раз введемо без помилок

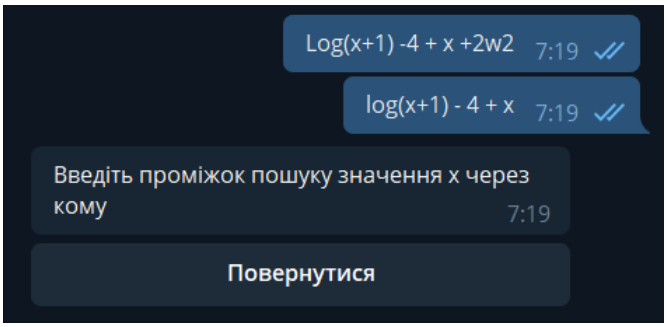

Рисунок 3.14 — Ведення діалогу з користувачем

Так як в «Налаштування» ми відключили графіки і не знаємо проміжок пошуку, то повернемось назад і включимо відображення. Після виконаних дій повторно вводим проміжок.

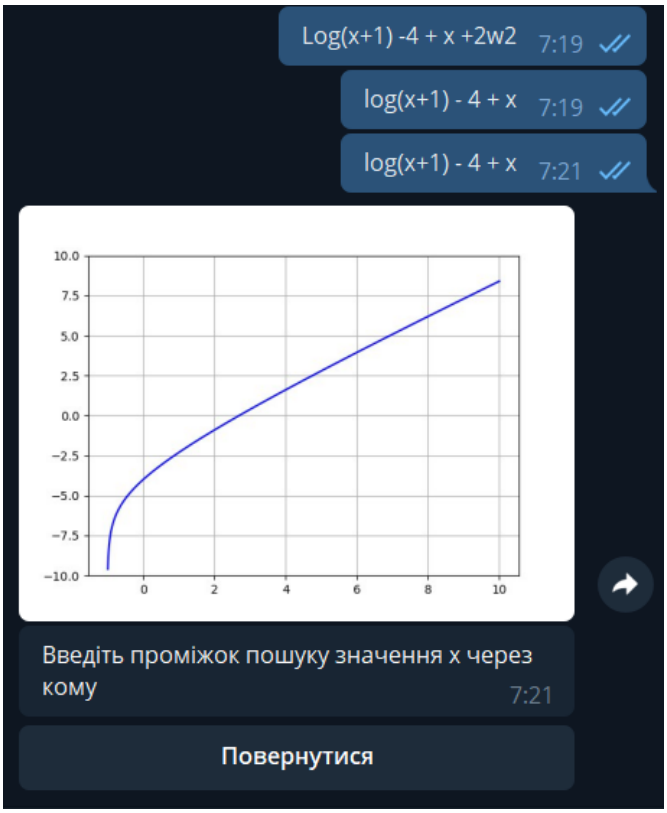

Рисунок 3.15 — Графік функції log(x+1)-4+x

Ми отримали графік. З нього видно, що x знаходиться десь між 2 та 3, введемо ці числа через кому.

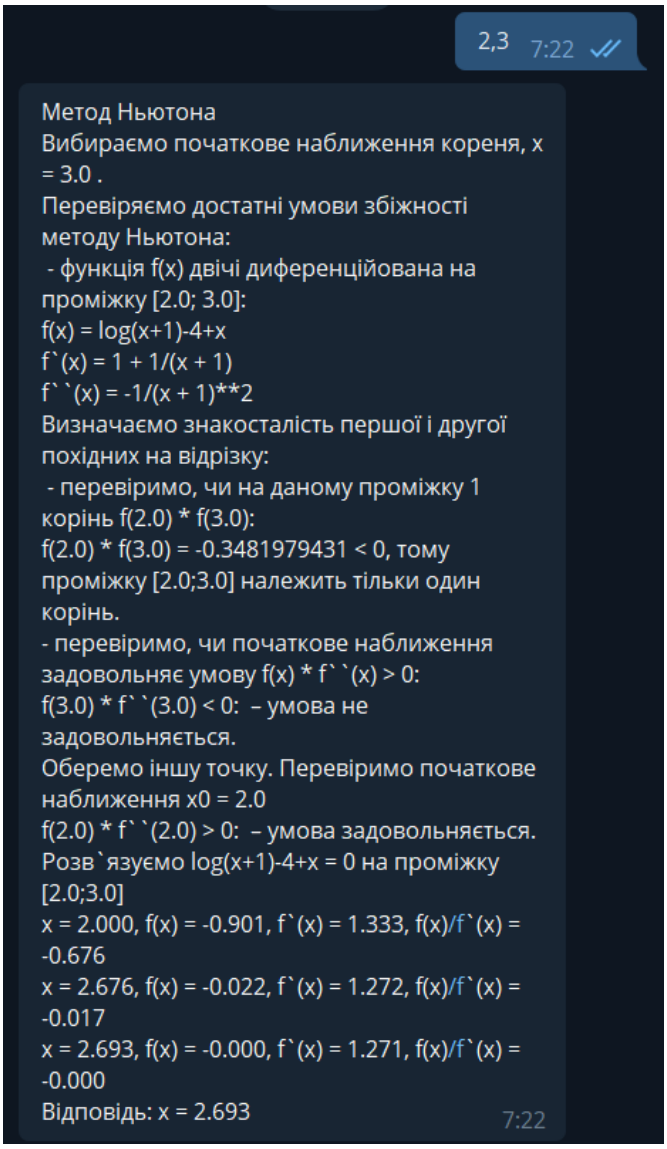

Рисунок 3.16 — Результат методу Ньютона

Після введення проміжку ми отримали результат з повним описом виконанням методу. Спробуємо інший метод для перевірки точності. Після результату вже якраз є «Головне» меню. Натиснемо знову рівняння та виберемо метод Хорд. Введемо знов ліву частину рівняння, проміжок

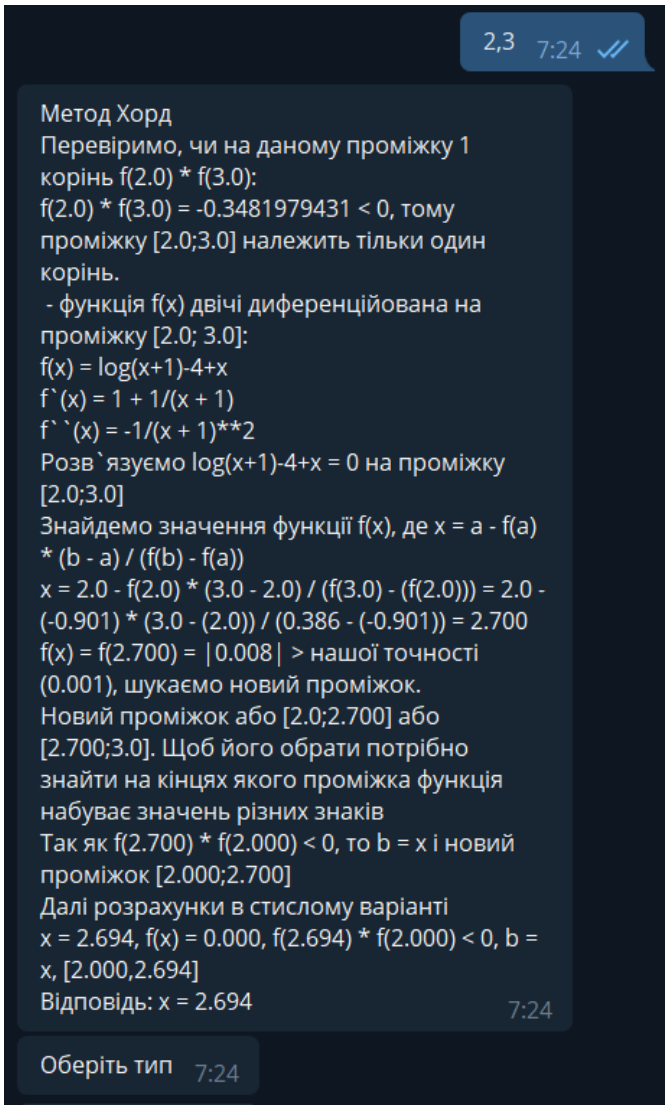

Рисунок 3.17 — Результат методу Хорд

Отримана відповідь не збігається з попередньою на нашу точність. Нічого. Спробуємо останній метод, але напередодні змінимо налаштування на точність 5, не відображати зображення.

Після введення лівої частини рівняння та проміжку пошуку, отримуємо 3 відповідь методом Дихотомії.

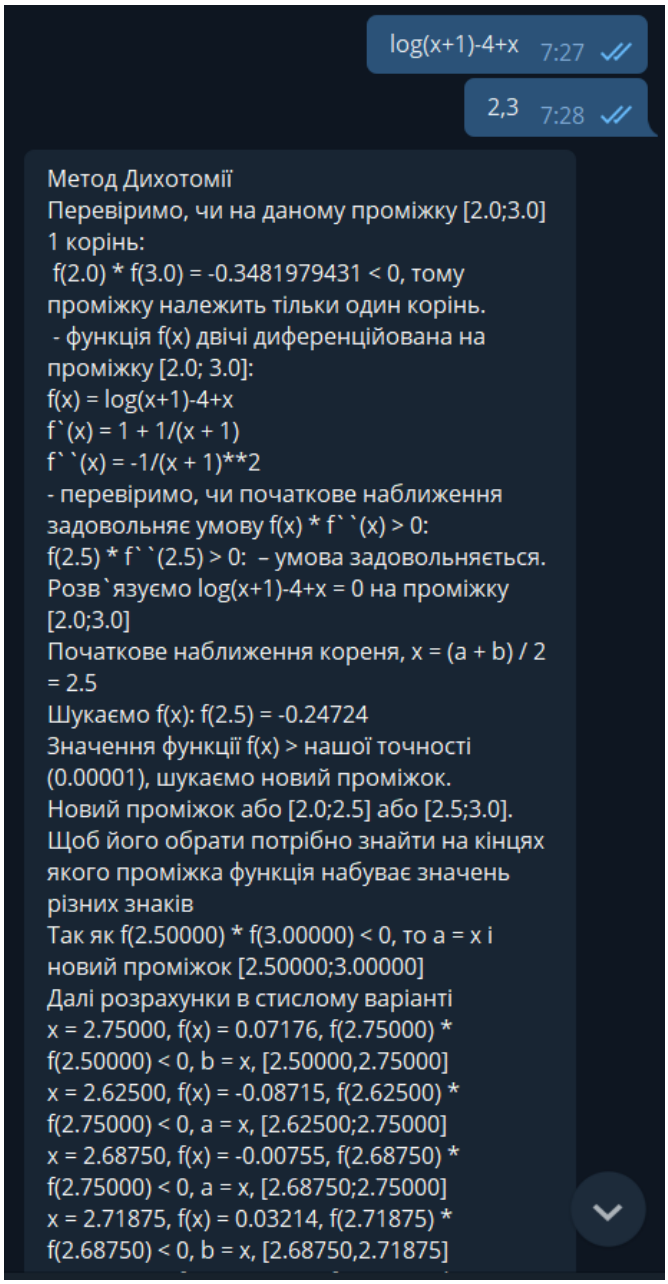

Рисунок 3.18 — Результат методу Дихотомії

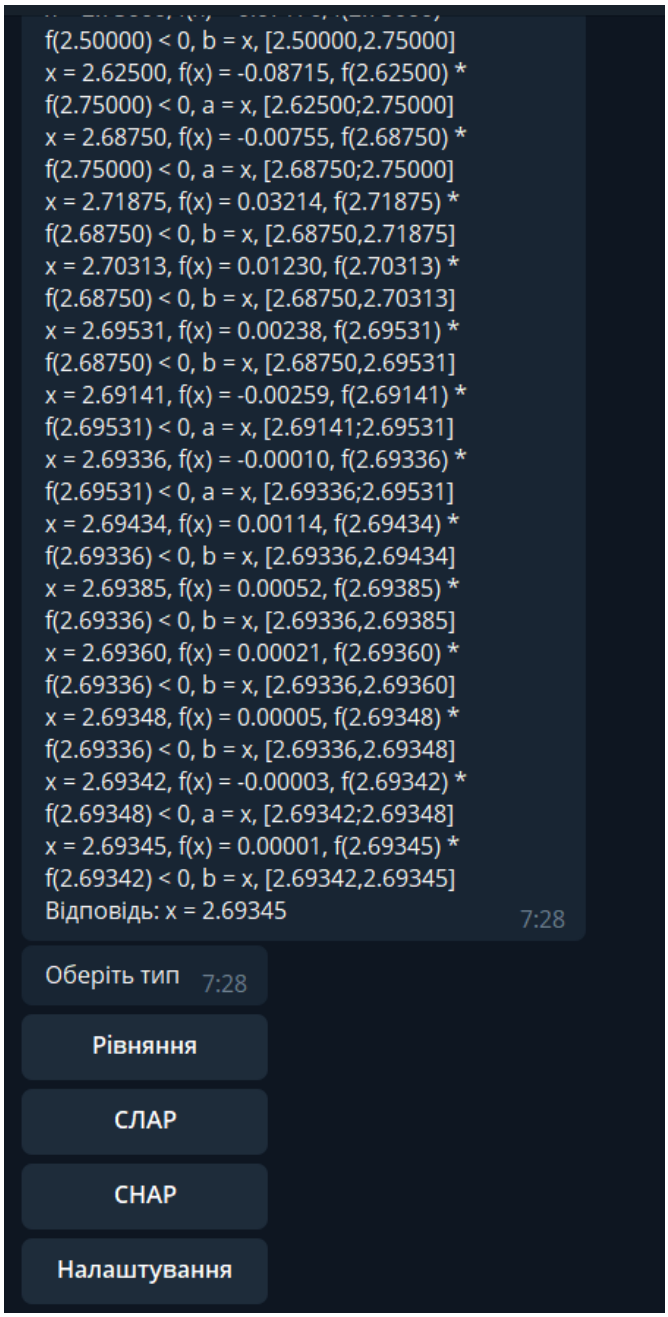

Рисунок 3.19 — Результат метода Дихотомії

З останнього метода видно, що наша відповідь була між 2.69345. Отже 2 попередніх метода були вірними.

Перейдімо з «Головного меню» на СЛАР.

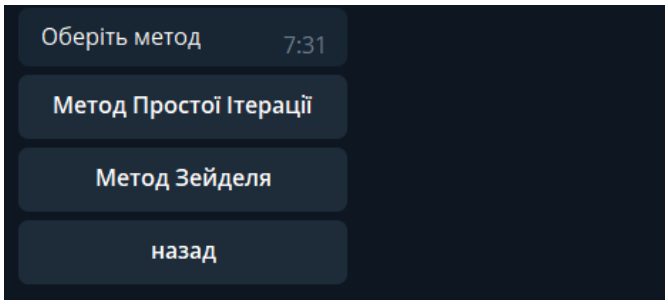

Рисунок 3.20 — Меню «СЛАР»

Натиснемо на «Метод Простої Ітерації»

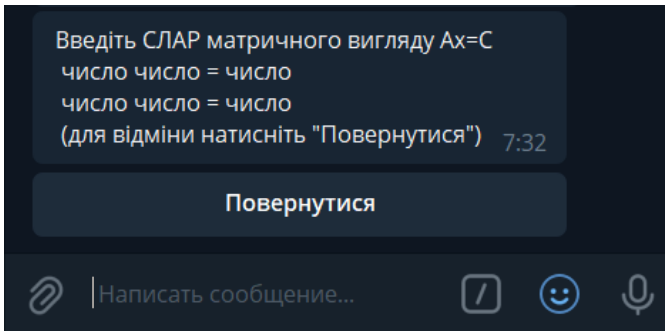

Рисунок 3.21 — Інструкція правильного введення СЛАР

Введемо Систему на натиснемо «Enter»

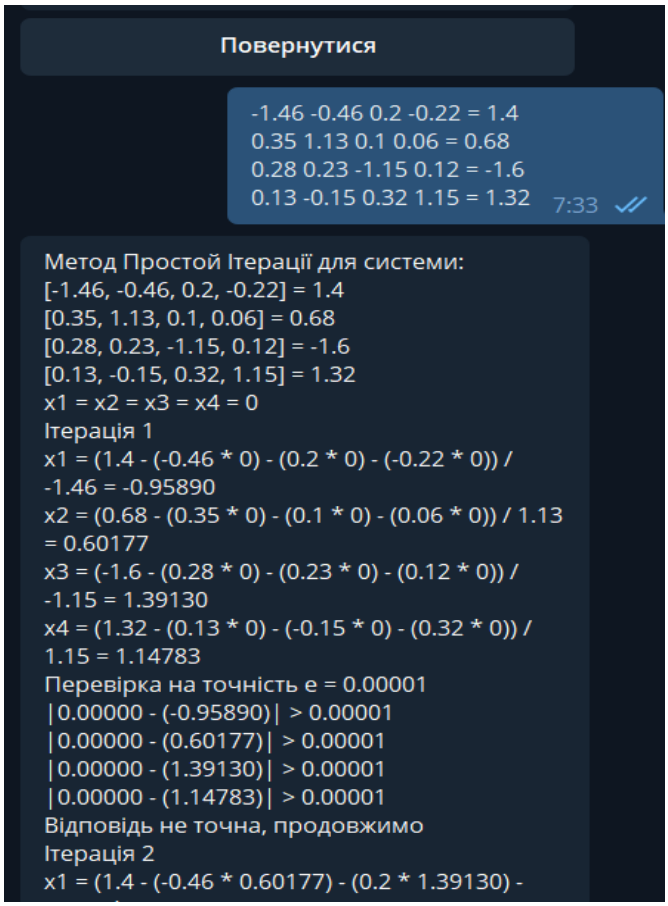

Рисунок 3.22 — Результат методу Простої Ітерації

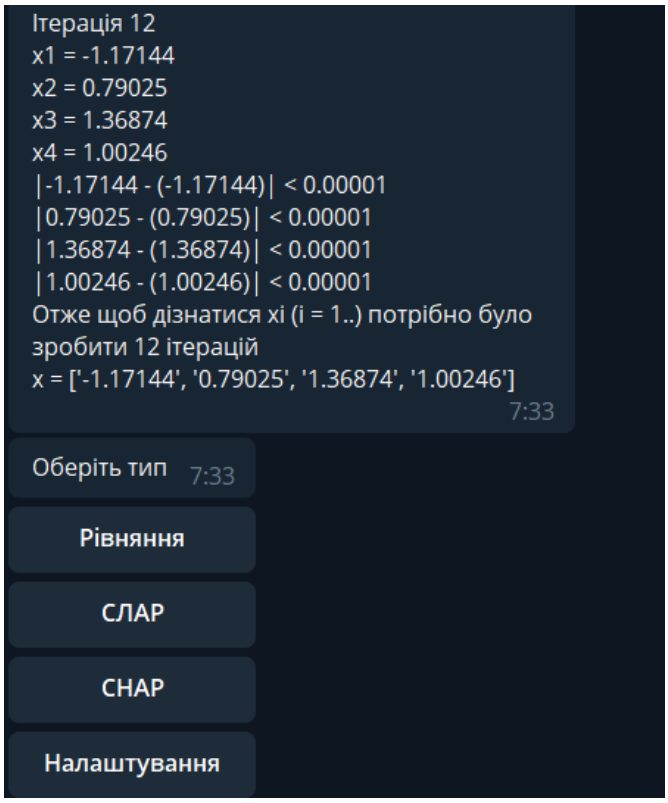

Рисунок 3.23 — Результат метода Простої Ітерації

Відповідь ми отримали, перевіримо її методом Зейделя

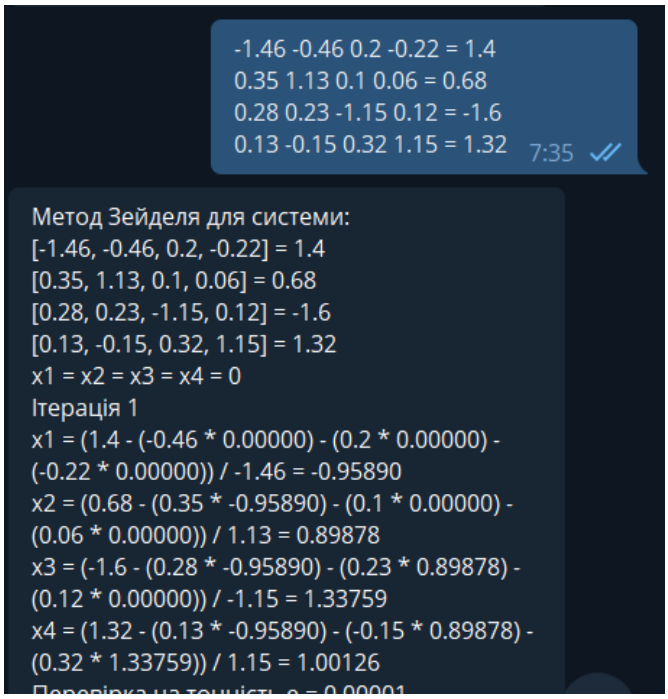

Рисунок 3.24 — Результат методу Зейделя

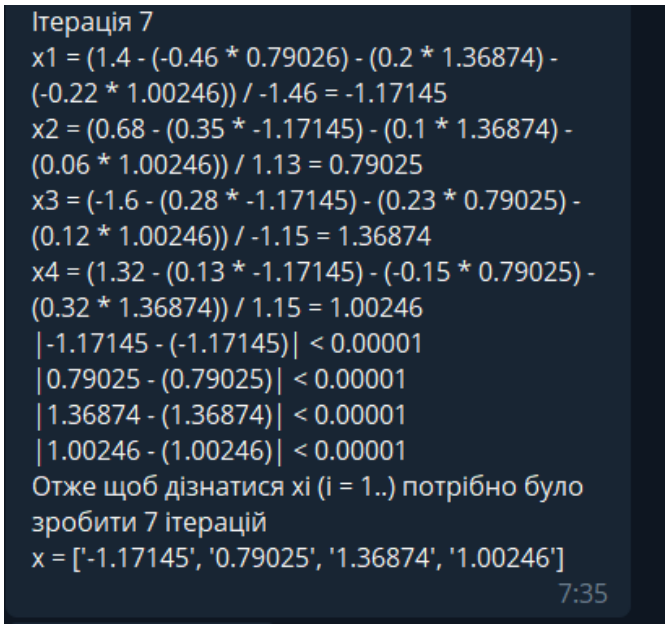

Рисунок 3.25— Результат метода Зейделя

Відповідь ми отримали, збігається з попереднім методом, але є й відмінності. В даному випадку метод Зейделя виконав лише 7 ітерацій.

Спробуємо останній функціонал чат-бота, вирішення СНАР. Натиснемо на «СНАР» .Введемо систему

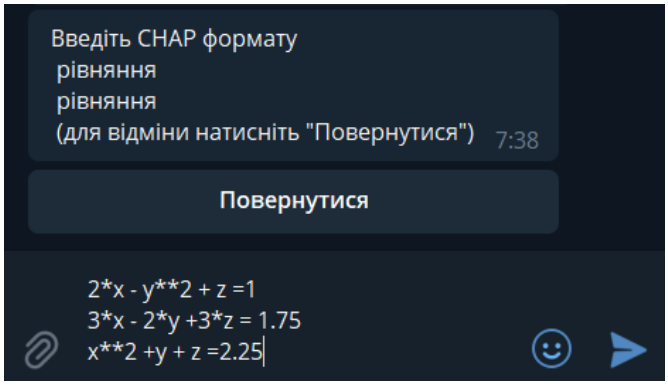

Рисунок 3.26 — Інструкція правильного введення СНАР

Натиснемо «Enter»

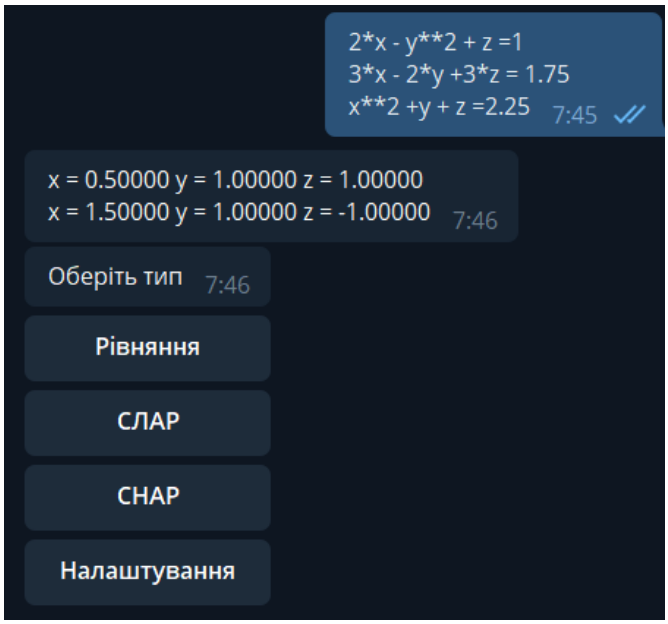

Рисунок 3.27 — Результат пошуку відповіді для СНАР

Отримані відповіді звіримо з калькулятором, підставивши. Дані відповіді дійсно є правильною.

ВИСНОВКИ

Новітні тенденції розповсюдження мобільний пристроїв говорять про актуальність написання кросплатформленого коду, що гарно працював на настільних та персональних пристроях так і на мобільних пристроях. Платформа Telegram ідеально підходить для поставленої задачі, через своє розповсюдження на всіх платформах, популярність та технології ботів.

За основу було взяти мову програмування Python та бібліотеки такі як mathplotlib, sqlite, os. База даних, що зберігає інформацію про налаштування користувачі та графіки є Sqlite.

Для написання програми було використано Середовище програмування Pycharm та DB Browser for SQLite.

У результаті виконання даної роботи було створено чат-бота для месенджера Telegram, що матиме можливість вирішувати задачі чисельними методами.

Даних бот допоможе користувачу опанувати метод вирішення алгебраїчних , що дасть змогу швидше опанувати курс «Чисельні методи» чи просто допомогти користувачеві найти шукану відповідь.

Було обрано декілька найбільш розповсюджених методів пошуку точних значень це метод Ньютона, Хорд, Дихотомії для рівнянь; метод простої ітерації, Зейделя для Слар. Також є код, що вирішує Снар.

При можливості, що користувач забажає повернутися в стрічці до старого повідомлення було вирішено видаляти всі допоміжні діалогові вікна.

Для оптимізації та зменшення кількості невикористаної пам’яті було обрано видаляти тимчасові дані користувача щоденно. В разі повернення користувача через певний період до бота його налаштування зберігають в базі даних.

Даним ботом можуть користуватися як студенти, викладачі так і інші користувачі месенджером Telegram.

СПИСОК ЛІТЕРАТУРИ

1. Gigital 2020 [Електронний ресурс] – Режим доступу: https://wearesocial.com/digital-2020.
2. Gigital 2021 [Електронний ресурс] – Режим доступу: https://wesaresocial.com/digital-2021.
3. Ranked: The World’s Most Downloaded Apps [Електронний ресурс]. – Режим доступу: https://www.visualcapitalist.com/ranked-most-downloaded-apps/.
4. Рейтинг мобільних додатків за січень 2021 [Електронний ресурс]. – Режим доступу: https://tns-ua.com/news/reyting-mobilnih-dodatkiv-za-sichen-2021.
5. Telegram Bot API [Електронний ресурс] – Режим доступу: https://core.telegram.org/bots/api.
6. Почему чатботы захватывают мир [Електронний ресурс] – режим доступу: https://bit.ly/2TEMfJa.
7. Чат-бот : зачем создавать, кому использовать [Електронний ресурс] – Режим доступу:https://blog.click.ru/chat-bot-na-sajte-zachemsozdavat-komu-ispolzovat-i-kak-nastroit/.
8. Лутц М. Программирование на Python, том II, 4-е издание. – Пер. з англ. – СПб.: Символ-Плюс, 2011. – 992 с.
9. Модуль os [Електронний ресурс] – Режим доступу: https://pythonworld.ru/moduli/modul-os.html.
10. Модуль sqlite — Работаем с базой данных [Електронний ресурс] – Режим доступу: https://python-scripts.com/sqlite.
11. DB Browser for SQLite [Електронний ресурс] – Режим доступу: https://sqlitebrowser.org/.
12. Matplotlib: Visualization with Python [Електронний ресурс] – Режим доступу: https://matplotlib.org/.
13. PyCharm: IDE для Python [Електронний ресурс] – Режим доступу: https://timeweb.com/ru/community/articles/pycharm-ide-dlya-python-1

14. pyTelegramBotAPI [Електронний ресурс] – Режим доступу: https://github.com/eternnoir/pyTelegramBotAPI.
15. Инструкция по работе с BotFather ботом [Електронний ресурс] – Режим доступу: https://botcreators.ru/blog/botfather-instrukciya/

ДОДАТОК А

Файл main.py:

```
from time import sleep
from Schedule import SafeScheduler
from threading import Thread
from database import *
from storage import action user, user setting, user stage
from menu import *
from method import *

scheduler = SafeScheduler()

bot = TeleBot(config.TOKEN)

def schedule_checker():
    while True:
        scheduler.run_pending()
        sleep(1)

def every_day():
    try:
        user_setting.clear()

    except Exception as e:
        print(e)


def info():
    return """Радий Вас вітати. Даний бот допоможе Вам знайти корені
заданих функцій чисельними методами"""


@bot.message_handler(commands=['start'])
def birth(message):
    bot.send_message(message.chat.id,
                                text=info())
    write_new_user(message.from_user.id)
    menu_main(message.from_user.id)


@bot.callback_query_handler(func=lambda call: True)
def callback_worker(call):
    if call.data == "menu":
        bot.delete_message(call.message.chat.id, call.message.message_id)
        menu_main(call.message.chat.id)
    elif call.data == "back":
        bot.delete message(call.message.chat.id, call.message.message id)
        menu_main(call.message.chat.id)

    elif "setting" in call.data:
        if call.data == "setting":
            bot.delete_message(call.message.chat.id,
call.message.message_id)
            menu_setting(call.message.chat.id)

        elif call.data == "setting_accuracy":
            action_user[call.message.chat.id] = 'accuracy'
            bot.delete_message(call.message.chat.id,
call.message.message_id)
```

```
            keyboard = types.InlineKeyboardMarkup()
            key_main = types.InlineKeyboardButton(text='Повернутися',
callback_data='setting')
            keyboard.add(key_main)
            bot.send_message(call.message.chat.id,
                             text='Введіть точність пошуку ',
                             reply_markup=keyboard)

        elif call.data == "setting_graf":
            bot.delete_message(call.message.chat.id,
call.message.message_id)
            menu_faq_graf(call.message.chat.id)

        elif call.data == "setting_graf_yes":
            write_new_user_setting(call.message.chat.id, 'graf', True, 0)
            bot.delete_message(call.message.chat.id,
call.message.message_id)
            menu_setting(call.message.chat.id)

        elif call.data == "setting_graf_no":
            write_new_user_setting(call.message.chat.id, 'graf', False, 0)
            bot.delete_message(call.message.chat.id,
call.message.message_id)
            menu_setting(call.message.chat.id)

        elif call.data == "setting_details":
            bot.delete_message(call.message.chat.id,
call.message.message_id)
            menu_faq_details(call.message.chat.id)

        elif call.data == "setting_details_yes":
            write_new_user_setting(call.message.chat.id, 'details', True,
1)
            bot.delete_message(call.message.chat.id,
call.message.message_id)
            menu_setting(call.message.chat.id)

        elif call.data == "setting_details_no":
            write_new_user_setting(call.message.chat.id, 'details', False,
1)
            bot.delete_message(call.message.chat.id,
call.message.message_id)
            menu_setting(call.message.chat.id)

    elif "fun" in call.data:
        if call.data == "fun":
            user_stage[call.message.chat.id] = [None, None]
            bot.delete_message(call.message.chat.id,
call.message.message_id)
            menu_fun(call.message.chat.id)
        elif call.data == "fun_newton":
            action_user[call.message.chat.id] = 'fun_newton'
            bot.delete_message(call.message.chat.id,
call.message.message_id)
            menu_fun_graf(call.message.chat.id)

        elif call.data == "fun_secant":
            action_user[call.message.chat.id] = 'fun_secant'
            bot.delete_message(call.message.chat.id,
call.message.message_id)
            menu_fun_graf(call.message.chat.id)
```

```
        elif call.data == "fun_dichotomy":
            action_user[call.message.chat.id] = 'fun_dichotomy'
            bot.delete_message(call.message.chat.id,
call.message.message_id)
            menu_fun_graf(call.message.chat.id)



    elif "slar" in call.data :
        print(call.data)
        if call.data == "slar":
            user_stage[call.message.chat.id] = [None, None]
            bot.delete_message(call.message.chat.id,
call.message.message_id)
            menu_slar(call.message.chat.id)
        elif call.data == "slar_f":
            action_user[call.message.chat.id] = 'slar_f'
            bot.delete_message(call.message.chat.id,
call.message.message_id)
            menu_slar_dop(call.message.chat.id)

        elif call.data == "slar_seidel":
            action_user[call.message.chat.id] = 'slar_seidel'
            bot.delete_message(call.message.chat.id,
call.message.message_id)
            menu_slar_dop(call.message.chat.id)

    elif call.data == "snar":
        action_user[call.message.chat.id] = "snar"
        user_stage[call.message.chat.id] = [None, "snar"]
        bot.delete_message(call.message.chat.id, call.message.message_id)
        keyboard = types.InlineKeyboardMarkup()
        key_main = types.InlineKeyboardButton(text='Повернутися',
callback_data='menu')
        keyboard.add(key_main)
        bot.send_message(call.message.chat.id,
                         text='Введіть СНАР формату '
                              '\n рівняння'
                              '\n рівняння\n '
                              '(для відміни натисніть "Повернутися")',
                         reply_markup=keyboard)



@bot.message_handler(content_types=['text'])
def write(message):
    print(action_user)
    print(user_stage)
    if not message.chat.id in user_stage.keys():
        menu_main(message.chat.id)
        return
    if not message.chat.id in action_user.keys():
        menu_main(message.chat.id)
        return

    if action_user[message.chat.id] == 'accuracy':
        try:
            accuracy = int(message.text)
            if accuracy <= 0 :
                bot.send_message(message.chat.id,
                                 text=f'Точність не може бути < 0')
                return
```

```
            elif accuracy >= 7:
                accuracy = 7
            write_new_user_setting(message.chat.id, 'accuracy', accuracy,
2)
            bot.delete_message(message.chat.id, message.message_id-1)
            bot.send_message(message.chat.id,
                             text=f'Зміни в налаштуванні успішно
записано')
            menu_setting(message.chat.id)
        except:
            bot.send_message(message.chat.id,
                             text=f'Дані введено некоректно')
    elif "fun" in action_user[message.chat.id]:
        setting = read_user_setting(message.chat.id)
        print(setting)
        if user_stage[message.chat.id][1]  == 'graf':
            try:
                formula = message.text.replace(' ', '').replace('^',
'**').lower()
                if (not 'x' in formula) or 'xx' in formula:
                    bot.send_message(message.chat.id,
                                     text=f'У формулі відсутній x')
                    return

                id_formula, x = read_formula(formula)
                if id_formula is None:
                    write_new_formula(formula)
                    id_formula, x = read_formula(formula)
                if x is None:
                    x = graf(formula, id_formula)
                    write_x(id_formula, x)
                if bool(int(setting[0])):
                    bot.send_photo(message.chat.id,
photo=open(f'{id_formula}.png', 'rb'))
                user_stage[message.chat.id][0] = formula
                bot.delete_message(message.chat.id, message.message_id -
1)
                menu_fun_pr(message.chat.id)
                return
            except:
                bot.send_message(message.chat.id,
                                 text=f'Дані введено некоректно')

        elif user_stage[message.chat.id][1] == 'interval':
            text = message.text.replace(' ', '')
            if not check_a_b(text):
                bot.send_message(message.chat.id,
                                 text=f'Дані введено некоректно')
                return
            text = text.replace(' ','').split(',')
            a, b = float(text[0]), float(text[1])
            bot.delete_message(message.chat.id, message.message_id - 1)

            setting = read_user_setting(message.chat.id)
            ans = None
            if action_user[message.chat.id] == 'fun_newton':
                ans =
fun_newtons_method([user_stage[message.chat.id][0],a,b], setting[2])
            elif action_user[message.chat.id] == 'fun_secant':
                ans =
fun_secant_method([user_stage[message.chat.id][0],a,b], setting[2])
            elif action_user[message.chat.id] == 'fun_dichotomy':
```

```
                ans =
fun_dichotomy_method([user_stage[message.chat.id][0],a,b], setting[2])
            if ans is not None:
                answer = ''.join(ans)
                bot.send_message(message.chat.id,
                                 text=f'{answer}')
                menu_main(message.chat.id)


    elif "slar" in action_user[message.chat.id]:
        try:
            if user_stage[message.chat.id][1]  == 'slar':
                setting = read_user_setting(message.chat.id)
                print(setting)
                ans = check_slar(message.text)
                if ans is None:
                    bot.send_message(message.chat.id,
                                     text=f'Дані введено некоректно')
                    return

                a, c = ans[0], ans[1]
                if action_user[message.chat.id] == "slar_f":
                    ans = slat_f(a,c,setting[2])
                    if ans is None:
                        bot.send_message(message.chat.id,
                                         text=f'Дані введено некоректно')
                        return
                    else:
                        ans = ''.join(ans)
                        bot.send_message(message.chat.id,
                                         text=f'{ans}')
                    menu_main(message.chat.id)


                elif action_user[message.chat.id] == "slar_seidel":
                    ans = slar_seidel(a, c, setting[2])
                    if ans is None:
                        bot.send_message(message.chat.id,
                                         text=f'Дані введено некоректно')
                        return
                    else:
                        ans = ''.join(ans)
                        bot.send_message(message.chat.id,
                                         text=f'{ans}')
                    menu_main(message.chat.id)
        except:
            bot.send_message(message.chat.id,
                             text=f'Дані введено некоректно')
            return
    elif "snar" in action_user[message.chat.id]:
        if user_stage[message.chat.id][1]  == 'snar':

            setting = read_user_setting(message.chat.id)

            ans = snar(message.text, setting[2])
            if ans is None:
                bot.send_message(message.chat.id,
                                 text=f'Дані введено некоректно')
                return
            else:
                answer = ''.join(ans)
                bot.send_message(message.chat.id,
```

```
                                text=f'{answer}')
            menu_main(message.chat.id)


if __name__ == '__main__':
    scheduler.every().day.at('03:00').do(every_day)
    Thread(target=schedule_checker).start()
    bot.polling(none_stop=False)
```

Файл menu.py:

```
from telebot import TeleBot, types
from config import TOKEN
bot = TeleBot(TOKEN)
from storage import  user_stage


def menu_main(user_id):
    if user_id in user_stage.keys():
        user_stage[user_id] = [None,None]
    keyboard = types.InlineKeyboardMarkup()
    key_fun = types.InlineKeyboardButton(text='Рівняння',
callback_data='fun')
    keyboard.add(key_fun)
    key_slar = types.InlineKeyboardButton(text='СЛАР',
callback_data='slar')
    keyboard.add(key_slar)
    key_snar = types.InlineKeyboardButton(text='СНАР',
callback_data='snar')
    keyboard.add(key_snar)
    key_setting = types.InlineKeyboardButton(text='Налаштування',
callback_data='setting')
    keyboard.add(key_setting)
    bot.send_message(user_id,
                     text='Оберіть тип',
                     reply_markup=keyboard)

def menu_fun(user_id):
    keyboard = types.InlineKeyboardMarkup()
    key_newton = types.InlineKeyboardButton(text='Метод Ньютона',
callback_data='fun_newton')
    keyboard.add(key_newton)
    key_secant = types.InlineKeyboardButton(text='Метод Хорд',
callback_data='fun_secant')
    keyboard.add(key_secant)
    key_dichotomy = types.InlineKeyboardButton(text='Метод Дихотомії',
callback_data='fun_dichotomy')
    keyboard.add(key_dichotomy)
    key_back = types.InlineKeyboardButton(text='назад',
callback_data='back')
    keyboard.add(key_back)
    bot.send_message(user_id,
                     text='Оберіть метод',
                     reply_markup=keyboard)

def menu_slar(user_id):
    keyboard = types.InlineKeyboardMarkup()
    key_f = types.InlineKeyboardButton(text='Метод Простої Ітерації',
callback_data='slar_f')
    keyboard.add(key_f)
    key_seidel = types.InlineKeyboardButton(text='Метод Зейделя',
callback_data='slar_seidel')
    keyboard.add(key_seidel)
```

```
    key_back = types.InlineKeyboardButton(text='назад',
callback_data='back')
    keyboard.add(key_back)
    bot.send_message(user_id,
                     text='Оберіть метод',
                     reply_markup=keyboard)


def menu_setting(user_id):
    keyboard = types.InlineKeyboardMarkup()
    key_accuracy = types.InlineKeyboardButton(text='Точність пошуку',
callback_data='setting_accuracy')
    keyboard.add(key_accuracy)
    key_chart = types.InlineKeyboardButton(text='Візуальне відображення',
callback_data='setting_graf')
    keyboard.add(key_chart)
    key_details = types.InlineKeyboardButton(text='Повнота опису',
callback_data='setting_details')
    keyboard.add(key_details)
    key_back = types.InlineKeyboardButton(text='назад',
callback_data='back')
    keyboard.add(key_back)
    bot.send_message(user_id,
                     text='Налаштування',
                     reply_markup=keyboard)

def menu_faq_graf(user_id):
    keyboard = types.InlineKeyboardMarkup()
    key_accuracy = types.InlineKeyboardButton(text='Так',
callback_data='setting_graf_yes')
    keyboard.add(key_accuracy)
    key_chart = types.InlineKeyboardButton(text='Ні',
callback_data='setting_graf_no')
    keyboard.add(key_chart)
    key_back = types.InlineKeyboardButton(text='назад',
callback_data='setting')
    keyboard.add(key_back)
    bot.send_message(user_id,
                     text='Будувати графік?',
                     reply_markup=keyboard)

def menu_faq_details(user_id):
    keyboard = types.InlineKeyboardMarkup()
    key_accuracy = types.InlineKeyboardButton(text='Так',
callback_data='setting_details_yes')
    keyboard.add(key_accuracy)
    key_chart = types.InlineKeyboardButton(text='Ні',
callback_data='setting_details_no')
    keyboard.add(key_chart)
    key_back = types.InlineKeyboardButton(text='назад',
callback_data='setting')
    keyboard.add(key_back)
    bot.send_message(user_id,
                     text='Докладна відповідь',
                     reply_markup=keyboard)

def menu_fun_graf(user_id):
    if user_id in user_stage.keys():
        user_stage[user_id][1] = 'graf'
        keyboard = types.InlineKeyboardMarkup()
        key_main = types.InlineKeyboardButton(text='Повернутися',
callback_data='menu')
```

```
        keyboard.add(key_main)
        bot.send_message(user_id,
                                text='Замініть вашу змінну на x, перенесіть
все у ліву частину та введіть сюди',
                         reply_markup=keyboard)
    else:
        menu_main(user_id)


def menu_fun_pr(user_id):
    if user_id in user_stage.keys():
        user_stage[user_id][1] = 'interval'
        keyboard = types.InlineKeyboardMarkup()
        key_main = types.InlineKeyboardButton(text='Повернутися',
callback_data='menu')
        keyboard.add(key_main)
        bot.send_message(user_id,
                         text='Введіть проміжок пошуку значення x через
кому',
                         reply_markup=keyboard)
    else:
        menu_main(user_id)

def menu_slar_dop(user_id):
    if user_id in user_stage.keys():
        user_stage[user_id][1] = 'slar'
        keyboard = types.InlineKeyboardMarkup()
        key_main = types.InlineKeyboardButton(text='Повернутися',
callback_data='menu')
        keyboard.add(key_main)
        bot.send_message(user_id,
                         text='Введіть СЛАР матричного вигляду Ax=C '
                              '\n число число = число'
                              '\n число число = число\n '
                              '(для відміни натисніть "Повернутися")',
                         reply_markup=keyboard)
    else:
        menu_main(user_id)
```

Файл method.py:

```
import numpy as np
import matplotlib.pyplot as plt
from sympy import *
import sympy
import decimal
from decimal import Decimal as D

def check_a_b(text):
    try:
        text = text.replace(' ', '').split(',')
        if len(text) != 2 or float(text[0]) >= float(text[1]):
            return False
        return True
    except Exception as e:
        print(e)
        return False

def check_slar(text):
    try:
        text = text.replace('   ', '  ').replace('  ', ' ').replace('  ',
' ').split('\n')
        if len(text) <= 1:
```

```
            return None
        c = []
        a = []

        for i in range(len(text)):
            f = text[i].split('=')
            f1 = [float(j) for j in f[0].split(' ') if j is not '' and
float(j) != 0]
            c.append(float(f[1]))
            if len(f1) < len(text):
                return None
            else:
                a.append(f1[0:4])

        if len(a) != len(c):
            return None
        return [a,c]
    except:
        return None

def der1(eq, x):
    return dround(N(str(diff(eq)).replace('x', str(x))))

def der2(eq, x):
    return dround(N(str(diff(diff(eq))).replace('x', str(x))))

def F(eq, x):
    return dround(N(eq.replace('x', str(x))))

def dround(d, ndigits = 5, rounding=decimal.ROUND_HALF_UP):
    result = D(str(d)).quantize(D('0.1')**ndigits, rounding=rounding)
    return result

def slat_f(a,c, accuracy):
    try:
        t = 1 / (10 ** accuracy)
        n = 1
        answer = []
        answer.append('Метод Простой Ітерації для системи:\n')
        for _ in range(len(c)):
            answer.append(f'{a[_]} = {c[_]} \n')
        for _ in range(len(c)):
            answer.append(f'x{_+1} = ')

        answer.append('0\n')

        x = [0 for _ in range(len(c))]

        answer.append(f'Ітерація {n}\n')
        x_last = x.copy()
        for i in range(len(c)):
            xn = x_last.copy()
            a_last = a[i].copy()
            xn[i] = 0
            x[i] = c[i]
            answer.append(f'x{i + 1} = ({c[i]}')
            for j in range(len(c)):
                if j == i:
                    continue
                x[i] = x[i] - a_last[j] * xn[j]
                answer.append(f' - ({a_last[j]} * {xn[j]})')
            x[i] = x[i] / a_last[i]
```

```
            answer.append(f') / {a_last[i]} = {dround(x[i], accuracy)}\n')
        answer.append(f'Перевірка на точність e = {dround(t,accuracy)}\n')

        s = True
        for i in range(len(c)):
            if abs(x_last[i] - x[i]) < t:
                answer.append(f'|{dround(x_last[i],accuracy)} -
({dround(x[i],accuracy)})| < {dround(t,accuracy)}\n')
            else:
                s = False
                answer.append(f'|{dround(x_last[i],accuracy)} -
({dround(x[i],accuracy)})| > {dround(t,accuracy)}\n')

        if s:
            answer.append(f'Отже щоб дізнатися xi (i = 1..) потрібно було
зробити {n} ітерацій')
            return answer
        else:
            answer.append(f'Відповідь не точна, продовжимо\n')


        n += 1
        answer.append(f'Ітерація {n}\n')
        x_last = x.copy()
        for i in range(len(c)):
            xn = x_last.copy()
            a_last = a[i].copy()
            xn[i] = 0
            x[i] = c[i]
            answer.append(f'x{i + 1} = ({c[i]}')
            for j in range(len(c)):
                if j == i:
                    continue
                x[i] = x[i] - a_last[j] * xn[j]
                answer.append(f' - ({a_last[j]} * {dround(xn[j],
accuracy)})')
            x[i] = x[i] / a_last[i]
            # print(x[i])
            answer.append(f') / {a_last[i]} = {dround(x[i], accuracy)}\n')

        s = True
        for i in range(len(c)):
            if abs(x_last[i] - x[i]) < t:
                answer.append(f'x{i+1} |{dround(x_last[i],accuracy)} -
({dround(x[i],accuracy)})| < {dround(t, accuracy)}\n')
            else:
                s = False
                answer.append(f'|{dround(x_last[i],accuracy)} -
({dround(x[i],accuracy)})| > {dround(t, accuracy)}\n')

        if s:
            answer.append(f'Отже щоб дізнатися xi (i = 1..) потрібно було
зробити {n} ітерацій')
            return answer
        else:
            answer.append(f'Відповідь не точна, продовжимо\n')



        while True:
            n +=1
            answer.append(f'Ітерація {n}\n')
```

```
            x_last = x.copy()
            for i in range(len(c)):
                xn = x_last.copy()
                a_last = a[i].copy()
                xn[i] = 0
                x[i] = c[i]

                for j in range(len(c)):
                    if j == i:
                        continue
                    x[i] = x[i] - a_last[j]*xn[j]

                answer.append(f'x{i + 1} = ')
                x[i] = x[i] / a_last[i]
                answer.append(f'{dround(x[i],accuracy)}\n')

            s = True
            for i in range(len(c)):
                if abs(x_last[i] - x[i]) < t:
                    answer.append(f'|{dround(x_last[i],accuracy)} -
({dround(x[i],accuracy)})| < {dround(t, accuracy)}\n')
                else:
                    s = False
                    answer.append(f'|{dround(x_last[i],accuracy)} -
({dround(x[i],accuracy)})| > {dround(t, accuracy)}\n')

            if s:
                answer.append(f'Отже щоб дізнатися xi (i = 1..) потрібно
було зробити {n} ітерацій \n'
                              f'x = {[str(dround(i,accuracy)) for i in
x]}')
                return answer

            if n == 20:
                return answer
    except:
        return None

def slar_seidel(a,c, accuracy):
    try:
        t = 1 / (10 ** accuracy)
        n = 1
        answer = []
        answer.append('Метод Зейделя для системи:\n')
        for _ in range(len(c)):
            answer.append(f'{a[_]} = {c[_]} \n')
        for _ in range(len(c)):
            answer.append(f'x{_+1} = ')

        answer.append('0\n')

        x = [0 for _ in range(len(c))]

        answer.append(f'Ітерація {n}\n')
        x_last = []
        for i in range(len(c)):
            x_last = x.copy()
            a_last = a[i].copy()
            x[i] = c[i]
            answer.append(f'x{i + 1} = ({c[i]}')
            for j in range(len(c)):
                if j == i:
```

```
                    continue
                x[i] = x[i] - a_last[j] * x_last[j]
                answer.append(f' - ({a_last[j]} *
{dround(x_last[j],accuracy)})')
            x[i] = x[i] / a_last[i]
            answer.append(f') / {a_last[i]} = {dround(x[i], accuracy)}\n')
        answer.append(f'Перевірка на точність e = {dround(t,accuracy)}\n')

        s = True
        for i in range(len(c)):
            if abs(x_last[i] - x[i]) < t:
                answer.append(f'|{dround(x_last[i],accuracy)} -
({dround(x[i],accuracy)})| < {dround(t,accuracy)}\n')
            else:
                s = False
                answer.append(f'|{dround(x_last[i],accuracy)} -
({dround(x[i],accuracy)})| > {dround(t,accuracy)}\n')

        if s:
            answer.append(f'Отже щоб дізнатися xi (i = 1..) потрібно було
зробити {n} ітерацій')
            return answer
        else:
            answer.append(f'Відповідь не точна, продовжимо\n')


        n += 1
        answer.append(f'Ітерація {n}\n')
        for i in range(len(c)):
            x_last = x.copy()
            a_last = a[i].copy()
            x[i] = c[i]
            answer.append(f'x{i + 1} = ({c[i]}')
            for j in range(len(c)):
                if j == i:
                    continue
                x[i] = x[i] - a_last[j] * x_last[j]
                answer.append(f' - ({a_last[j]} * {dround(x_last[j],
accuracy)})')
            x[i] = x[i] / a_last[i]
            answer.append(f') / {a_last[i]} = {dround(x[i], accuracy)}\n')


        s = True
        for i in range(len(c)):
            if abs(x_last[i] - x[i]) < t:
                answer.append(f'x{i+1} |{dround(x_last[i],accuracy)} -
({dround(x[i],accuracy)})| < {dround(t, accuracy)}\n')
            else:
                s = False
                answer.append(f'|{dround(x_last[i],accuracy)} -
({dround(x[i],accuracy)})| > {dround(t, accuracy)}\n')

        if s:
            answer.append(f'Отже щоб дізнатися xi (i = 1..) потрібно було
зробити {n} ітерацій')
            return answer
        else:
            answer.append(f'Відповідь не точна, продовжимо\n')
```

```
        while True:
            n +=1
            answer.append(f'Ітерація {n}\n')
            x_last = x.copy()
            for i in range(len(c)):
                x_last = x.copy()
                a_last = a[i].copy()
                x[i] = c[i]
                answer.append(f'x{i + 1} = ({c[i]}')
                for j in range(len(c)):
                    if j == i:
                        continue
                    x[i] = x[i] - a_last[j] * x_last[j]
                    answer.append(f' - ({a_last[j]} * {dround(x_last[j],
accuracy)})')
                x[i] = x[i] / a_last[i]
                answer.append(f') / {a_last[i]} = {dround(x[i],
accuracy)}\n')

            s = True
            for i in range(len(c)):
                if abs(x_last[i] - x[i]) < t:
                    answer.append(f'|{dround(x_last[i],accuracy)} -
({dround(x[i],accuracy)})| < {dround(t, accuracy)}\n')
                else:
                    s = False
                    answer.append(f'|{dround(x_last[i],accuracy)} -
({dround(x[i],accuracy)})| > {dround(t, accuracy)}\n')

            if s:
                answer.append(f'Отже щоб дізнатися xi (i = 1..) потрібно
було зробити {n} ітерацій \n'
                              f'x = {[str(dround(i,accuracy)) for i in
x]}')
                return answer

            if n == 20:
                return answer
    except:
        return None

def snar(text, accuracy):
    try:
        text = text.replace('   ', '  ').replace('  ', ' ').replace('  ',
' ').split('\n')
        if len(text) <= 1:
            return None
        print(text)
        for i in range(len(text)):
            text[i] = text[i].replace('=', '-(') + ')'
        print(text)
        ans = solve(text)
        print(ans)
        answer = []
        for i in range(len(ans)):

            for j in ans[i]:
                ans[i][j] = round(ans[i][j],accuracy)
                answer.append(f'{j} = {dround(ans[i][j],accuracy)} ')
            answer.append(f'\n')
        print(answer)
        return answer
```

```
    except:
        return None


#print(snar('2*x - y**2 + z = 1\n 3*x**2 - 2*y +3*z = 1.75\n x**2 +y + z =
2.25',5))

def fun_newtons_method(info, accuracy):
    answer = []
    try:
        eq, a, b = info[0], info[1], info[2]
        answer.append(f'Метод Ньютона\n'
                      f'Вибираємо початкове наближення кореня, x = {b}
.\n'
                      f'Перевіряємо достатні умови збіжності методу
Ньютона:\n')

        if diff(diff(eq)) != 0:
            answer.append(f' - функція f(x) двічі диференційована на
проміжку [{a}; {b}]:\n')
            answer.append(f'f(x) = {eq}\n'
                          f'f`(x) = {diff(eq)}\n'
                          f'f``(x) = {diff(diff(eq))}\n')
        else:
            answer.append(f' - функція f(x) двічі не диференційована на
проміжку [{a}; {b}], тому майбутні обрахунки неможливі:\n')
            answer.append(f'f(x) = {eq}\n'
                          f'f`(x) = {diff(eq)}\n'
                          f'f``(x) = {diff(diff(eq))}\n')
            return answer

        answer.append(f'Визначаємо знакосталість першої і другої похідних
на відрізку:\n')
        answer.append(f' - перевіримо, чи на даному проміжку 1 корінь
f({a}) * f({b}):\n')

        if F(eq, a) * F(eq, b) < 0:
            answer.append(f'f({a}) * f({b}) = {F(eq, a) * F(eq, b)} < 0,
тому проміжку [{a};{b}] належить тільки один корінь.\n')
        else:
            answer.append(f'f({a}) * f({b}) = {F(eq, a) * F(eq, b)} > 0,
тому проміжку [{a};{b}] належить або декілька коренів або жодного. Оберіть
інший проміжок.\n')
            return answer

        answer.append(f'- перевіримо, чи початкове наближення задовольняє
умову f(x) * f``(x) > 0:\n')

        if F(eq, b) * der2(eq, b) > 0:
            answer.append(f'f({b}) * f``({b}) > 0:  - умова
задовольняється.\n')
        else:
            answer.append(f'f({b}) * f``({b}) < 0:  - умова не
задовольняється.\n'
                          f'Оберемо іншу точку. Перевіримо початкове
наближення x0 = {a}\n')
            if F(eq, a) * der2(eq, a) > 0:
                answer.append(f'f({a}) * f``({a}) > 0:  - умова
задовольняється.\n')
                b, a = a, b
            else:
                answer.append(f'f({a}) * f``({a}) < 0:  - умова не
```

```
задовольняється.\n')
                return answer

        x0 = dround(b, accuracy)
        xn = F(eq, x0)
        xd = der1(eq, x0)
        xn1 = xn / xd

        answer.append(f'Розв`язуємо {eq} = 0 на проміжку
[{info[1]};{info[2]}]\n'
                      f'x = {dround(x0, accuracy)}, '
                      f'f(x) = {dround(xn,accuracy)}, '
                      f'f`(x) = {dround(xd,accuracy)}, '
                      f'f(x)/f`(x) = {dround(xn1,accuracy)}\n')

        ans, c, t = x0, 1, 1 / (10 ** accuracy)

        while abs(x0 - xn1) >= t and c != 20 and
float(dround(xn1,accuracy)) != 0 and float(dround(xn,accuracy)) != 0:
            c +=1

            x0 = dround(x0 - xn1)
            xn = F(eq, x0)
            xd = der1(eq, x0)
            xn1 = xn / xd
            answer.append(f'x = {dround(x0,accuracy)}, '
                          f'f(x) = {dround(xn,accuracy)}, '
                          f'f`(x) = {dround(xd,accuracy)}, '
                          f'f(x)/f`(x) = {dround(xn1,accuracy)}\n')
            ans = x0

        answer.append(f'Відповідь: x = {dround(ans,accuracy)}')
    except Exception as e:
        answer.append(f'Не можу виконати обрахунки на даному проміжку.\n')
    finally:
        return answer

def fun_newtons_method_mod(info, accuracy):
    answer = []
    try:
        eq, a, b = info[0], info[1], info[2]
        x0 = (a + b) / 2
        answer.append(f'Метод Ньютона(модифікований)\n'
                      f'Вибираємо початкове наближення кореня, н-д, x0 =
(a + b) / 2 = {x0} .\n'
                      f'Перевіряємо достатні умови збіжності методу
Ньютона:\n')

        if diff(diff(eq)) != 0:
            answer.append(f' - функція f(x) двічі диференційована на
проміжку [{a}; {b}]:\n')
            answer.append(f'f(x) = {eq}\n'
                          f'f`(x) = {diff(eq)}\n'
                          f'f``(x) = {diff(diff(eq))}\n')
        else:
            answer.append(
                f' - функція f(x) двічі не диференційована на проміжку
[{a}; {b}], тому майбутні обрахунки неможливі:\n')
            answer.append(f'f(x) = {eq}\n'
                          f'f`(x) = {diff(eq)}\n'
                          f'f``(x) = {diff(diff(eq))}\n')
            return answer
```

```
        answer.append(f'Визначаємо знакосталість першої і другої похідних
на відрізку:\n')
        answer.append(f' - перевіримо, чи на даному проміжку 1 корінь
f(x0) * f(x1):\n')

        if F(eq, a) * F(eq, b) < 0:
            answer.append(f'f({a}) * f({b}) = {F(eq, a) * F(eq, b)} < 0,
тому проміжку [{a};{b}] належить тільки один корінь.\n')
        else:
            answer.append(f'f({a}) * f({b}) = {F(eq, a) * F(eq, b)} > 0,
тому проміжку [{a};{b}] належить або декілька коренів або жодного. Оберіть
інший проміжок.\n')
            return answer

        answer.append(f'- перевіримо, чи початкове наближення задовольняє
умову f(x0) * f``(x0) > 0:\n')

        if F(eq, x0) * der2(eq, x0) > 0:
            answer.append(f'f({b}) * f``({b}) > 0:  - умова
задовольняється.\n')
        else:
            answer.append(f'f({b}) * f``({b}) < 0:  - умова не
задовольняється.\n')
            return answer

        x0 = dround(x0, accuracy)
        xn = F(eq, x0)
        xn1 = xn - F(eq, xn) / der1(eq, xn)

        answer.append(f'Розв`язуємо {eq} = 0 на проміжку [{a};{b}]\n'
                      f'x = {dround(x0, accuracy)}, '
                      f'f(x) = {dround(xn, accuracy)}, '
                      f'f(x) - f(x)/f`(x) = {dround(xn1, accuracy)}\n')

        ans, c, t = x0, 1, 1 / (10 ** accuracy)

        while abs(x0 - xn1) >= t and c != 20 and float(dround(xn1,
accuracy)) != 0 and float(dround(xn, accuracy)) != 0:
            c += 1

            x0 = dround(x0 - xn1)
            xn = F(eq, x0)
            xd = der1(eq, x0)
            xn1 = xn / xd
            answer.append(f'x = {dround(x0, accuracy)}, '
                          f'f(x) = {dround(xn, accuracy)}, '
                          f'f(x) - f(x)/f`(x) = {dround(xn1,
accuracy)}\n')
            ans = x0

        answer.append(f'Відповідь: x = {dround(ans, accuracy)}')
    except Exception as e:
        answer.append(f'Не можу виконати обрахунки на даному проміжку.\n')
    finally:
        return answer

def fun_dichotomy_method(info, accuracy):
    answer = []
    try:
        eq, a, b = info[0], info[1], info[2]
        x0 = (a + b) / 2
```

```
        answer.append(f'Метод Дихотомії\n'
                      f'Перевіримо, чи на даному проміжку [{a};{b}] 1
корінь:\n f({a}) * f({b}) = ')

        if F(eq, a) * F(eq, b) < 0:
            answer.append(f'{F(eq, a) * F(eq, b)} < 0, тому проміжку
належить тільки один корінь.\n')
        else:
            answer.append(
                f'{F(eq, a) * F(eq, b)} > 0, тому проміжку належить або
декілька коренів або жодного. Оберіть інший проміжок.\n')
            return answer

        if diff(diff(eq)) != 0:
            answer.append(f' - функція f(x) двічі диференційована на
проміжку [{a}; {b}]:\n')
            answer.append(f'f(x) = {eq}\n'
                          f'f`(x) = {diff(eq)}\n'
                          f'f``(x) = {diff(diff(eq))}\n')
        else:
            answer.append(f' - функція f(x) двічі не диференційована на
проміжку [{a}; {b}], тому майбутні обрахунки неможливі:\n')
            answer.append(f'f(x) = {eq}\n'
                          f'f`(x) = {diff(eq)}\n'
                          f'f``(x) = {diff(diff(eq))}\n')
            return answer

        answer.append(f'- перевіримо, чи початкове наближення задовольняє
умову f(x) * f``(x) > 0:\n')

        if F(eq, x0) * der2(eq, x0) > 0:
            answer.append(f'f({x0}) * f``({x0}) > 0:  - умова
задовольняється.\n')
        else:
            answer.append(f'f({b}) * f``({x0}) < 0:  - умова не
задовольняється.\n')
            return answer

        answer.append(f'Розв`язуємо {eq} = 0 на проміжку [{a};{b}]\n')
        answer.append(f'Початкове наближення кореня, x = (a + b) / 2 =
{x0}\n')
        t = 1 / (10 ** accuracy)
        c = 1

        x0 = (a + b) / 2
        ans = F(eq, x0)

        answer.append(f'Шукаємо f(x): f({x0}) = {dround(ans, accuracy)}\n'
                      f'Значення функції f(x)')

        if not (abs(ans) > t and c != 20):
            answer.append(f' < {round(t, accuracy)}\n, тому шукане
значення x = {x0}')
        else:
            answer.append(f' > нашої точності ({dround(t, accuracy)}),
шукаємо новий проміжок.\n'
                          f'Новий проміжок або [{a};{x0}] або [{x0};{b}].
Щоб його обрати потрібно знайти на кінцях '
                          f'якого проміжка функція набуває значень різних
знаків\n')
```

```
        if F(eq, a) * ans < 0:
            b = x0
            answer.append(f'Так як f({dround(x0,accuracy)}) *
f({dround(a,accuracy)}) < 0, то b = x і новий проміжок
[{dround(a,accuracy)};{dround(b,accuracy)}]\n')
        else:
            a = x0
            answer.append(f'Так як f({dround(x0,accuracy)}) *
f({dround(b,accuracy)}) < 0, то a = x і новий проміжок
[{dround(a,accuracy)};{dround(b,accuracy)}]\n')
        answer.append(f'Далі розрахунки в стислому варіанті\n')

        while abs(ans) > t and c != 20:

            x0 = (a + b) / 2
            ans = F(eq, x0)

            answer.append(f'x = {dround(x0, accuracy)}, '
                          f'f(x) = {dround(ans, accuracy)}')

            if F(eq,a) * ans < 0:
                b = x0
                answer.append(f', f({dround(x0,accuracy)}) *
f({dround(a,accuracy)}) < 0, b = x,
[{dround(a,accuracy)},{dround(b,accuracy)}] \n')
            else:
                a = x0
                answer.append(f', f({dround(x0,accuracy)}) *
f({dround(b,accuracy)}) < 0, a = x,
[{dround(a,accuracy)};{dround(b,accuracy)}] \n')
            c += 1

        answer.append(f'Відповідь: x = {dround(x0, accuracy)}')
    except Exception as e:
        answer.append(f'Не можу виконати обрахунки на даному проміжку.\n')
    finally:
        return answer

def fun_secant_method(info, accuracy):
    answer = []
    try:
        eq, a, b = info[0], info[1], info[2]
        print( eq, a, b)
        answer.append(f'Метод Хорд\n'
                      f'Перевіримо, чи на даному проміжку 1 корінь f({a})
* f({b}):\n')

        if F(eq, a) * F(eq, b) < 0:
            answer.append(f'f({a}) * f({b}) = {F(eq, a) * F(eq, b)} < 0,
тому проміжку [{a};{b}] належить тільки один корінь.\n')
        else:
            answer.append(f'f({a}) * f({b}) = {F(eq, a) * F(eq, b)} > 0,
тому проміжку [{a};{b}] належить або декілька коренів або жодного. Оберіть
інший проміжок.\n')
            return answer

        if diff(diff(eq)) != 0:
            answer.append(f' - функція f(x) двічі диференційована на
проміжку [{a}; {b}]:\n')
            answer.append(f'f(x) = {eq}\n'
                          f'f`(x) = {diff(eq)}\n'
                          f'f``(x) = {diff(diff(eq))}\n')
```

```
        else:
            answer.append(f' - функція f(x) двічі не диференційована на
проміжку [{a}; {b}], тому майбутні обрахунки неможливі:\n')
            answer.append(f'f(x) = {eq}\n'
                          f'f`(x) = {diff(eq)}\n'
                          f'f``(x) = {diff(diff(eq))}\n')
            return answer

        answer.append(f'Розв`язуємо {eq} = 0 на проміжку
[{info[1]};{info[2]}]\n'
                      f'Знайдемо значення функції f(x), де x = a - f(a) *
(b - a) / (f(b) - f(a))\n')


        x = dround(a) - F(eq, a) * dround(b - a) / (F(eq, b) - F(eq, a))
        ans = F(eq,x)
        answer.append(f'x = {a} - f({a}) * ({b} - {a}) / (f({b}) -
(f({a}))) ='
                      f' {a} - ({dround(F(eq, a),accuracy)}) * ({b} -
({a})) / ({dround(F(eq, b),accuracy)} - ({dround(F(eq, a),accuracy)})) = '
                      f'{dround(x,accuracy)}\n'
                      f'f(x) = f({dround(x,accuracy)}) =
|{dround(ans,accuracy)}|')

        t = 1 / (10 ** accuracy)
        if not abs(ans) > t:
            answer.append(f' < {round(t, accuracy)}\n, тому шукане
значення x = {x}')
        else:
            answer.append(f' > нашої точності ({dround(t, accuracy)}),
шукаємо новий проміжок.\n'
                          f'Новий проміжок або [{a};{dround(x,accuracy)}]
або [{dround(x,accuracy)};{b}]. Щоб його обрати потрібно знайти на кінцях
'
                          f'якого проміжка функція набуває значень різних
знаків\n')

        if F(eq, a) * ans < 0:
            b = x
            answer.append(f'Так як f({dround(x,accuracy)}) *
f({dround(a,accuracy)}) < 0, то b = x і новий проміжок
[{dround(a,accuracy)};{dround(b,accuracy)}]\n')
        else:
            a = x
            answer.append(f'Так як f({dround(x,accuracy)}) *
f({dround(b,accuracy)}) < 0, то a = x і новий проміжок
[{dround(a,accuracy)};{dround(b,accuracy)}]\n')
        answer.append(f'Далі розрахунки в стислому варіанті\n')

        c = 1
        while abs(ans) > t and c != 20:
            x = dround(a) - F(eq, a) * (dround(b) - dround(a)) / (F(eq, b)
- F(eq, a))
            ans = F(eq,x)
            answer.append(f'x = {dround(x, accuracy)}, '
                          f'f(x) = {dround(ans, accuracy)}')
            if F(eq,a) * ans < 0:
                b = x
                answer.append(
                    f', f({dround(x, accuracy)}) * f({dround(a,
accuracy)}) < 0, b = x, [{dround(a, accuracy)},{dround(b, accuracy)}] \n')
            elif F(eq,b) * ans < 0:
```

```
                a = x
                answer.append(
                    f', f({dround(x, accuracy)}) * f({dround(b,
accuracy)}) < 0, a = x, [{dround(a, accuracy)},{dround(b, accuracy)}] \n')

            else:
                answer.append(f'Відповідь: x = {dround(ans,accuracy)}')

        answer.append(f'Відповідь: x = {dround(x,accuracy)}')
    except Exception as e:
        answer.append(f'Не можу виконати обрахунки на даному проміжку.\n')
    finally:
        return answer

def graf(text, id_formula):
    eqn = text
    a = 10
    b = -10
    c = abs(a - b) * 100+1

    y_list = []
    x_list = np.linspace(a, b, c)
    for x in range(len(x_list)):
        eqn_list = list(eqn)
        for i in range(0, len(eqn_list)):
            if eqn_list[i] == 'x':
                eqn_list[i] = str(round(x_list[x],3))
        f = sympy.N(''.join(eqn_list))
        try:
            f = float(f)
            if f <= 10:
                y_list.append(float(f))
            else:
                y_list.append(None)

        except:
            y_list.append(None)
    #print(len(x_list),len(y_list))
    xx_list = []
    yy_list = []

    min_y = 10000
    min2_y = 10000
    ind = 1000
    ind2 = 1000
    plt.grid()
    plt.ylim(-10, 10)
    for i in range(len(x_list)):

        y = y_list[i]
        if y is not None:
            #print(x_list[i], y)
            xx_list.append(x_list[i])
            yy_list.append(y)
            if abs(y) < min_y:
                min2_y = min_y
                ind2 = ind
                ind = i
                min_y = abs(y)
        else:
            if len(xx_list) > 1 :
                line = plt.plot(xx_list, yy_list)
```

```
                plt.setp(line, color="blue")
                xx_list, yy_list = [], []
    if len(xx_list) !=0:
        line = plt.plot(xx_list, yy_list)
        plt.setp(line, color="blue")
        xx_list, yy_list = [], []

    #plt.plot(xx_list, yy_list)

    plt.savefig(f'{id_formula}.png')
    plt.close()

    return #x_list[ind]
```

Файл config.py:

```
TOKEN = '1277387830:AAFyeX0opX7P1lEHO5WoG_cBvPYZilBBqJE'
database_name = 'db.db'
table_user = 'user'
table_formula = 'formula'
graf = True
accuracy = 5
details = True
```

Файл Schedule.py:

```
from logging import getLogger
from traceback import format_exc
from datetime import datetime
from schedule import Scheduler

logger = getLogger('schedule')


class SafeScheduler(Scheduler):

    def __init__(self, reschedule_on_failure=True):
        self.reschedule_on_failure = reschedule_on_failure
        super().__init__()

    def _run_job(self, job):
        try:
            super()._run_job(job)
        except Exception:
            logger.error(format_exc())
            job.last_run = datetime.now()
            job._schedule_next_run()
```

Файл database.py:

```
from SQlighter import SQLighter
import config
from storage import action_user, user_setting

def write_new_user(id):
    try:
        db_worker = SQLighter(config.database_name)

db_worker.write_new_user(config.table_user,id,config.graf,config.details,c
onfig.accuracy)
        #print('Користувач успішно записаний')
        db_worker.close()
    except Exception as e:
        print(e)
```

```
def write_new_user_setting(id,name,value,type):
    try:
        db_worker = SQLighter(config.database_name)
        db_worker.write_new_user_setting(config.table_user,id,name,value)
        #print(f'{id} Змінив {name} на {value}')
        db_worker.close()
        if id in action_user.keys():
            #print(action_user)
            del action_user[id]
    except Exception as e:
        print(e)

def read_user_setting(id):
    try:
        if id in user_setting.keys():
            #print('read_user_setting')
            #print(user_setting[id])
            return user_setting[id]
        else:
            db_worker = SQLighter(config.database_name)
            setting = db_worker.read_user_setting(config.table_user,id)
            db_worker.close()
            return setting[0]
    except Exception as e:
        print(e)

def write_new_formula(formula):
    try:
        db_worker = SQLighter(config.database_name)
        db_worker.write_new_formula(config.table_formula,formula)
        db_worker.close()
    except Exception as e:
        print(e)

def write_x(id_formula, x):
    try:
        db_worker = SQLighter(config.database_name)
        db_worker.write_x(config.table_formula,id_formula, x)
        db_worker.close()
    except Exception as e:
        print(e)


def read_formula(formula):
    try:
        db_worker = SQLighter(config.database_name)
        id_formula, x =
db_worker.read_formula(config.table_formula,formula)
        db_worker.close()
        return id_formula, x
    except Exception as e:
        print(e)


def ckeck_user_db(id):
    try:
        db_worker = SQLighter(config.database_name)
        db_worker.check_user(config.table_user,id)
        db_worker.close()
    except Exception as e:
        print(e)
```

```
def read_users(table_name):
    try :
        db_worker = SQLighter(config.database_name)
        text = db_worker.read_user(table_name)
        db_worker.close()

        if not text:
            print('Пусто')
        else:
            for id, graf, details, accuracy in text:
                user_setting[id] = []
                user_setting[id] = [graf, details, accuracy]
            print('Дані з бази успішно отримані')
    except Exception as e:
        print(e)
```

Файл Sqlighter.py:

```
import sqlite3

class SQLighter:

    def __init__(self, database):
        self.connection = sqlite3.connect(database)
        self.cursor = self.connection.cursor()

    def write_new_user(self, table_name, id, graf, details, accuracy):
        with self.connection:
            self.cursor.execute(f"""INSERT INTO {table_name} ('id',
'graf', 'details', 'accuracy')
                                    VALUES ({id}, {graf}, {details},
{accuracy})""")
            self.connection.commit()

    def write_new_user_setting(self, table_name, id, name,value):
        with self.connection:
            self.cursor.execute(f"""UPDATE {table_name} SET {name} =
{value} WHERE id = {id}""")
            self.connection.commit()

    def read_user_setting(self, table_name, id):
        with self.connection:
            return self.connection.execute(f""" SELECT graf, details,
accuracy
                                               FROM {table_name}
                                               WHERE id =
{id}""").fetchall()

    def check_user(self, table_name, id):
        with self.connection:
            return self.connection.execute(f""" SELECT COUNT(id)
                                               FROM {table_name}
                                               WHERE id =
{id}""").fetchall()[0][0]


    def write_new_formula(self, table_name, name):
        with self.connection:
            print(table_name,name)
            self.cursor.execute(f"""INSERT INTO {table_name} ('name')
                                        VALUES ('{name}')""")
            self.connection.commit()
```

```
    def write_x(self, table_name, id_formula, x):
        with self.connection:
            self.cursor.execute(f"""UPDATE {table_name} SET x = '{x}'
WHERE id = {id_formula}""")
            self.connection.commit()

    def read_formula(self, table_name, name):
        with self.connection:
            x = self.connection.execute(f""" SELECT id,x
                                             FROM {table_name}
                                             WHERE name =
'{name}'""").fetchall()
            if not x:
                return None, None
            else:
                return x[0][0], x[0][1]



    def read_users(self, table_name):
        with self.connection:
            return self.connection.execute(f"SELECT id, graf, details,
accuracy FROM '{table_name}'").fetchall()



    def write_user(self, table_name, id, graf, details, accuracy):
        with self.connection:
            self.cursor.execute(f"""INSERT INTO {table_name} ('id',
'graf', 'details', 'accuracy')
                                    VALUES ({id}, {graf}, {details},
{accuracy})""")
            self.connection.commit()

    def read_user(self, table_name):
        with self.connection:
            return self.connection.execute(f"SELECT id, graf, details,
accuracy FROM '{table_name}'").fetchall()

    def close(self):
        self.connection.close()
```

Файл starage.py:

```
action_user = {}
user setting = {}
user_stage = {}
```