ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка телегра чат-бота на мові Python з використанням баз даних SQL»

на здобуття освітнього ступеня бакалавра
зі спеціальності 122 Комп'ютерні науки
(код, найменування спеціальності)
освітньо-професійної програми Комп'ютерні науки
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Станіслав КОНИК
(підпис) *(Ім'я, ПРІЗВИЩЕ здобувача)*

Виконав:
здобувач вищої освіти Станіслав КОНИК
група КНД-41

Керівник: Олена ШИКУЛА
науковий ступінь, вчене звання д.ф.-м.н., професор

Рецензент:
науковий ступінь, вчене звання
_____
(Ім'я, ПРІЗВИЩЕ)

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Комп'ютерних наук

Ступінь вищої освіти Бакалавр

Спеціальність Комп'ютерні науки

Освітньо-професійна програма Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедрою Комп'ютерних наук

_____ Віктор ВИШНІВСЬКИЙ
«_____» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Конику Станіславу Павловичу
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Розробка телеграм чат-бота на мові Python з використанням баз даних SQL

керівник кваліфікаційної роботи Олена ШИКУЛА д.ф.-м.н., професор,
(Ім'я, ПРІЗВИЩЕ науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» 02.2024р. №36

2. Строк подання кваліфікаційної роботи «31» травня 2024р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література, середовище Python, бази даних SQLite, середовище розроби PyCharm.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
Огляд та дослідження існуючих рішень у сфері розробки чат-ботів на мові Python.
Інтеграція бази даних SQL
Аналіз вимог до функціональності чат-бота з урахуванням потреб користувача та можливості технологічної реалізації.

Тестування та налагодження роботи чат-бота з різними сценаріями взаємодій.

5. Перелік графічного матеріалу: *презентація*

1) Тема дипломної роботи
2) Мета роботи. Предмет дослідження.
3) Актуальність роботи.
4) Аналіз розробки телеграм чат-ботів.
5) Інтеграція бази даних в телеграм чат-бота.

6. Дата видачі завдання 23.02.2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	23.02-09.03.24	виконане
2	Дослідження поставленої задачі	09.03-16.03.24	виконане
3	Опис теоритичних відомостей розробки телеграм чат-бота	17.03- 23.03.24	виконане
4	Аналіз середовища програмування, вибір платформи для хостингу бота	24.03- 29.03.24	виконане
5	Розробка телеграм чат-бота	29.03-10.04.24	виконане
6	Інтеграція бази даних SQLite	10.04-20.04.24	виконане
7	Оформлення роботи : вступ, висновки, реферат	20.04-02.05.24	виконане
8	Доповідь та демонстраційні матеріали	02.05-11.05.24	виконане

Здобувач вищої освіти _____ Станіслав КОНИК
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Олена ШИКУЛА
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 60 стор., 46 рис., 1 табл., 21 джерело

Наукове завдання – дослідження та розробка методів і технологій створення телеграм-чат бота на мові Python з використанням баз даних SQL з метою оптимізації процесів взаємодії з користувачами, підвищення його ефективності та надійності.

Мета роботи – створення функціонального чат-бота, який здатен взаємодіяти з користувачами у реальному часі через месенджер Telegram.

Об'єкт дослідження – телеграмм чат-бот.

Предмет дослідження – процес розробки та функціонування телеграм чат-бота на мові Python з використанням баз даних SQL.

Короткий зміст роботи:

Робота спрямована на розробку та дослідження телеграм чат-бота на мові програмування Python з використанням баз даних SQL.

У роботі вивчаються і аналізуються існуючи методи та технології розробки чат-ботів, створюється програмне забезпечення для функціонування чат-бота та його взаємодії з користувачами через месенджер Telegram.

Основний акцент робиться на інтеграції з базою даних SQL для збереження та обробки інформації. Результати дослідження використовуються для вдосконалення та оптимізації функціональності чат-бота з метою поліпшення його ефективності та зручності використання.

КЛЮЧОВІ СЛОВА: ТЕЛЕГРАМ ЧАТ-БОТ, СЕРЕДОВИЩЕ ПРОГРАМУВАННЯ PYTHON, БАЗА ДАНИХ SQL

ABSTRACT

Text part of the bachelor's qualification work:60 pages, 46 fig., 1 tab., 21 sources

The purpose of the work is to create a functional chatbot capable of interacting with users in real time through the Telegram messenger.

Object of research – Telegram chat bot.

Subject of research –process of development and operation of telegram chatbot in Python using SQL database.

Summary of the work: The work is aimed at the development and research of chatbot telegrams in the Python programming language using SQL databases.

The work studies and analyzes existing methods and technologies for the development of chatbots, creates software for the operation of chatbot and it’s interaction with users through the Telegram messenger.

The main emphasis is on integration with an SQL database for storing and processing information. The results of the research and used to improve and optimize the functionality of the chatbot on order to improve it’s efficiency and easy of use.

KEYWORDS: INFORMATION SYSTEM, CERAMIC TILE STORE, RAD STUDIO, MICROSOFT ACCESS DATABASE

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

БД – база даних

ОС – операційна система

API – набір визначень підпрограм, протоколів взаємодії та засобів для створення програмного забезпечення

Aiogram - фреймворк для створення телеграм-ботів у Python

PATH - це змінна середовища в операційних системах, яка містить список каталогів, у яких система шукає виконувані файли програм.

SOAP (Simple Object Access Protocol) - це протокол обміну повідомленнями, що використовує XML для комунікації між різними програмами у розподілених системах.

SDK - набір інструментів для розробки програмного забезпечення.

ВСТУП

Чат-боти є комп'ютерними програмами, розробленими для взаємодії з користувачами шляхом обміну інформацією, що базується на попередньо визначених алгоритмах та/або використанні елементів штучного інтелекту.

Швидкий розвиток та поширення чат-ботів обумовлені загальною тенденцією втрати популярності звичайних додатків, оскільки чат-боти можуть виконувати аналогічні функції. Крім того, зростання популярності різноманітних месенджерів також сприяє поширенню чат-ботів. Таким чином, замість завантаження окремого додатка на пристрій користувача, він може скористатися послугою, що працює всередині звичайного месенджеру. Цей підхід забезпечує зручність та легкість використання для користувача, в той же час раціоналізуючи простір на пристрої та оптимізуючи використання ресурсів.

Невід’ємною частиною цифрової екосистеми стали чат-боти, які можуть оперативно та автоматично відповідати на запитання користувачів у різних мессенджерах, таких як Facebook Messenger, Telegram, Viber та інші. Чат боти виконують широкий спектр функцій, починаючи від надання інформації та закінчуючи вирішенням повсякденних завдань. Їх використання користувачами є дуже зручним і ефективним способом отримання потрібної інформації та виконання різних завдань. З іншого боку, для компанії використання чат-ботів в мессенджерах стає важливим інструментом для взаємодії з клієнтами, покращення обслуговування та підвищення впізнаваності бренду. Telegram є відомим мессенджером, який гарантує конфіденційність і захищеність даних завдяки використанню закритого коду та сучасних алгоритмів шифрування, таких як RSA-2048 та AES.

Розробка та впровадження телеграм чат-бота є актуальною ніж будь-коли. Вони знаходять своє застосування в різних сферах, починаючи від віртуальної підтримки клієнтів і закінчуючи автоматизацією бізнес процесів. Тож основною метою

данної дипломної роботи полягає в тому, щоб розробити та реалізувати телеграм чат-бота, використовуючи при цьому зручну мову програмування Python разом з інтеграцією бази даних SQL.

Проект далеко не обмежується лише однією розробкою програмного забезпечення. Він передбачає глибокий аналіз вимог, ретельне проектування архітектури системи, послідовне тестування та оптимізацію роботи чат-бота.

Python відомий своєю простотою, ефективністю та безмежною гнучкістю, що робить його ідеальним вибором для реалізації різноманітних проектів, включаючи розробку Telegram чат-ботів. Використання SQL, у свою чергу, дозволить забезпечити надійне збереження та ефективний доступ до інформації, необхідної для функціонування та взаємодії з користувачем.

Мета дипломної роботи полягає в глибокому аналізі, розробці та реалізації проєкту – телеграм-чат-бота, спрямованого на систематичне відстеження та облік фінансових операцій. Цей бот розробляється з метою забезпечення користувачів зручним та ефективним інструментом для контролю їх фінансового стану. Він буде здатен автоматично відслідковувати доходи та витрати, фіксувати їх історію, а також надавати корисні фінансові поради на основі зібраної інформації.

Для досягнення поставленої мети необхідно виконати ряд завдань:

1. Дослідження теоретичних основ чат-боту Telegram: Провести аналіз принципів функціонування чат-ботів у Telegram, вивчити їхню архітектуру та можливості для подальшого використання у проекті.
2. Визначення всіх переваг та недоліків боту: Проаналізувати переваги та недоліки чат-ботів у Telegram з точки зору їхньої ефективності, зручності використання, можливостей і обмежень.
3. Створення технічного завдання для розробки системи: Розробити детальне технічне завдання, яке містить вимоги до функціоналу чат-бота, його інтерфейсу, методи взаємодії з користувачем, а також вимоги до безпеки та захисту даних.
4. Дослідження мови програмування для створення бота: Вивчити різні мови програмування, їхні особливості та придатність для реалізації

5. поставленого завдання. Обрати найбільш підходящу мову програмування для розробки чат-бота, враховуючи його потреби та особливості.

6. Інтегрувати базу даних для створення унікального користувача в чат-боті.

1 ТЕОРЕТИЧНІ ВІДОМОСТІ ПРЕДМЕТНОЇ ОБЛАСТІ. АКТУАЛЬНІСТЬ ЧАТ-БОТІВ ТА МОЖЛИВОСТІ ЇХ РЕАЛІЗАЦІЇ

1.1 Мета проекту

В сучасному світі, коли швидкість розвитку інформаційних технологій постійно зростає, дослідження актуальності та можливостей чат-ботів стає надзвичайно важливим завданням. Ці інтелектуальні програмні агенти, які спроектовані для взаємодії з людьми через текстові або голосові команди, відіграють все більш вирішальну роль у спілкуванні та наданні послуг.

Початковою метою цього дослідження є аналіз ролі чат-ботів у сучасному світі технологій. Розуміння того, як чат-боти впливають на спосіб спілкування та вирішують різні завдання, дозволить краще розуміти їх значення та перспективи розвитку.

Далі, ми спрямовуємося на вивчення потреб користувачів у сфері використання чат-ботів. Зрозуміння цих потреб дозволить нам розробити чат-ботів, які ефективно відповідають на запити та вимоги різних категорій користувачів. Враховуючи широкий спектр сфер використання чат-ботів, від клієнтського обслуговування до освіти та розваг, важливо з'ясувати, які саме завдання вони можуть виконувати найефективніше.

Окрім того, ми докладемо зусиль для вивчення технічних аспектів створення чат-ботів. Це включає в себе вивчення різних платформ, мов програмування, алгоритмів машинного навчання та природньої мови, а також розуміння їх переваг та обмежень у конкретних сценаріях використання.

Загалом, цей проект спрямований на розширене дослідження ролі, потреб та технічних аспектів чат-ботів, що допоможе нам краще розуміти їхнє місце у сучасному цифровому світі та потенціал їх використання в різних сферах життя.

1.2 Актуальність чат-ботів

Актуальність чат-ботів визначається їхнім потенціалом у вирішенні різноманітних завдань та забезпеченні зручного спілкування з користувачами. З розвитком штучного інтелекту, машинного навчання та обробки природної мови, чат-боти стають все більш інтелектуальними та гнучкими у взаємодії з людьми (рис 1.1).

Однією з ключових областей актуальності чат-ботів є клієнтське обслуговування. Вони дозволяють компаніям автоматизувати підтримку користувачів, відповідати на питання та вирішувати проблеми без прив'язки до годин роботи. Це допомагає підвищити задоволеність клієнтів і знизити витрати на підтримку [1].

У сфері електронної комерції чат-боти можуть допомогти в забезпеченні персоналізованої консультації та підтримки під час покупок, що підвищує конверсію та забезпечує задоволеність клієнтів.

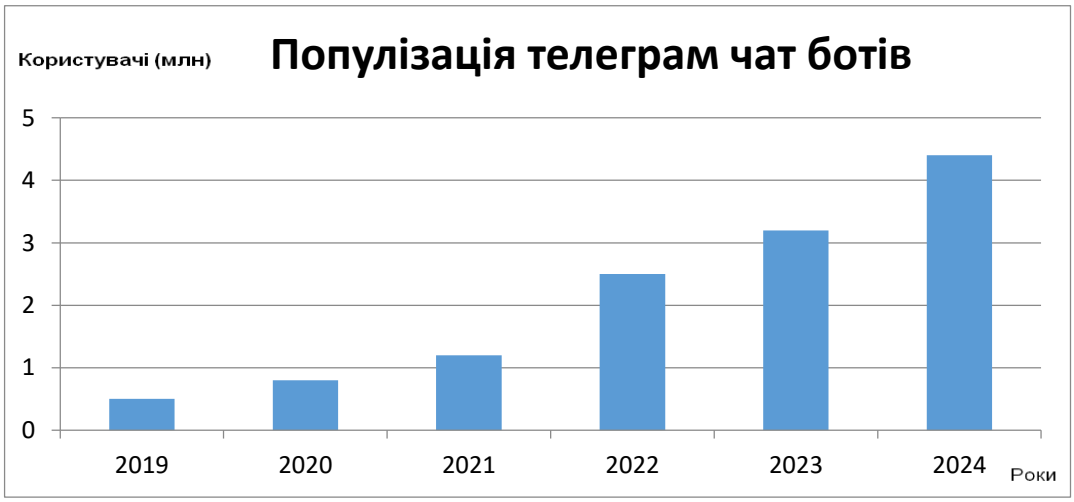


Рисунок 1.1 – Діаграма популізації телеграм чат ботів з 2019 по 2024 роки

Також, чат-боти можуть бути використані в освітній сфері для надання навчальних матеріалів, проведення тестувань та індивідуалізованої підтримки учнів.

Загалом, актуальність чат-ботів полягає у їхньому потенціалі в різних сферах, починаючи від підтримки користувачів та закінчуючи освітніми та розважальними застосуваннями. Ці інтелектуальні програмні агенти відкривають нові можливості для автоматизації та зручного спілкування, що робить їх невід'ємною складовою сучасного цифрового світу [20].

1.3 Порівняльний аналіз платформ для інтеграції чат-бота

При виборі платформи для інтеграції чат-бота на мові Python з використанням баз даних SQL необхідно врахувати ряд факторів, що впливають на продуктивність, зручність розробки та майбутню можливість розширення функціональності. Основні критерії порівняння включають легкість використання, наявність документації та спільноти користувачів, підтримку мови програмування Python, інтеграцію з базами даних SQL та можливості розширення.

Перш за все, розглянемо легкість використання. Цей аспект особливо важливий для швидкого розвитку та розгортання чат-бота. Деякі платформи, такі як Dialogflow від Google або Microsoft Bot Framework, можуть мати інтуїтивно зрозумілі інтерфейси, що полегшують створення та налаштування бота, але можуть бути обмежені у можливостях кастомізації. З іншого боку, розвиток власного чат-бота з використанням бібліотеки Python, такої як Aiogram або Telebot, може забезпечити більшу гнучкість, але вимагатиме більшого рівня технічної експертизи (рис.1.2).

Рисунок 1.2 – Популярні месенджери в Україні на квітень 2023 [2]

Другим критерієм є наявність документації та активної спільноти користувачів. Це важливо для швидкого вирішення проблем та підтримки. Популярні платформи, як правило, мають велику кількість документації та активну спільноту користувачів, що сприяє швидкому розвитку та вирішенню проблем.

Підтримка мови програмування Python та інтеграція з базами даних SQL є ключовими для нашого проекту. Ідеальна платформа має надати зручні інструменти для роботи з Python та можливість легко взаємодіяти з базою даних SQL для зберігання та обробки інформації.

Нарешті, важливо врахувати можливості розширення. Платформа повинна мати потенціал для майбутнього розвитку та додавання нового функціоналу без значних зусиль.

З урахуванням усіх цих критеріїв, необхідно ретельно проаналізувати кожну платформу та обрати ту, яка найкращим чином відповідає потребам нашого проекту з розробки телеграм чат-бота на мові Python з використанням баз даних SQL. Враховуючи усі ці критерії, ми приходимо до висновку, що найкращим варіантом для нашого проекту з розробки телеграм чат-бота на мові Python з використанням баз даних SQL є використання середовища Telegram.

1.4 Функціонал Telegram чат-бота

- Приймання повідомлень від користувачів:

Бот здатний приймати повідомлення від користувачів у режимі реального часу. Це включає в себе текстові повідомлення, зображення, аудіо та відео.

- Відправлення повідомлень:

Бот має можливість відправляти різні типи повідомлень користувачам, включаючи текст, зображення, аудіо та відео. Це може бути відповіддю на запит користувача або сповіщення про певні події.

- Використання клавіатур і кнопок:

Для полегшення взаємодії з користувачами бот може використовувати клавіатури і кнопки. Це дозволить користувачам здійснювати швидкий доступ до певних функцій або команд, а також забезпечить зручну навігацію.

- Обробка команд користувачів:

Бот повинен мати можливість розпізнавати та обробляти команди, введені користувачем. Наприклад, це може бути команда на відображення інформації про певний товар або послугу, або запит на виконання певної дії.

- Інтеграція з базою даних SQL:

Для зберігання та обробки інформації бот може використовувати базу даних SQL. Це дозволить зберігати дані користувачів, історію чатів, інформацію про продукти чи послуги, а також будь-яку іншу необхідну інформацію.

- Функції штучного інтелекту:

Бот може використовувати штучний інтелект для покращення якості обробки запитів користувачів. Наприклад, це може бути система розпізнавання мови, генерація відповідей або навіть система рекомендацій на основі попередніх взаємодій.

- Відправлення сповіщень:

Бот може надсилати сповіщення користувачам згідно з певними подіями або регулярними оновленнями. Наприклад, це може бути сповіщення про нові товари чи послуги, акції та знижки або повідомлення про планові технічні роботи.

- Захист інформації користувачів:

Надзвичайно важливою є захист інформації користувачів. Бот повинен використовувати шифрування та інші методи безпеки даних для забезпечення конфіденційності та цілісності інформації користувачів [3].

2 ВИБІР ПРОГРАМНИХ СЕРЕДОВИЩ. ПЕРЕВАГИ SQL. ПРОЕКТУВАННЯ ЧАТ-БОТА

Після вибору платформи для створення чат-бота та розуміння її ролі у сучасному житті, наступним важливим кроком є визначення програмних засобів для реалізації ідеї. Це включає в себе вибір мови програмування, фреймворків, бібліотек та інших інструментів, які допоможуть втілити концепцію чат-бота в життя.

Важливо врахувати потреби проекту, а також технічні вимоги та можливості обраної платформи. Наприклад, якщо для розробки обрано мову Python, можливо, буде доцільно скористатися бібліотеками як Aiogram або Python-telegram-bot. Якщо ж обрано JavaScript, то можна використовувати фреймворки типу Node.js або бібліотеку Telegraf для розробки бота для Telegram.

Крім того, варто врахувати інтеграцію з базою даних, яка забезпечить зберігання та обробку інформації, та можливості розширення функціоналу з моментом подальшого розвитку проекту. Вибір правильних програмних засобів є ключовим для успішної реалізації ідеї чат-бота та його подальшого функціонування у сучасному цифровому середовищі.

2.1 Мова програмування

Написання телеграм чат-бота можна виконати як з використанням Python, так і з використанням JavaScript. Давайте порівняємо обидві мови з точки зору їхньої придатності для цієї задачі:

1. Python:

- Мова програмування: Python – це високорівнева мова програмування з простим синтаксисом, що дозволяє швидко створювати програми. Вона має велику

кількість бібліотек та фреймворків, що робить її популярним вибором для розробки чат-ботів.

- Фреймворки: У Python є кілька популярних фреймворків для створення телеграм-чат-ботів, таких як Telebot, Python-telegram-bot та Aiogram.

- Бібліотеки: Python має багато бібліотек для роботи з базами даних, зокрема з SQL. Наприклад, SQLAlchemy – це потужний інструмент для роботи з реляційними базами даних у Python.

- Інтеграція з SQL: Python має декілька бібліотек для роботи з SQL-базами даних, що дозволяють легко інтегрувати їх у ваший телеграм-чат-бот. [4]

2. JavaScript:

- Мова програмування: JavaScript – це скриптова мова програмування, яка широко використовується для розробки веб-застосунків. З використанням Node.js, JavaScript може бути використана і для створення серверних застосунків, включаючи телеграм-чат-ботів.

- Фреймворки: Для JavaScript існують фреймворки, такі як Telegraf або Botpress, які дозволяють легко створювати телеграм-чат-ботів.

- Бібліотеки: JavaScript також має багато бібліотек для роботи з базами даних. Наприклад, для роботи з SQL-базами даних можна використовувати бібліотеки, такі як Sequelize або knex.js.

- Інтеграція з SQL: Інтеграція з SQL базами даних також можлива в JavaScript з використанням відповідних бібліотек. [15]

На основі порівняння Python та JavaScript для написання телеграм-чат-бота можна зробити висновок, що Python має деякі переваги:

1. Python відомий своєю простотою синтаксису, що робить його дуже доступним для початківців та швидкої розробки програм. Це особливо важливо для створення чат-ботів, де важливо швидко реагувати на повідомлення та реалізовувати новий функціонал (рис 2.1).

PYTHON

```python
from telegram.ext import Updater, CommandHandler, MessageHandler, Filters

# Функція, яка обробляє команду /start
def start(update, context):
    update.message.reply_text('Привіт! Я чат-бот. Як я можу допомогти вам?')

# Функція, яка відповідає на повідомлення користувача
def echo(update, context):
    update.message.reply_text(update.message.text)
```

JavaScript

```javascript
const TelegramBot = require('node-telegram-bot-api');

// Створення бота з токеном
const bot = new TelegramBot('YOUR_TOKEN', { polling: true });

// Обробник команди /start
bot.onText(/\/start/, (msg) => {
  const chatId = msg.chat.id;
  bot.sendMessage(chatId, 'Привіт! Я чат-бот. Як я можу допомогти вам?');
});

// Обробник повідомлень від користувача
bot.on('message', (msg) => {
  const chatId = msg.chat.id;
  bot.sendMessage(chatId, msg.text);
});
```

Рисунок 2.1 – Порівняння синтаксису Python та JavaScript

2. Багато бібліотек та фреймворків: Python має великий екосистему бібліотек та фреймворків, спеціально створених для розробки телеграм-чат-ботів. Це робить процес розробки швидшим і простішим.

3. Широкі можливості інтеграції з SQL: Python має потужні бібліотеки для роботи з SQL, що дозволяє легко інтегрувати бази даних у телеграм-чат-боти.

Отже, на підставі цих переваг Python може бути вибраний в якості переважної мови програмування для написання телеграм-чат-ботів, особливо для початківців або тих, хто шукає швидкий та ефективний спосіб реалізації такого проєкту .

2.2 Середовище розробки

Середовище розробки (IDE - Integrated Development Environment) – це програмне забезпечення, яке надає інтегрований набір інструментів та функцій для розробки програмного забезпечення. Ці інструменти зазвичай включають текстовий редактор, компілятор або інтерпретатор, дебагер, засоби автоматичної перевірки коду, сховище для керування версіями (наприклад, Git), інструменти для рефакторингу коду, а також підтримку різних мов програмування.

Середовища розробки спрощують процес розробки програм шляхом надання зручного та інтегрованого робочого середовища, яке допомагає розробникам писати, тестувати, налагоджувати та вдосконалювати програми. Вони забезпечують різні інструменти та підтримують кращі практики розробки програмного забезпечення, що сприяє підвищенню продуктивності та якості коду.

При виборі середовища розробки важливо враховувати його спрямованість на конкретну мову програмування. Чим більше можливостей надає інтегроване середовище розробки, тим більше можливостей для реалізації потенціалу чат-бота. Усе це відкриває широкі можливості для творчості та інновацій, дозволяючи програмістам ефективно втілювати свої ідеї у програмний код. Наступний аналіз деяких середовищ допоможе з'ясувати це краще.

PyCharm – це інтегроване середовище розробки (IDE) для мови програмування Python, яке розробляється компанією JetBrains. Воно відоме своєю потужністю та багатофункціональністю, яка робить його одним з найпопулярніших інструментів для розробки на Python. PyCharm має потужний редактор коду з розумним автодоповненням, підсвічуванням синтаксису, перетягуванням та впровадженням рефакторингу. Середовище надає зручні інструменти для написання та виконання тестів, що допомагає забезпечити якість коду. PyCharm підтримує різні версії мови програмування Python, що дозволяє працювати з різними проектами. Узагальнюючи, PyCharm – це потужне та універсальне

середовище розробки для Python, яке допомагає розробникам створювати якісний та ефективний код.

Visual Studio – це інтегроване середовище розробки (IDE), розроблене компанією Microsoft, яке підтримує різні мови програмування, включаючи C#, C++, Visual Basic, F#, Python та інші. Основна версія, яка використовується для розробки на Python, називається Visual Studio Community, і вона є безкоштовною для особистих, некомерційних та відкритих проектів. Має потужний редактор коду з функціями автодоповнення, підсвічування синтаксису, перетягування та впровадження рефакторингу. Інтегрована підтримка віртуальних середовищ допомагає ізолювати проекти та їх залежності. Visual Studio має інтеграцію з іншими інструментами та сервісами Microsoft, такими як Azure, що дозволяє легко розгортати та керувати хмарними додатками.

Sublime Text – текстовий редактор, який використовується для написання коду та роботи з текстовими документами. Він має дуже широкий спектр можливостей та інструментів, які роблять процес написання коду більш ефективним і зручним. Дозволяє відкривати кілька файлів одночасно в різних панелях, що полегшує порівняння та редагування. Підтримує різноманітні мови програмування та формати файлів, що дозволяє виділяти синтаксис для зручності читання та редагування коду. Редактор забезпечує швидкий доступ до різних функцій за допомогою клавішних комбінацій та командного рядка, що дозволяє суттєво підвищити продуктивність роботи [5].

2.3 Переваги SQL

SQL (Structured Query Language) – це стандартизована мова запитів, призначена для взаємодії з базами даних. Вона використовується для створення, модифікації та керування базами даних, де дані зберігаються у вигляді таблиць. SQL дозволяє виконувати різноманітні операції з даними, такі як вибірка (вилучення), вставка, оновлення та видалення даних, а також управління

структурою бази даних (створення таблиць, індексів, виделених обмежень тощо) [6].

Переваги SQL в контексті розробки телеграм чат-бота на мові Python є невід'ємною складовою успішної і ефективної реалізації проекту. Ось деякі з найзначущих переваг використання SQL:

1. SQL дозволяє зберігати дані в структурованому вигляді, що робить їх легко доступними і зрозумілими для подальшого використання.

2. Запити SQL надають можливість швидкого і ефективного пошуку, фільтрації та сортування даних за різними критеріями, що дозволяє швидко отримувати необхідну інформацію.

3. Бази даних SQL можуть легко масштабуватися для роботи з великим обсягом даних і великою кількістю користувачів, що робить їх ідеальним вибором для проектів з великою кількістю користувачів, таких як чат-боти.

4. SQL забезпечує підтримку транзакцій, що гарантує цілісність та консистентність даних в базі даних, навіть у випадку великої кількості одночасних операцій.

5. SQL бази даних забезпечують високий рівень безпеки, включаючи можливість налаштування прав доступу до даних і застосування різних методів шифрування для захисту конфіденційної інформації.

6. SQL має велику кількість вбудованих функцій та можливостей, які спрощують обробку даних, виконання різних агрегатних операцій та аналітику.

7. SQL бази даних можуть легко інтегруватися з різними мовами програмування, включаючи Python, що спрощує розробку і підтримку проекту.

Використання SQL у розробці телеграм чат-бота на мові Python дозволяє створити потужну та надійну систему обробки та управління даними, що відповідає вимогам сучасних інформаційних технологій.

2.4 Створення бота у Telegram

Перш ніж приступити до написання коду для створення бота, необхідно зареєструвати його в Telegram. Цей процес здійснюється за допомогою вбудованого бота з назвою «BotFather». Щоб розпочати, відкрийте додаток Telegram і скористайтеся пошуковим полем, щоб знайти @BotFather. Після знаходження його, натисніть на нього та відкрийте чат.

Потім введіть команду /newbot. Бот запросить вас ввести ім'я майбутнього бота. Оберіть унікальне ім'я, яке ще не використовується іншими користувачами. Це ім'я буде ім'ям вашого бота у Telegram.

Після введення імені BotFather надасть вам токен доступу. Цей токен потрібно буде використовувати для з'єднання вашого бота з Telegram Bot API, тобто це основний ідентифікатор вашого бота для зв'язку з Telegram.

Крім цього, ви також можете затвердити власний тег для доступу до вашого бота у додатку. Наприклад, якщо ваш бот буде пов'язаний з конкретною темою або послугою, ви можете обрати тег, який буде відображати його при використанні у додатку Telegram. Тег може допомогти іншим користувачам знаходити вашого бота, коли вони шукають ботів за певною тематикою [14].

Для початку роботи з ботом потрібно ознайомитись з функціями та командами які надає «BotFather».

Команди для управління ботом (таб. 2.1) – це набір інструментів, що надаються у вбудованому функціоналі Telegram. Ці інструменти дозволяють взаємодіяти з ботом у різних аспектах, таких як налаштування, видалення, зміна даних при втраті доступу тощо. Таким чином, Telegram гарантує зручність та ефективність для розміщення проектів.

Таблиця 2.1 – Функції для управління ботом

Функція	Характеристика
/start	Початок роботи з BotFather
/newbot	Команда для створення бота
/setname	Зміна назви бота
/setdescription	Встановлення опис
/setcommands	Налаштування списку команд бота
/setinline	Налаштування in-line режиму
/setinlinefeedback	Налаштування повідомлень з варіантами відповіді користувачу
/setjoingroups	Приймання повідомлень від групових чатів
/setprivacy	Налаштування правил доступу до бота
/revokebot	Відключення бота від Telegram
/help	Команда для отримання допомоги

2.5 Серверна платформа для доступу чат-бота

При розгляді процесу розгортання та функціонування телеграм-чат-бота виникає ключове питання вибору підходящої серверної платформи. Вибір оптимальної платформи визначається кількома важливими аспектами, які варто розглянути.

По-перше, слід врахувати підтримку мов програмування, оскільки вона визначає можливості реалізації бота. Наприклад, важливо переконатися, що обрана платформа підтримує мову програмування, яку ми плануємо використовувати для розробки бота, таку як Python або Node.js.

Другим важливим критерієм є легкість використання обраної платформи. Це особливо актуально для початківців у програмуванні, тому інтуїтивно зрозумілий і простий у використанні інтерфейс стає ключовим фактором при виборі.

Третім аспектом є наявність безкоштовного тарифного плану. Для економії коштів та можливості випробування платформи перед придбанням платного плану важливо мати можливість скористатися безкоштовним варіантом або пробним періодом.

Крім того, масштабованість та продуктивність обраної платформи мають вирішальне значення, оскільки вони визначають здатність бота працювати при великому навантаженні та зростанні користувацької бази.

Не менш важливим аспектом є підтримка веб-серверів та API. Обрана платформа повинна безперешкодно підтримувати веб-сервери та з'єднання з Telegram Bot API без обмежень, що забезпечить надійну роботу бота. Слід звернути увагу на спільноту та рівень підтримки платформи. Активна спільнота користувачів та наявність документації та підтримки допоможуть вирішити будь-які проблеми або запитання, що можуть виникнути під час роботи з платформою.

Обираючи серверну платформу для розгортання телеграм-чат-бота, я обрав PythonAnywhere з кількох ключових причин:

По-перше, PythonAnywhere підтримує Python, мову програмування, яку я обрав для створення мого телеграм-чат-бота. Це дозволило мені легко і без проблем розгорнути та запустити свій проект на цій платформі.

По-друге, PythonAnywhere має простий у використанні інтерфейс, що робить процес розгортання та налаштування бота надзвичайно зручним. Це особливо важливо для мене як для початківця в програмуванні.

Крім того, PythonAnywhere пропонує безкоштовний тарифний план для початківців, що дозволяє мені економити кошти та випробовувати платформу перед тим, як перейти на платний тариф. Це дає мені можливість ефективно вивчити можливості платформи та розвинути мій проект без необхідності великих витрат [16].

Загалом, обираючи PythonAnywhere для розгортання свого телеграм-чат-бота, я впевнений у його надійності, продуктивності та зручності використання, що дозволить мені зосередитися на розвитку мого проекту та забезпечить його успішне функціонування.

2.6 Бібліотека Aiogram, переваги та функціонал

Aiogram – це бібліотека програмного забезпечення для мови програмування Python, яка надає інтерфейс для створення чат-ботів для популярної платформи миттєвих повідомлень Telegram. Завдяки бібліотеки розробники можуть легко створювати, налаштовувати та управляти чат-ботами, які можуть відповідати на повідомлення користувачів, відправляти повідомлення, обробляти команди, працювати з клавіатурою і багато іншого. Aiogram використовує офіційне API Telegram для взаємодії з платформою, що забезпечує стабільну та надійну роботу ботів [7].

Щодо функціоналу, Aiogram надає засоби для обробки різних типів повідомлень, відправлення різноманітного контенту користувачам, реагування на спеціальні команди, створення клавіатур для взаємодії з користувачами, логування подій та підтримку різних мов програмування. Цей функціонал дозволяє розробникам створювати потужні та інтерактивні чат-боти для платформи Telegram з мінімальними зусиллями.

Серед переваг, які Aiogram має порівняно з іншими бібліотеками для створення чат-ботів для Telegram, можна виділити кілька ключових аспектів: простоту використання, широкий функціонал, активну підтримку та спільноту розробників, підтримку асинхронності та надійність та стабільність.

2.7 Розробка моделі бота з використанням сучасних технік проектування та аналізу

Розробка моделі бота з використанням сучасних технік проектування та аналізу передбачає систематичне використання передових методів розробки програмного забезпечення, включаючи машинне навчання, природну мову обробки, а також алгоритмічні стратегії для підтримки інтерактивного та ефективного взаємодії з користувачем. Цей підхід дозволяє створити бота, який не лише відповідає потребам сучасного ринку, а й максимально точно відтворює поведінку та реагує на запити користувачів, забезпечуючи високу якість обслуговування.

Для досягнення мети розробки моделі бота застосовуються передові методи аналізу даних та обробки інформації, такі як глибоке навчання, алгоритми класифікації та кластеризації, що дозволяють моделі бота розпізнавати патерни в поведінці користувачів і надавати відповіді з високою точністю. Крім того, використання аналітики даних дозволяє постійно вдосконалювати функціональні можливості бота, адаптуючи його до змінних потреб користувачів і вимог ринку. На рисунку 2.2 зображено структуру контекстної діаграми процесу створення бота «ITHelper» для платформи Telegram.

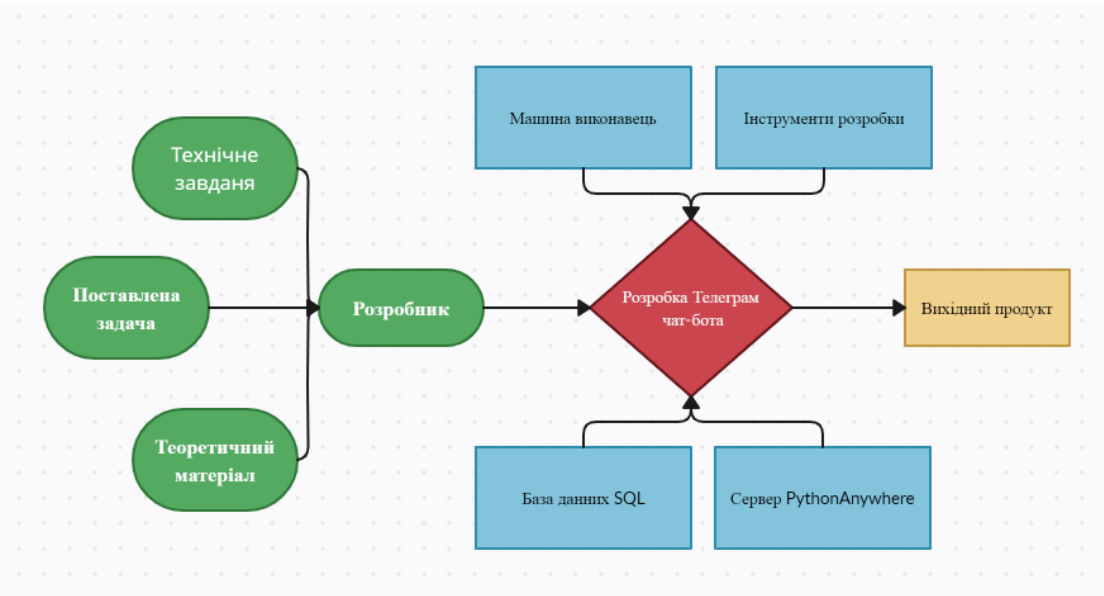


Рисунок 2.2 – Діаграмма розробки чат-бота

Графічне зображення варіантів використання розкриває роль «Користувача до Адміністратора» у системі та функції, які вони можуть використовувати. Це надає можливість встановити взаємозв'язок між можливостями інтерфейсу бота та його користувачами, що є важливим етапом у процесі розробки. Аналіз цієї діаграми допомагає зрозуміти, як краще організувати навігацію та встановити необхідні обмеження. Зображення цього графіка подане на рисунку 2.3.

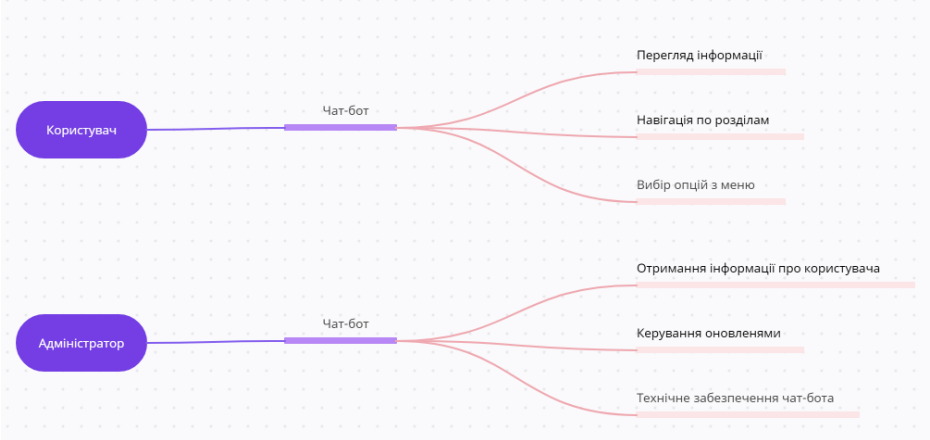


Рисунок 2.3 – Діаграмма взаємодій користувача та адміністратора до чат-бота

У системі існують два види користувачів, кожен з яких має свої унікальні привілеї та доступ до функціоналу. Перший тип - це звичайний користувач, який має обмежений набір можливостей у системі. Зазвичай це доступ до основних функцій, які дозволяють взаємодіяти з системою, здійснювати запити, отримувати інформацію та виконувати базові дії.

Другий тип користувача – адміністратор. Адміністратор, крім того, що має доступ до всіх можливостей звичайного користувача, також має додаткові привілеї та функції. Наприклад, адміністратор може виконувати адміністративні дії, такі як керування користувачами, налаштування системи, здійснення моніторингу та відстеження активності користувачів.

Однією з ключових можливостей адміністратора є отримання інформації про взаємодії користувачів в системі. Це може включати статистику використання, аналіз популярних запитів, виявлення паттернів поведінки користувачів та інші дані, які допомагають у покращенні функціоналу та ефективності системи. Такий

доступ до інформації дозволяє адміністраторам здійснювати інформовані рішення щодо подальшого розвитку та вдосконалення системи.

2.8 Додаткові компоненти для розробки чат-бота

Після вибору мови програмування, налаштування середовища розробки та реєстрації бота у Telegram залишається лише одне важливе завдання: вирішення питання щодо компонентів та модулів, необхідних для належної реалізації логіки бота.

API (Application Programming Interface) – це набір правил, протоколів та інструментів, які дозволяють різним програмам взаємодіяти між собою. API визначає, які запити та команди можуть бути використані для обміну даними між програмами, а також формати даних, які використовуються для цього обміну [17].

У сутності, API виступає як посередник між різними програмами, що дозволяє їм взаємодіяти і обмінюватися інформацією, незалежно від їхньої мови програмування або архітектури. Це дає можливість розробникам використовувати функціонал, який надається іншими програмами або сервісами, без необхідності знати деталі їхньої реалізації.

API може бути реалізоване у вигляді бібліотек, веб-сервісів, як RESTful API або SOAP, а також у вигляді SDK (Software Development Kit), який надає набір інструментів та документації для розробки програм, що використовують API. Загалом, API є важливим інструментом у розробці програмного забезпечення, оскільки він дозволяє підключати та використовувати функціонал інших програм та сервісів, що робить розробку більш ефективною та швидкою.

Telegram Bot API, у свою чергу, є HTTP-інтерфейсом для комунікації з ботами у Telegram. Це забезпечує можливість реєстрації бота в додатку та отримання унікального токена для взаємодії з API. Кожен запит до API має вигляд URL-посилання з вказанням протоколу передачі та необхідного токена бота, а також методу, який потрібно виконати.

3 РОЗРОБКА ТА НАЛАШТУВАННЯ ЧАТ-БОТА

3.1 Реалізація розробки телеграм чат-бота

Після завершення відбору технологій і програмного забезпечення для реалізації проекту настав час розпочати його виконання. Першим кроком в цьому напрямку є процес реєстрації бота. Досліджуючи цей аспект, важливо звернути увагу на кожен етап реєстрації, розуміючи всі вимоги та обмеження.

Засоби реєстрації ботів можуть варіюватися в залежності від обраної платформи та мети проекту. Іноді це включає в себе створення облікового запису, налаштування API ключів, підписування угод про використання послуг, а також інші дії, необхідні для отримання доступу до інструментів створення ботів.

У пошуковій стрічці Telegram ми шукаємо @botfather, який відповідає за керування всіма ботами у цьому мессенджері (рис 3.1). @BotFather є важливим інструментом для будь-якого, хто прагне створити, налаштувати чи управляти ботами у Telegram. Цей користувач надає доступ до ряду корисних функцій, включаючи створення нових ботів, налаштування їх параметрів, отримання API ключів та управління повідомленнями й командами.

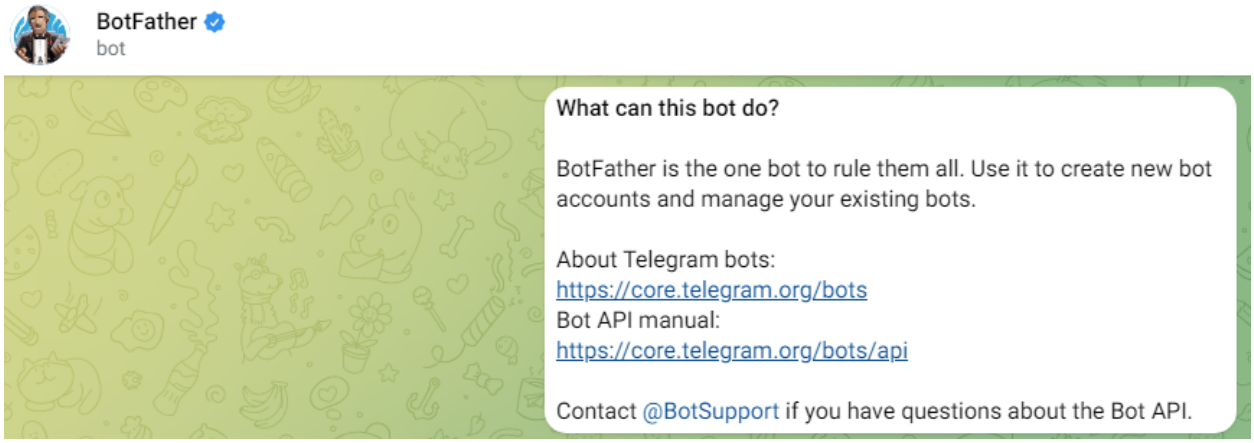


Рисунок 3.1 – Початок реєстрації бота в Telegram

Після знаходження @botfather натиснемо кнопку "Почати". Це запустить діалогове вікно, в якому ми зможем взаємодіяти з @BotFather та виконати різноманітні дії, пов'язані з управлінням ботами. Наприклад створити нового бота, надати йому ім'я та налаштувати його параметри, такі як аватарка, опис тощо.

Крім того, @botfather надає доступ до командного інтерфейсу, який дозволяє налаштовувати різні аспекти бота, включаючи команди, реакції на повідомлення, доступність для користувачів тощо.

За допомогою команди /start розпочнемо діалог з ботом. Після обробки він відправить список доступних команд та функцій (рис 3.2).

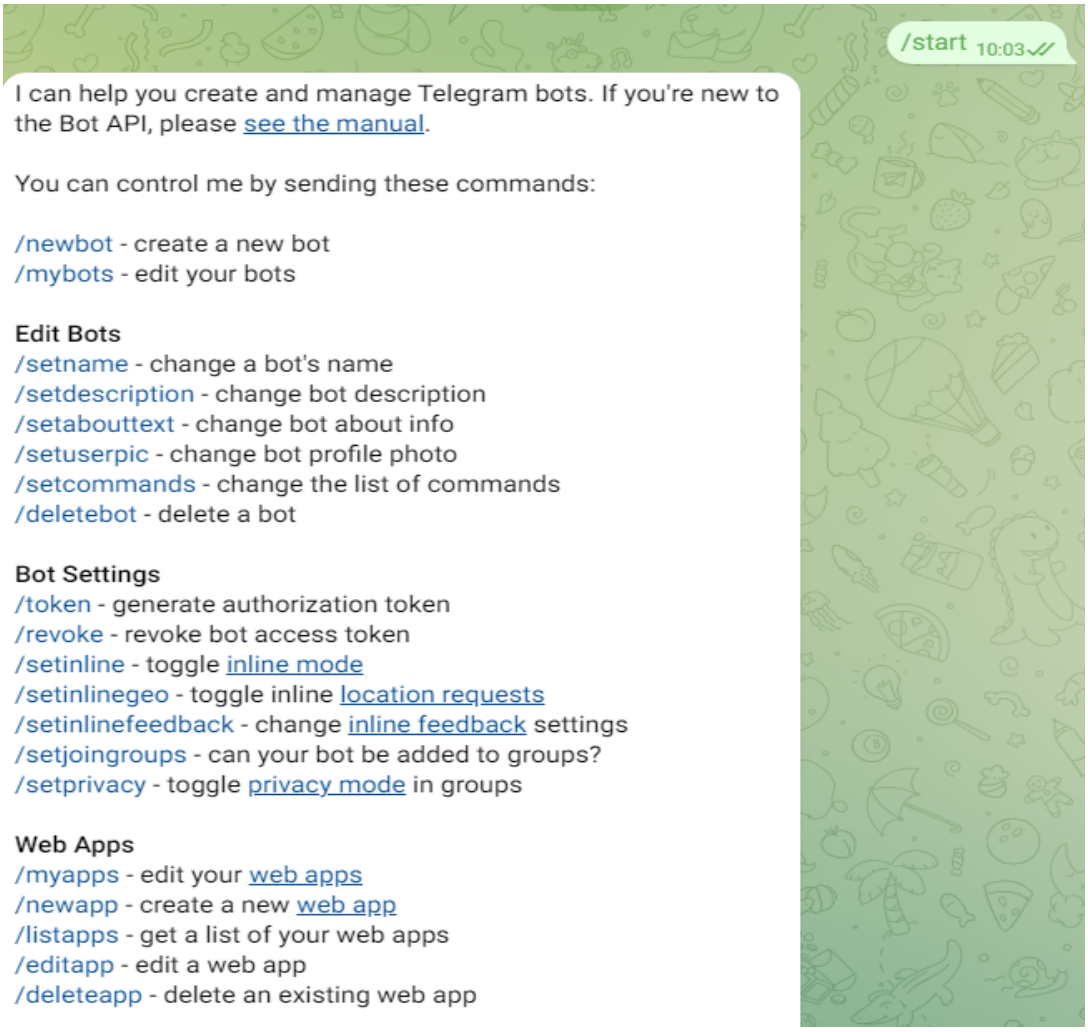


Рисунок 3.2 – Функції та команди @botfather

Для реєстрації нашого бота нас цікавить команда /newbot. Після відправки нам запропонують вибрати назву бота, та @ юзернейм. Важливо зазначити що

юзернейм в телеграм враховує реєстр символів та має містити лише латинські літери,цифри та нижню риску. Ім’я обов’язково повинно закінчуватись на bot.

Після проведення данної процедури ми отримуємо унікальний token для доступу до Telegram API та доступ до посилання на нашого новоствореного бота (рис. 3.3).

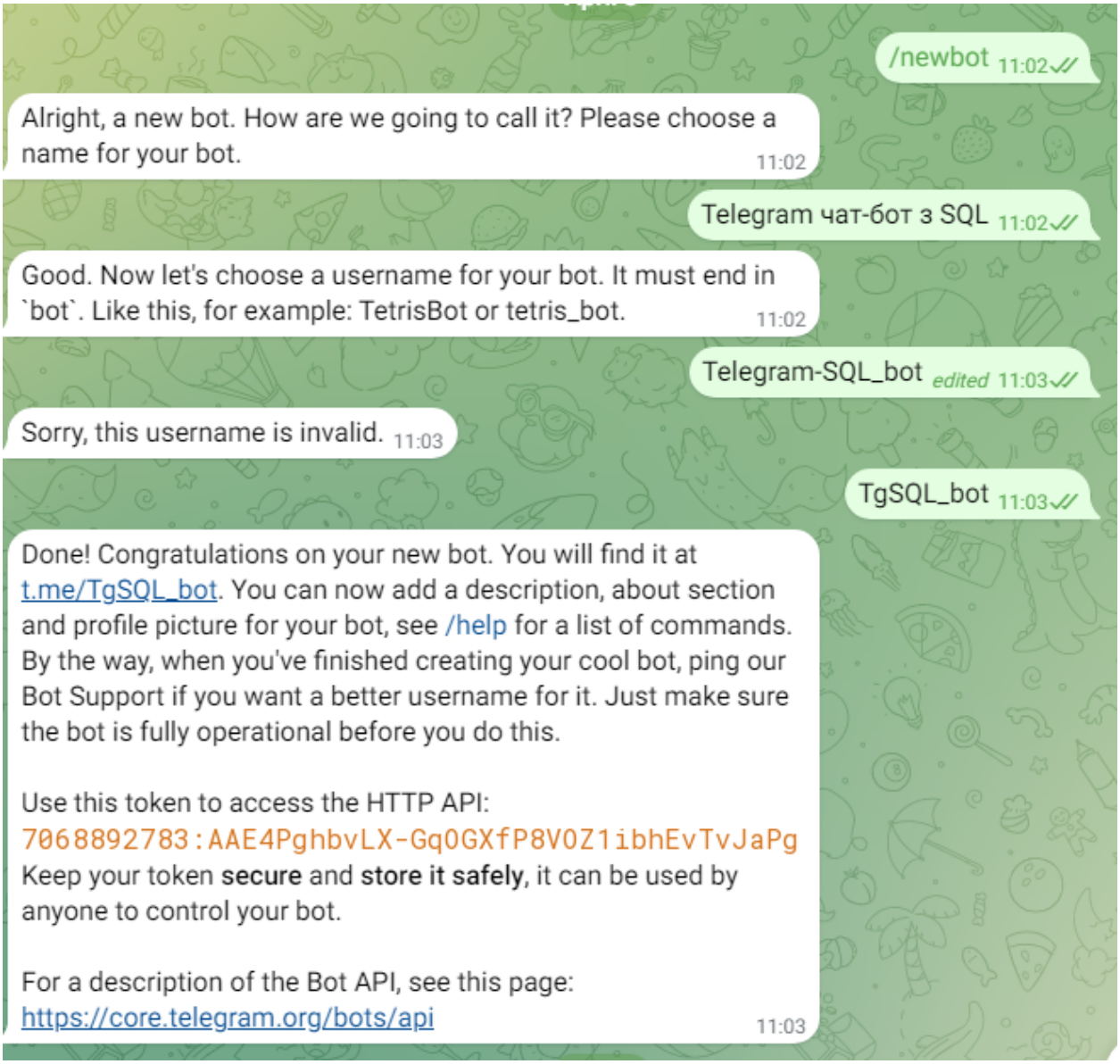


Рисунок 3.3 – Повідомлення про створення та посиланням на бота

Після отримання посилання можемо подивитися на створеного бота. Зараз він немає функціоналу, команд та оброки інформації. В подальшому він буде важливим аспектом для дипломної роботи. Важливо зазначити що перший запуск будь-якого чат-бота в Telegram виконується через команду /start. В нашому випадку

відповіді не буде, оскільки він ще не вміє відповідати на http запити. Він навчиться цьому пізніше після написання коду (рис. 3.4).

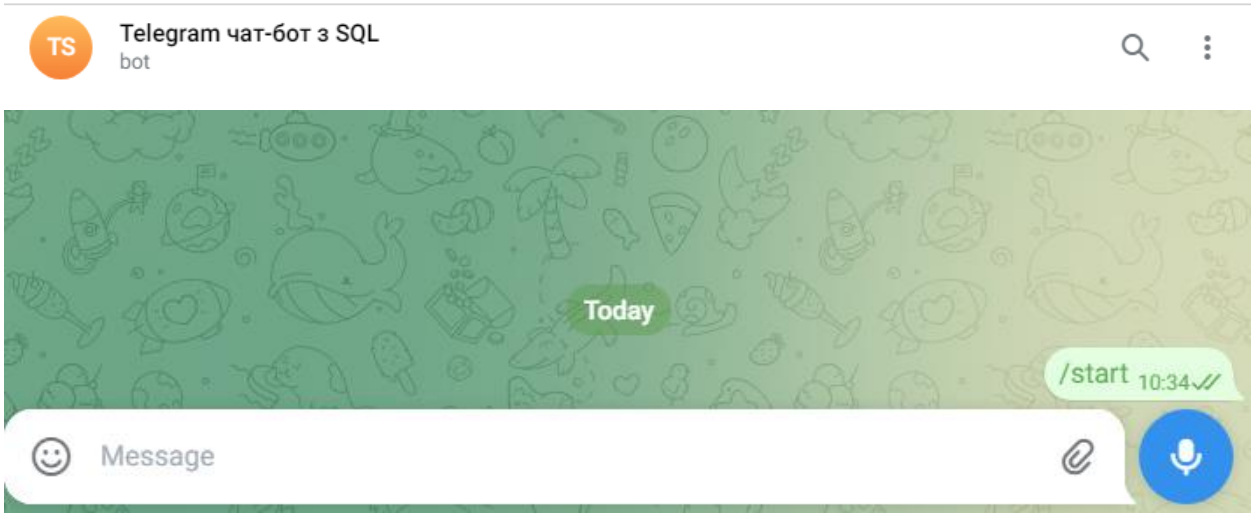


Рисунок 3.4 – Вікно чату з створеним чат-ботом

При огляді даного інформаційного блоку стає зрозумілим, що він мало відрізняється від звичайного чату з реальним користувачем. Фактично, це чатове вікно виступає як пустий аркуш для взаємодії, оскільки весь код, написаний у середовищі розробки, буде оброблюватися та виводитися у чаті бота у вигляді інформаційних вікон у текстовому форматі або у вигляді зображень та кнопок. Це є спеціальним інтерфейсом для взаємодії з програмою. Проте важливо відзначити, що всі зв'язки та шифрування не мають стосунку до протоколу шифрування Telegram. Для надсилання відповідей на запити користувача вони передаються через протокол HTTPS. Формат цих запитів має наступний вигляд: api.telegram.org/bot’token’/назва_метода [19].

Для оновлення вигляду бота варто звернутися до Bot Father і відправити команду «/setuserpic». Після цього можна надіслати в чат потрібне зображення. Після зміни графічного оформлення, повернувшись до чату з ботом, можна побачити, що зміни вже відобразилися, і у бота встановлено нове зображення профілю [21].

Звернення уваги до візуального оформлення та професійний підхід до цього аспекту можуть відігравати ключову роль у привертанні уваги користувачів.

Візуально привабливий бот здатний не лише зацікавити користувачів, але й спонукати їх до початку діалогу. Таким чином, важливо приділяти увагу не лише функціональності, але й зовнішньому вигляду свого програмного помічника (рис.3.5).

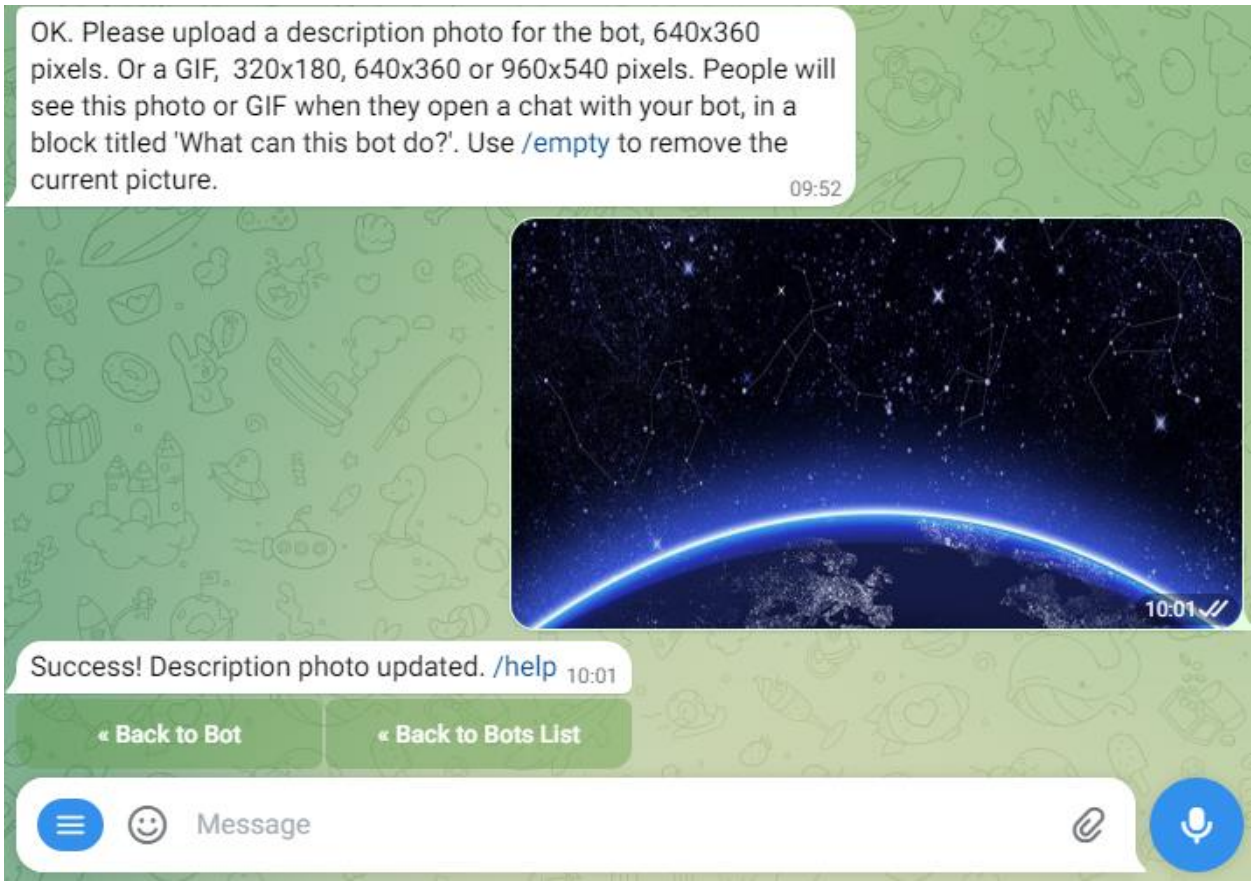


Рисунок 3.5 – Оновлення фотографії чат-бота

Наступним етапом в розробці чат-бота є налаштування середовища розробки та підключення ресурсів для зручного програмування. Першим кроком буде встановлення останньої версії Python на комп'ютер. Під час завантаження інсталятора з офіційного сайту важливо врахувати сумісність з вашою операційною системою та стабільність версії. Наприклад, найновіша версія, 3.12.2, не підтримується на ОС Windows 7 та більш старих версіях, тому вона може не підійти для нашої мети. В незалежності від нашої мети будемо працювати з цією версієї Python яка є останнім регулярним випуском і буде підтримуватися до 2024 року.

Важливо пам'ятати, що перед встановленням нової версії продукту корисно ознайомитися з нововведеннями та виправленнями, оскільки це може значно спростити процес програмування (рис. 3.6).

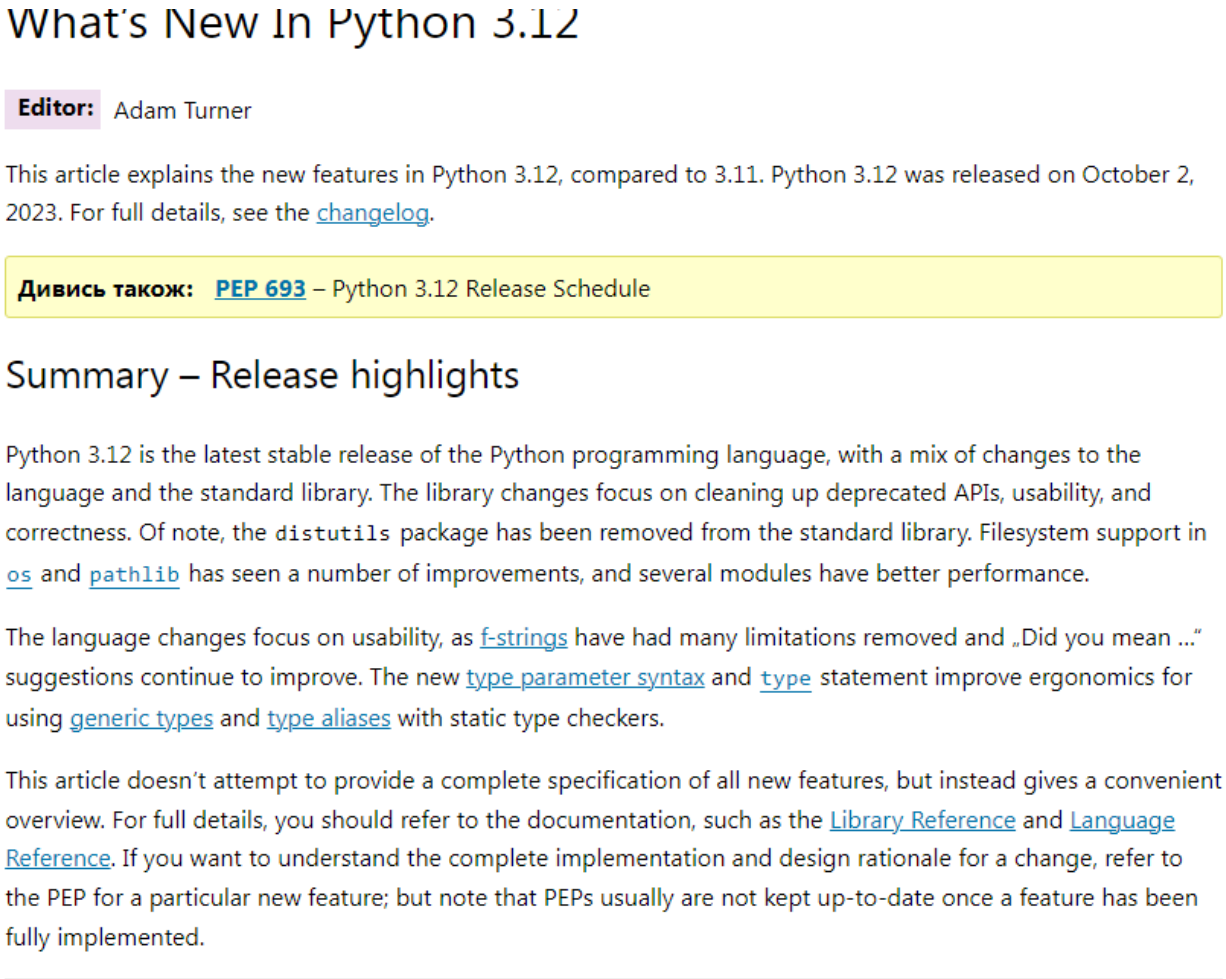


Рисунок 3.6 – Оновлення в Python 3.12.2

Після вибору актуальної версії та стабільного релізу потрібно звернути увагу на процесс встановлення мови програмування. Важливо зазначити що обов'язково потрібно вибрати пункт "add python.exe to PATH". Це дозволяє запускати команди Python з будь-якої теки у командному рядку або терміналі без необхідності вказувати повний шлях до виконуваного файлу. Встановлення python.exe в PATH спрощує роботу з Python та виклик його програм з будь-якого місця в системі (рис. 3.7).

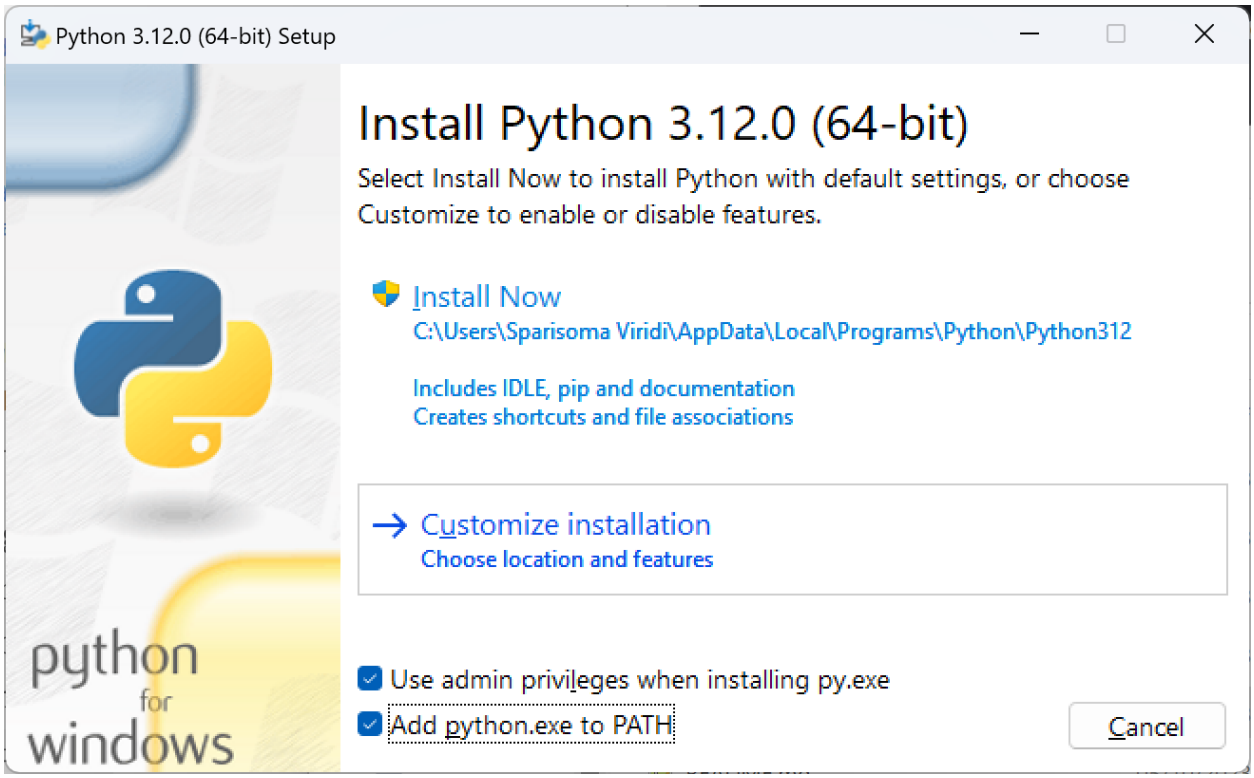


Рисунок 3.7 – Встановлення актуальної версії Python

Наступним важливим аспектом роботи є налаштування середовища розробки. Оскільки ми використовуєм PyCharm важливо оптимізувати процесс роботи, для цього в середовищі створені хот кеї (HotKeys — це комбінації клавіш на клавіатурі комп'ютера, які використовуються для виклику різноманітних функцій, операцій або команд у програмах або операційних системах.) (рис. 3.8).

Також корисно використовувати плагіни для простоти та легкості роботи. Плагіни які ми будемо використовувати в роботі: Tabnine - часто рятує під час введення довгих рядків, які він дописує самостійно. Він навіть може дописувати прості функції або логічні конструкції. Загалом, надзвичайно корисна річ. Звісно, є й мінус — більша частина функціоналу платна за підпискою. Але навіть безкоштовна версія буде корисною.

Rainbow Brackets – змінює колір дужок залежно від їх вкладеності. Дуже спрощує роботу та дозволяє швидше орієнтуватися в коді. Requirements – плагін для більш зручної роботи з файлом requirements.txt.

PyCharm

DEFAULT KEYMAP

Editing

Ctrl + Space	Basic code completion (the name of any class, method or variable)
Ctrl + Alt + Space	Class name completion (the name of any project class independently of current imports)
Ctrl + Shift + Enter	Complete statement
Ctrl + P	Parameter info (within method call arguments)
Ctrl + Q	Quick documentation lookup
Shift + F1	External Doc
Ctrl + mouse over code	Brief Info
Ctrl + F1	Show descriptions of error or warning at caret
Alt + Insert	Generate code...
Ctrl + O	Override methods
Ctrl + Alt + T	Surround with...
Ctrl + /	Comment/uncomment with line comment
Ctrl + Shift + /	Comment/uncomment with block comment
Ctrl + W	Select successively increasing code blocks
Ctrl + Shift + W	Decrease current selection to previous state
Ctrl + Shift +]	Select till code block end
Ctrl + Shift + [	Select till code block start
Alt + Enter	Show intention actions and quick-fixes
Ctrl + Alt + L	Reformat code
Ctrl + Alt + O	Optimize imports
Ctrl + Alt + I	Auto-indent line(s)
Tab	Indent selected lines
Shift + Tab	Unindent selected lines
Ctrl + X, Shift + Delete	Cut current line or selected block to clipboard
Ctrl + C, Ctrl + Insert	Copy current line or selected block to clipboard
Ctrl + V, Shift + Insert	Paste from clipboard
Ctrl + Shift + V	Paste from recent buffers...
Ctrl + D	Duplicate current line or selected block
Ctrl + Y	Delete line at caret
Ctrl + Shift + J	Smart line join
Ctrl + Enter	Smart line split
Shift + Enter	Start new line
Ctrl + Shift + U	Toggle case for word at caret or selected block
Ctrl + Delete	Delete to word end
Ctrl + Backspace	Delete to word start
Ctrl + NumPad+	Expand code block
Ctrl + NumPad-	Collapse code block
Ctrl + Shift + NumPad+	Expand all
Ctrl + Shift + NumPad-	Collapse all
Ctrl + F4	Close active editor tab

Running

Alt + Shift + F10	Select configuration and run
Alt + Shift + F9	Select configuration and debug
Shift + F10	Run
Shift + F9	Debug
Ctrl + Shift + F10	Run context configuration from editor
Ctrl + Alt + R	Run manage.py task

Debugging

F8 / F7	Step over/into
Shift + F8	Step out
Alt + F9	Run to cursor
Alt + F8	Evaluate expression
Ctrl + Alt + F8	Quick evaluate expression
F9	Resume program
Ctrl + F8	Toggle breakpoint
Ctrl + Shift + F8	View breakpoints

Navigation

Ctrl + N	Go to class
Ctrl + Shift + N	Go to file
Ctrl + Alt + Shift + N	Go to symbol
Alt + Right	Go to next editor tab
Alt + Left	Go to previous editor tab
F12	Go back to previous tool window
Esc	Go to editor (from tool window)
Shift + Esc	Hide active or last active window
Ctrl + Shift + F4	Close active run/messages/find/... tab
Ctrl + G	Go to line
Ctrl + E	Recent files popup
Ctrl + Alt + Right	Navigate forward
Ctrl + Alt + Left	Navigate back
Ctrl + Shift + Backspace	Navigate to last edit location
Alt + F1	Select current file or symbol in any view
Ctrl + B, Ctrl + Click	Go to declaration
Ctrl + Alt + B	Go to implementation(s)
Ctrl + Shift + I	Open quick definition lookup
Ctrl + Shift + B	Go to type declaration
Ctrl + U	Go to super method/super class
Alt + Up / Down	Go to previous/next method
Ctrl +] / [	Move to code block end/start
Ctrl + F12	File structure popup
Ctrl + H	Type hierarchy
Ctrl + Shift + H	Method hierarchy
Ctrl + Alt + H	Call hierarchy
F2 / Shift + F2	Next/previous highlighted error
F4	Edit source
Ctrl + Enter	View source
Alt + Home	Show navigation bar
F11	Toggle bookmark
Ctrl + Shift + F11	Toggle bookmark with mnemonic
Ctrl + #[0-9]	Go to numbered bookmark
Shift + F11	Show bookmarks

Search/Replace

Ctrl + F / Ctrl + R	Find/Replace
F3 / Shift + F3	Find next/previous
Ctrl + Shift + F	Find in path
Ctrl + Shift + R	Replace in path

Usage Search

Alt + F7 / Ctrl + F7	Find usages / Find usages in file
Ctrl + Shift + F7	Highlight usages in file
Ctrl + Alt + F7	Show usages

Refactoring

F5 / F6	Copy / Move
Alt + Delete	Safe Delete
Shift + F6	Rename
Ctrl + F6	Change Signature
Ctrl + Alt + N	Inline
Ctrl + Alt + M	Extract Method
Ctrl + Alt + V	Extract Variable
Ctrl + Alt + F	Extract Field
Ctrl + Alt + C	Extract Constant
Ctrl + Alt + P	Extract Parameter

VCS/Local History

Ctrl + K	Commit project to VCS
Ctrl + T	Update project from VCS
Alt + Shift + C	View recent changes
Alt + BackQuote (`)	VCS quick popup

Live Templates

Ctrl + Alt + J	Surround with Live Template
Ctrl + J	Insert Live Template

General

Alt + #[0-9]	Open corresponding tool window
Ctrl + S	Save all
Ctrl + Alt + Y	Synchronize
Ctrl + Shift + F12	Toggle maximizing editor
Alt + Shift + F	Add to Favorites
Alt + Shift + I	Inspect current file with current profile
Ctrl + BackQuote (`)	Quick switch current scheme
Ctrl + Alt + S	Open Settings dialog
Ctrl + Shift + A	Find Action
Ctrl + Tab	Switch between tabs and tool windows

To find any action inside the IDE use Find Action (Ctrl+Shift+A)

jetbrains.com/pycharm blog.jetbrains.com/pycharm @pycharm

Рисунок 3.8 – Хот кеї в середовищі програмування PyCharm [6]

Наступним важливим аспектом є вибір на налаштування бази даних. SQLite чудово підходить для написання телеграм-чат ботів - це локальна база даних, яка не вимагає окремого сервера. Це означає, що не потрібно налаштовувати складну інфраструктуру для роботи з ним, що особливо зручно для невеликих проєктів, таких як телеграм-чат боти. SQLite працює швидко, оскільки база даних зберігається локально на диску. Це означає, що телеграм-чат бот може швидко звертатися до бази даних для отримання необхідної інформації під час обробки запитів користувачів. База даних не вимагає значних обсягів ресурсів для роботи. Це означає, що ваш телеграм-чат бот може працювати ефективно навіть на обмежених серверах або на пристроях з низькими технічними характеристиками [8].

Наступним етапом є встановлення та налаштування бази даних. Оскільки SQLite є безкоштовним дистриб'ютором то скачуємо та встановлюємо з офіційного сайту (рис. 3.9).

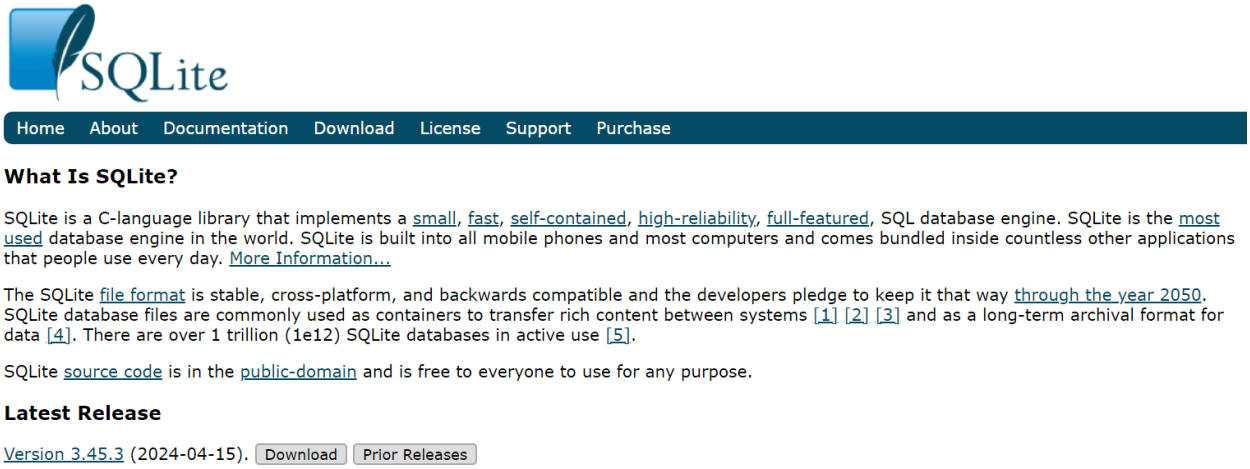


Рисунок 3.9 – Завантаження та встановлення 3.45.3 SQLite

Підготуємо базу даних для майбутнього бота. Налаштовуємо для унікального користувача (Primary Key) - додаємо значення які буде витягувати бот з бази даних (операції, значення, дати) (рис. 3.10).

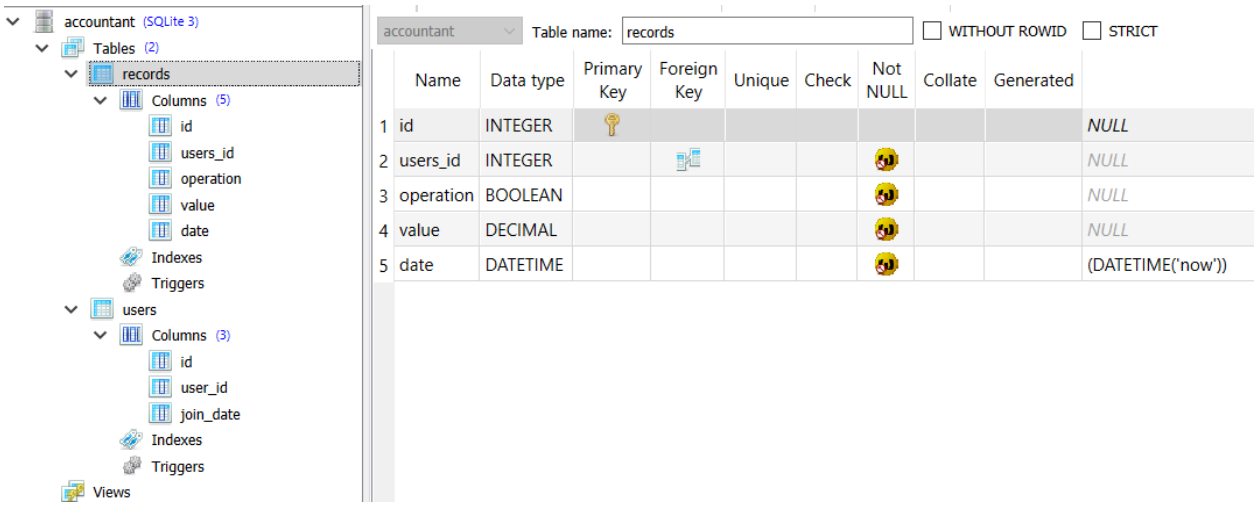


Рисунок 3.10 – Налаштування та створення бази даних

Дані які будуть надходити в базу даних будуть відображатися в таблицях Records та Users в пункті Data (рис. 3.11).

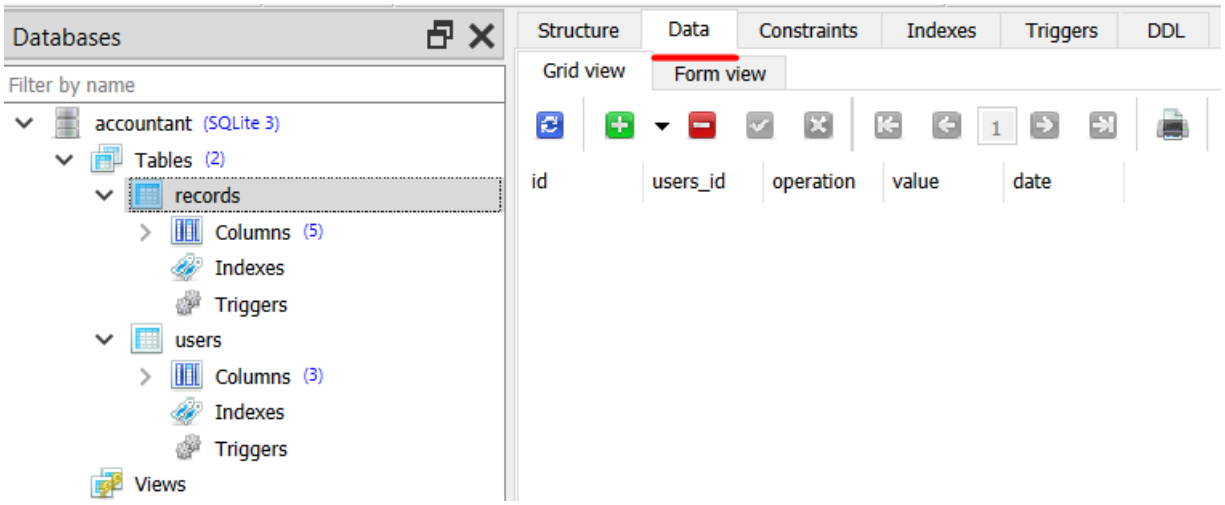


Рисунок 3.11 – Збереження інформації в пункті Data

Для початку розробки телеграм бота потрібно встановити бібліотеку Aiogram через команду PIP – це інструментарій для керування пакунками у Python. Він дозволяє легко встановлювати, оновлювати, видаляти та керувати пакунками Python з використанням віддалених репозиторіїв, таких як PyPI (Python Package Index) або приватних репозиторіїв [9], [11] (рис. 3.12).

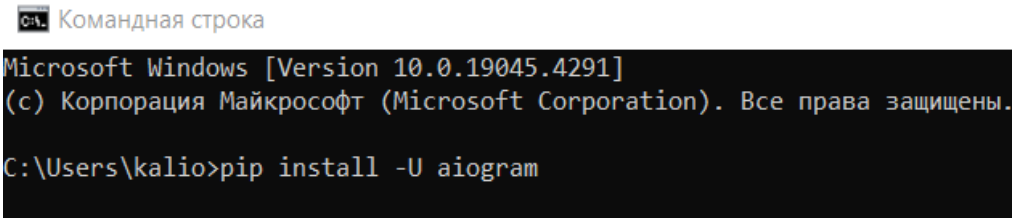


Рисунок 3.12 – Встановлення бібліотеки Aiogram

Основні команди PyPI

pip install package_name: Встановлює цей пакет.

pip uninstall package_name: Видаляє цей пакет.

pip list: Показує встановлені пакети.

pip show package_name: Виводить інформацію про конкретний пакет.

pip search query: Шукає пакети, пов'язані із запитом [10].

Встановлювати бібліотеку SQLite немає потреби, адже вона вже інтегрована в середовище програмування PyCharm.

Після повного налаштування робочого простору та додавання необхідних бібліотек, був створений головний файл у форматі ".py". У цьому файлі були імпортовані основні бібліотеки та модулі, щоб забезпечити коректну роботу їх компонентів та команд під час написання коду. Імпорт модулів відбувався безпосередньо в вікні програми у вигляді звичайного коду програми. Це забезпечило відсутність помилок у коді під час розробки (рис. 3.13). Щоб не плутатись по коду розділяємо код по модулям. Створюємо головний файл (bot.py), файл для роботою з базою даних (db.py), інтерфейс бібіотеки Aiogram (dispatcher.py), редагування фільтрів (filters.py), файл ініціалізації (_init_.py), конфіг в якому ми будемо зберегати наш телеграм токен (config.py), файл з налаштуванням бота (personal_actions.py) також в деректорії буде зберегатись файл з базою даних (DataBase.db) (рис.3.14).

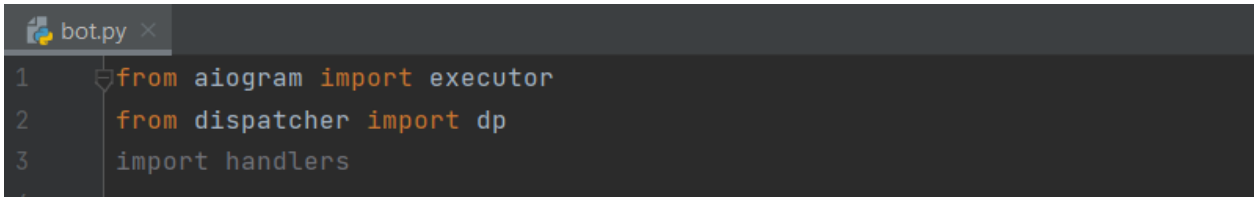


Рисунок 3.13 – Приклад імпорту модулів в Python

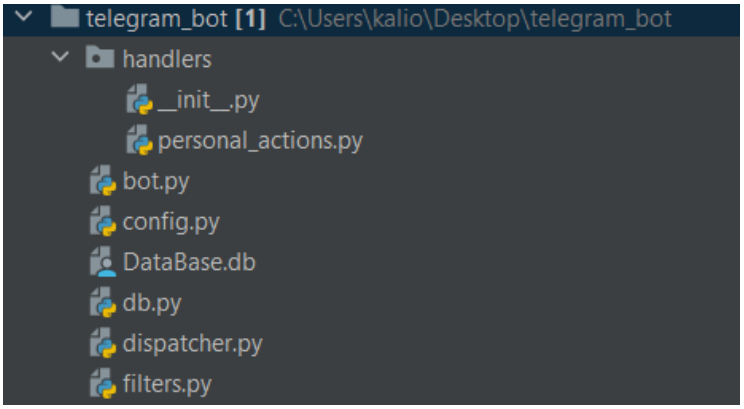


Рисунок 3.14 – Розподіл по модулям

Для того, щоб бот розпочав реагувати на повідомлення у чаті та міг пропонувати свої послуги, необхідно створити команду "/start". Після виклику цієї команди бот відправить користувачеві меню з вибором інформації. Додатково був доданий ключ, який був наданий Bot Father раніше, для того, щоб саме потрібний

бот отримував команди, що генеруються у коді програми, цей ключ буде зберігатись в файлі config.py (рис. 3.15).

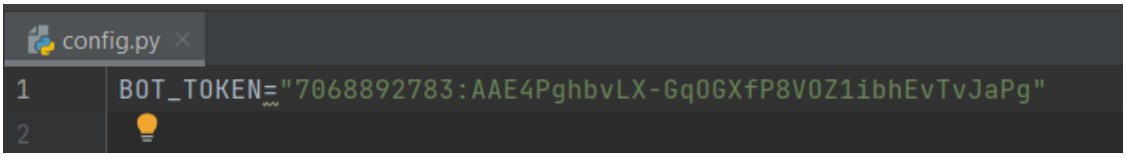


Рисунок 3.15 – Токен телеграм чат-бота

Коли ключ доступу надійшов у програму, можна безпосередньо звертатися до бота за допомогою команд, які існують у завантажених віртуальних середовище проекту бібліотек. Це спрощує взаємодію з ботом та забезпечує можливість автоматичної обробки повідомлень.

Наступним важливим аспектом є підключення бота до SQLite бази даних. Задля цього ми налаштували її раніше. Телеграм-бот може взаємодіяти з базою даних SQLite для зберігання, отримання та оновлення інформації про користувачів, їх повідомлення та інші дані. Взаємодія з базою даних SQLite може бути реалізована за допомогою стандартного модуля SQLite3 в Python. Цей модуль дозволяє виконувати різні операції з базою даних, такі як виконання SQL-запитів, збереження змін у базі даних та інше. Наприклад, бот може використовувати запити SQL для зберігання інформації про користувачів та їх повідомлення у відповідних таблицях бази даних, і використовувати ці дані для відповіді на запити користувачів або внутрішніх операцій бота. Для інтеграції бази даних потрібно створити конструктор класу який ініціалізує об’єкт для взаємодії з базою даних SQLite, не забувши про імпорт бібліотеки Sqlite3 (рис. 3.16) [18].

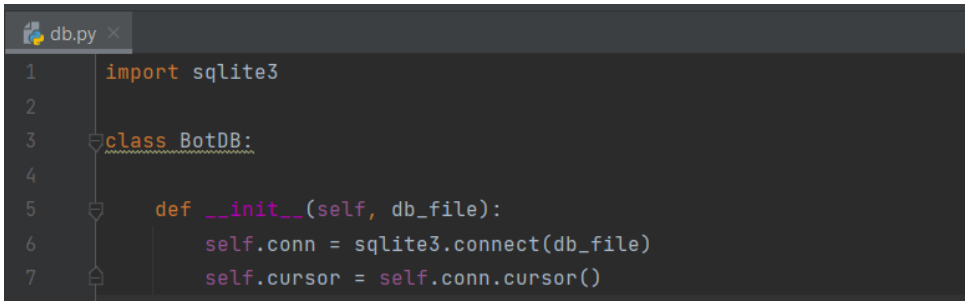


Рисунок 3.16 – Конструктор класу бази даних SQLite

Бібліотека Sqlite3 в Python надає зручний інтерфейс для взаємодії з базами даних SQLite. Вона дозволяє створювати, підключатися до, запитувати та змінювати дані в базах даних SQLite за допомогою мови Python. Ось основні аспекти синтаксису бібліотеки які будуть використані в телеграм чат-боті щоб отримувати\додавати інформацію з БД (рис. 3.17 – 3.21).

```
def user_exists(self, user_id):
    """Перевірка чи є користувач в базі"""
    result = self.cursor.execute("SELECT `id` FROM `users` WHERE `user_id` = ?", (user_id,))
    return bool(len(result.fetchall()))
```

Рисунок 3.17 – Перевірка чи є користувач в БД

```
def get_user_id(self, user_id):
    """Достаєм id користувача в базі по його user_id"""
    result = self.cursor.execute("SELECT `id` FROM `users` WHERE `user_id` = ?", (user_id,))
    return result.fetchone()[0]
```

Рисунок 3.18 – Запит до бази даних SQLite, де вибираємо id користувача за його user_id.

```
def add_user(self, user_id):
    """Добавляємо користувача в базу"""
    self.cursor.execute("INSERT INTO `users` (`user_id`) VALUES (?)", (user_id,))
    return self.conn.commit()
```

Рисунок 3.19 – Створення унікального користувача (Prime user)

```
def add_record(self, user_id, operation, value):
    """Створення запису про дохід/витрати"""
    self.cursor.execute("INSERT INTO `records` (`users_id`, `operation`, `value`) VALUES (?, ?, ?)",
        (self.get_user_id(user_id),
        operation == "+",
        value))
    return self.conn.commit()
```

Рисунок 3.20 – Внесення запитів до бази даних

```
def get_records(self, user_id, within_="all"):
    """Отримуємо історію про дохід/витрату"""

    if within == "day":
        result = self.cursor.execute("SELECT * FROM `records` WHERE `users_id` = ? AND `date` BETWEEN datetime('now', 'start of day') AND datetime('now', 'localtime')",
            (self.get_user_id(user_id),))
    else:
        result = self.cursor.execute("SELECT * FROM `records` WHERE `users_id` = ? ORDER BY `date`",
            (self.get_user_id(user_id),))

    return result.fetchall()

def close(self):
    """Закриваєм звязок з БД"""
    self.connection.close()
```

Рисунок 3.21– Запит про історію внесених запитів та закриття бази даних

Для того, щоб бот розпочав взаємодію з користувачем у чаті та надавав йому доступ до своїх послуг, необхідно створити команду "/start". Після активації цієї команди бот автоматично відправить користувачеві меню з різними варіантами інформації або послуг, які він може обрати. Наступна команда дозволяє боту розпізнавати та обробляти текстовий контент а також добавити id користувача в базу даних (рис. 3.22).

```
@dp.message_handler(commands="start")
async def start(message: types.Message):
    if not BotDB.user_exists(message.from_user.id):
        BotDB.add_user(message.from_user.id)
```

Рисунок 3.22 – Створення розпізнавання команди /start

Після розпізнавання команди та внесення нового користувача в базу даних потрібно щоб бот почав обробляти інформації та давати відповіді на запитання (рис. 3.23).

```
await message.bot.send_message(message.from_user.id, "Добрий день! Я Ваш фінансовий бот", reply_markup=keyboard)
```

Рисунок 3.23 – Відповідь бота на команду /start

Після вивчення роботи команди "/start" і виявлення її значення, наступним кроком було сформування загального концепту функціонування бота. Аналізуючи різноманітні варіанти оформлення інтерфейсу чату, прийшли до висновку, що додавання кнопок для зручної оброки інформації є важливим кроком. Це забезпечить якісну навігацію для користувачів бота. Однак перед початком реалізації необхідно з'ясувати різницю між різними типами кнопок у Telegram, щоб визначити найбільш підходящий варіант для використання у проекті.

ReplyKeyboardMarkup – це шаблони повідомлень. Наприклад, ваш бот ставить користувачеві запитання і пропонує варіанти відповіді. Користувач може самостійно надрукувати відповідь або натиснути на готову кнопку. Така клавіатура відображається замість основної і не прив'язана до жодного повідомлення. У кнопки такої клавіатури не можна закласти ніякої інформації, не можна

запрограмувати для неї подібний, якщо користувач натискає кнопку з текстом «abc» відправити текст «qwerty» алгоритм, надіслано буде тільки те, що написано на кнопці (рис 3.24).

```
keyboard = types.ReplyKeyboardMarkup(resize_keyboard=True)
keyboard.add(types.KeyboardButton("Дохід"))
keyboard.add(types.KeyboardButton("Витрати"))
keyboard.add(types.KeyboardButton("Історія"))
```

Рисунок 3.24 – Приклад коду для створення кнопок в Телеграм боті

У цьому прикладі наведено створення кнопок. Такі кнопки призначені лише для відображення тексту, що написаний на ній, і використовується переважно в звичайних чат-ботах, які здійснюють базові функції, наприклад, взаємодію з користувачем. Інтерфейс даних кнопок можна побачити нижче (рис. 3.25).

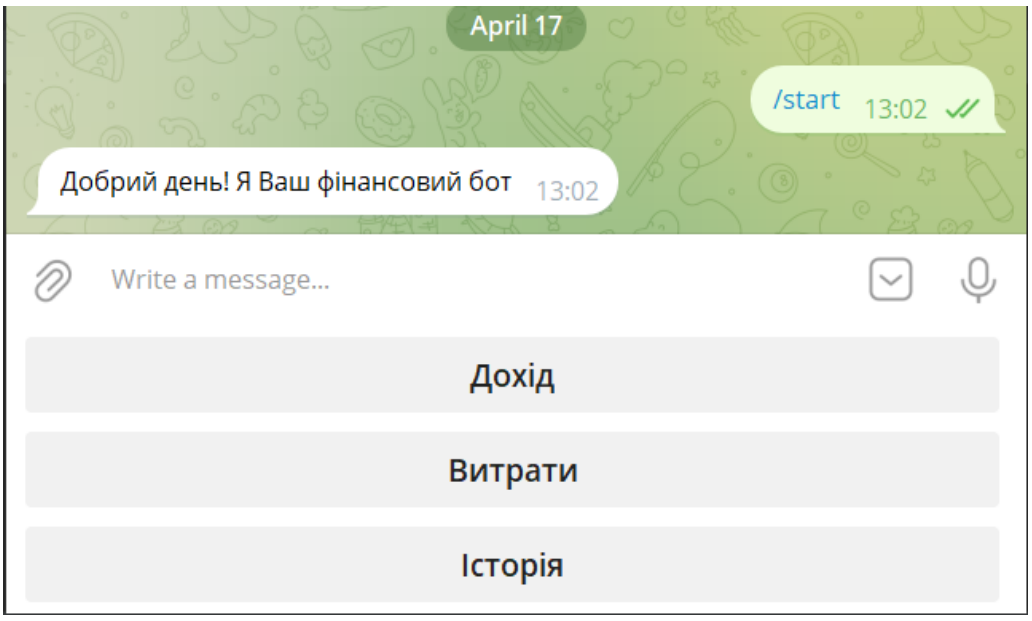


Рисунок 3.25 – Інтерфейс ReplyKeyboardMarkup кнопок

InlineKeyboardMarkup – це вже справжня кастомна клавіатура. З її допомогою ми можемо виконувати складніші дії. Вона прив'язується до повідомлення, з яким було відправлено. На кнопки можна закласти будь-який текст розміром від 1 до 64 байт (будьте обережні, недобросовісні клієнти дозволяють змінювати ці дані). Інлайн кнопки дозволяють приховати в собі внутрішню телеграму посилання,

посилання на зовнішній ресурс, а також шорткат для інлайн запиту (про інлайн режимі в одному з наступних уроків) (рис. 3.26).

Рисунок 3.26 – Приклад InlineKeyboardMarkup

І ту й іншу клавіатуру можна редагувати, але у різний спосіб. Перша оновлюється під час надсилання повідомлення з новою клавіатурою типу ReplyKeyboardMarkup, другу можна редагувати разом із повідомленням, до якого вона прикріплена (або лише саму розмітку) [12].

Наступним кроком після створення кнопок є інтеграція бази даних в наш чат-бот, створення логічної частини та створення дій бота на запити від користувача.

Фрагмент коду на рис 3.27 реєструє обробник повідомлень, який реагує на повідомлення, що містять текст "Дохід" або "Витрати". Після виявлення такого повідомлення бот відправляє відповідь, запрошуючи користувача ввести суму доходу або витрат, залежно від тексту, що надійшов.

```
@dp.message_handler(lambda message: message.text in ("Дохід", "Витрати"))
async def operation_handler(message: types.Message):
    operation_type = "+" if message.text == "Дохід" else "-"
    await message.reply(f"Введіть суму {message.text.lower()}:")
```

Рисунок 3.27 – Обробка повідомлень які надійшли від користувача

Наступний фрагмент коду обробляє повідомлення з текстом "Історія". Після виявлення такого повідомлення бот відправляє відповідь, повідомляючи користувача про початок показу історії операцій. Потім він отримує записи з бази даних, пов'язаних з користувачем, і формує відповідь з історією операцій. Якщо у користувача немає жодного запису, бот повідомляє його про це (рис. 3.28).

```python
@dp.message_handler(lambda message: message.text == "Історія")
async def history_handler(message: types.Message):
    await message.reply("Ваша історія операцій:")
    # Отримуємо історію операцій з бази даних
    records = BotDB.get_records(message.from_user.id, "day")

    if len(records) > 0:
        # Формуємо повідомлення з історією операцій
        answer = "🕘 Історія операцій:\n\n"
        for record in records:
            answer += "<b>" + ("➖ Витрати" if not record[2] else "➕ Дохід") + "</b>"
            answer += f" - {record[3]}"
            answer += f" <i>({record[4]})</i>\n"
        await message.reply(answer)
    else:
        await message.reply("Записів не виявлено!")
```

Рисунок 3.28 – Обробка повідомлення "Історія" від користувача.

Наступним кроком є створення декоратора. Декоратор – це функція в Python, яка приймає іншу функцію або клас як аргумент, додає якісь додаткові можливості до цієї функції або класу і повертає змінену або оновлену версію. У Python декоратори зазвичай використовуються для зміни поведінки функцій або методів, не змінюючи їхнього вихідного коду. Вони дозволяють додавати функціональність до існуючих функцій або методів, роблячи їх більш гнучкими і розширюваними. Декоратори використовуються для означення спеціальних маркерів, які позначають функції або методи, які потребують додаткової обробки. Коли викликається функція або метод, обгортка декоратора виконує додаткові дії перед, під час або після виконання основної функціональності. Сам декоратор буде призначений для обробки повідомлення, які мають валідний формат числа з плавучою кнопкою та опціональним знаком + або – (рис. 3.29).

```python
@dp.message_handler(lambda message: re.match(r"^[+-]?\d{1,3}(?:[ ]?\d{3})*(?:\.\d+)?$", message.text))
async def amount_handler(message: types.Message):
    if message.text.startswith("+"):
        operation_type = "+"
        operation_text = "доходу"
    elif message.text.startswith("-"):
        operation_type = "-"
        operation_text = "витрати"
    else:
        # Якщо користувач ввів неправильний формат
        await message.reply("Неправильний формат. Введіть суму ще раз.")
        return
```

Рисунок 3.29 – Логічна частина коду

Також, якщо користувач введе дані без опціонального знаку, програма не допустить запис даних до бази даних та виведе помилку: "Неправильний формат. Введіть суму ще раз." (рис. 3.30). Це допоможе забезпечити точність та коректність введених даних у базу даних, уникнути непорозумінь і сприяти зручності користувачів щоб вони вносили лише коректні дані в базу даних. Такий підхід зменшить ймовірність помилок та спростить процес обробки інформації.

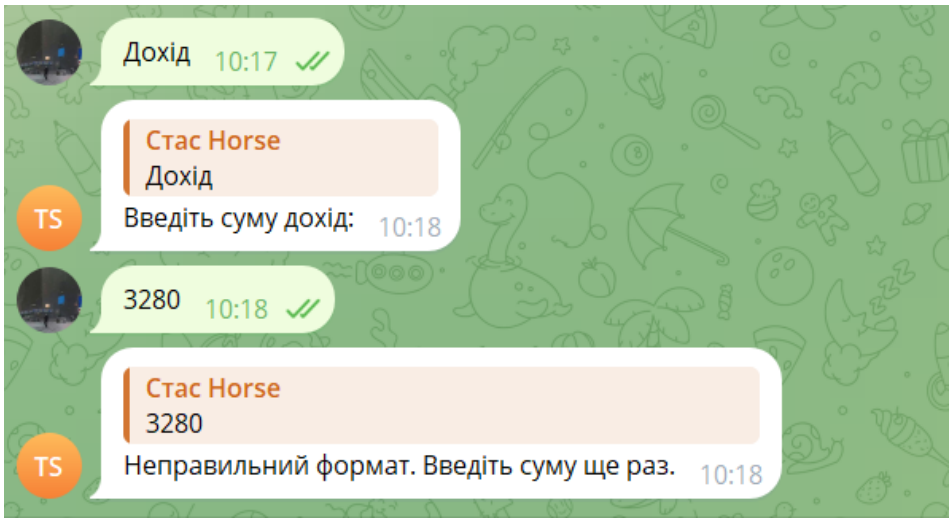


Рисунок 3.30 – Повідомлення про помилку введення даних

Наступний код видаляє всі символи, крім цифр та крапки з тексту, який користувач вводить. Це здійснюється за допомогою регулярного виразу `r"[^\d.]"`, який вказує на будь-який символ, що не є цифрою або крапкою, і замінює його на порожній рядок. Отримане значення конвертується у числовий формат `float`. Записується до бази даних (`BotDB.add_record`) із відомостями про користувача (`message.from_user.id`), тип операції (`operation_type`) та значення (`value`). Якщо тип операції є мінусом (`operation_type == "-"`), повідомляється користувачеві про успішне внесення запису про витрати. У протилежному випадку, коли тип операції не є мінусом, повідомляється про успішне внесення запису про дохід (рис. 3.31, 3.32).

```
value = float(re.sub(r"[^\d.]", "", message.text))  # Видаляємо всі символи, крім цифр і крапки
BotDB.add_record(message.from_user.id, operation_type, value)

if operation_type == "-":
    await message.reply("✅ Запис про витрати успішно внесено!")
else:
    await message.reply("✅ Запис про дохід успішно внесено!")
```

Рисунок 3.31 – Код інформації про внесення даних до БД

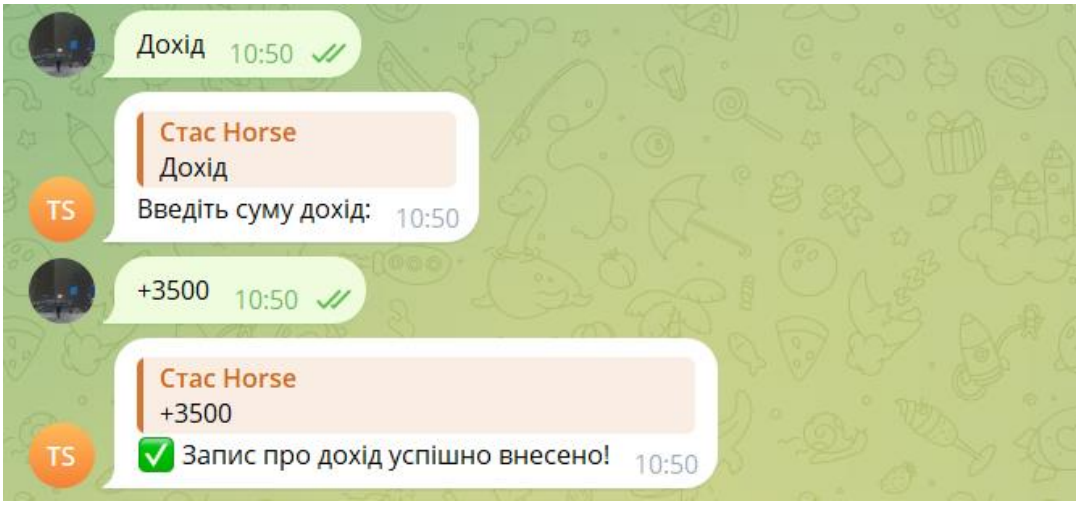


Рисунок 3.32 – Отримана користувачем інформація про запис в БД

Оскільки логічна частина чат-бота розроблена потрібно дізнатися чи коректно вводяться дані про користувача та надана інформація в базу даних SQLite. Для цього потрібно перейти в таблицю records та пункт Data (рис. 3.33, 3.34).

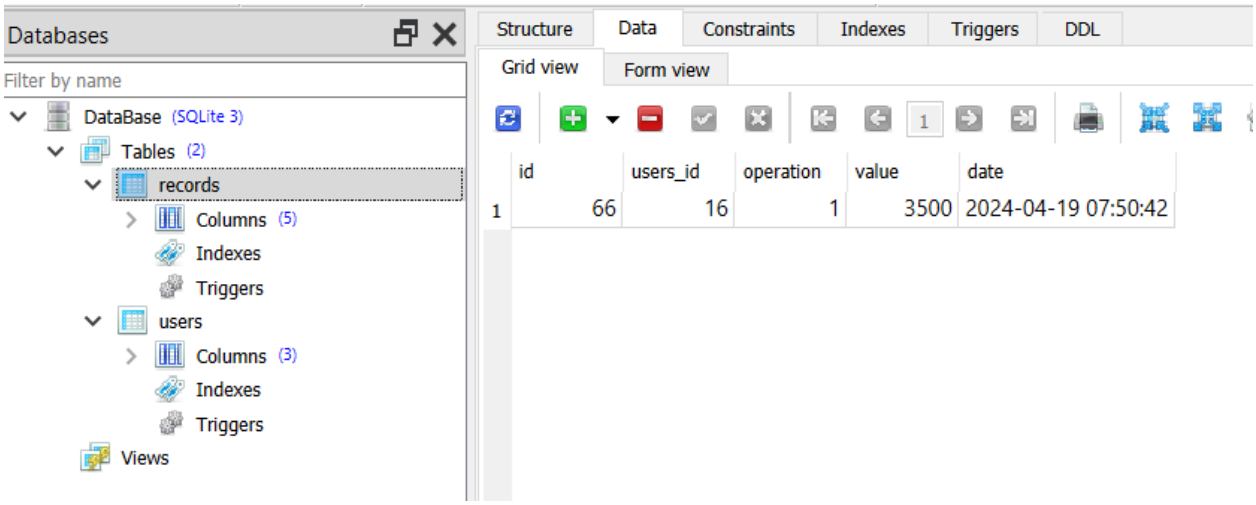


Рисунок 3.33 – Внесена інформація від користувача в БД

Кожен user_id зберігається та закріплюється за унікальним користувачем, щоб забезпечити ідентифікацію користувача і зберігання його особистих даних у базі даних.

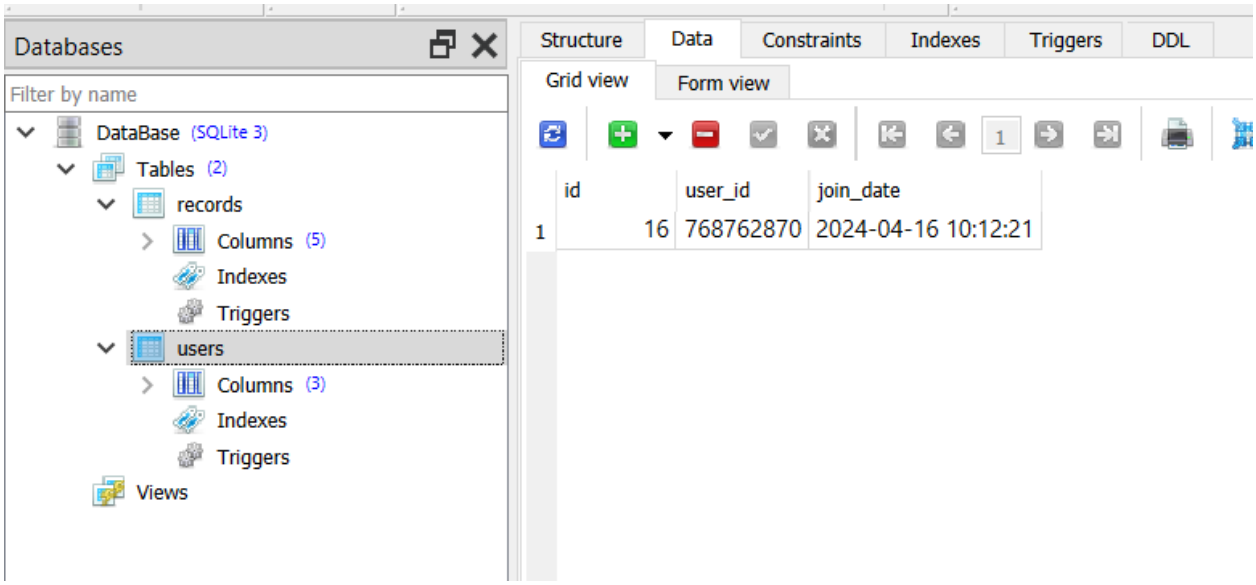


Рисунок 3.34 – Внесення унікального користувача в БД

Після успішного створення та налагодження нашого чат-боту для платформи Telegram настає час забезпечити його доступність для користувачів. Це включає в себе процес розгортання на веб-сервері, щоб чат-бот був доступний через Інтернет. Один зі способів здійснити це скористатися хмарною платформою PythonAnywhere.

PythonAnywhere - це сервіс, який дозволяє запускати та хостити веб-додатки, написані на Python, в хмарному середовищі. Після завершення розробки чат-боту ми можемо внести його на PythonAnywhere для забезпечення його постійної доступності.

Першим кроком буде створення віртуального середовища Python на PythonAnywhere, щоб мати можливість виконати наш код. Потім ми завантажуємо файли нашого чат-боту, включаючи сам код, файли налаштувань та всі необхідні залежності (рис. 3.35).

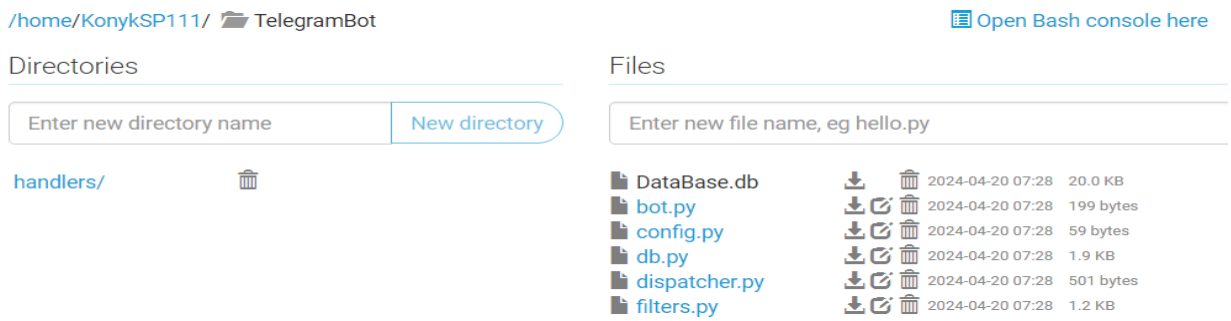


Рисунок 3.35 – Завантаження програмного коду на PythonAnywhere

При спробі запустити наш чат-бот на PythonAnywhere можуть виникнути численні проблеми, особливо пов'язані з встановленням необхідних бібліотек та залежностей в директорію сервісу. Це може стати причиною багатьох помилок, які можуть перешкодити успішному запуску бота. Однією з найпоширеніших проблем є неправильне встановлення або відсутність деяких бібліотек, необхідних для роботи нашого чат-бота. Це може бути викликано різними факторами, включаючи несумісність версій бібліотек, помилки в розширеннях або навіть проблеми з доступом до мережі для завантаження бібліотек. Відкриємо командне вікно та встановив потрібні бібліотеки (рис. 3.36).

```
07:31 ~/TelegramBot $ pip install -U aiogram
Defaulting to user installation because normal site-packages is not writeable
Looking in links: /usr/share/pip-wheels
Collecting aiogram
  Downloading aiogram-3.4.1-py3-none-any.whl (513 kB)
```

Рисунок 3.36 – Приклад встановлення бібліотеки на веб-сервісі PythonAnywhere

Для встановлення відсутніх бібліотек можна скористатися інструментами керування пакетами Python, такими як pip, для їхнього додавання до середовища виконання. Також може знадобитися оновлення версій існуючих бібліотек для забезпечення сумісності та вирішення конфліктів.

Один з переваг PythonAnywhere полягає в тому, що він надає можливість моніторингу та керування веб-додатками, що запущені на їх серверах. Це дозволяє вчасно реагувати на будь-які проблеми з функціонуванням чат-боту та забезпечує його надійну роботу для користувачів. Крім того, сервіс пропонує різні плани та можливості масштабування, що дозволяє збільшувати або зменшувати ресурси,

необхідні для нашого чат-боту в залежності від обсягу роботи та потреб користувачів. Завдяки PythonAnywhere ми можемо ефективно розгорнути та управляти нашим Telegram чат-ботом, забезпечуючи його доступність та надійність для користувачів, що забезпечує позитивний досвід взаємодії з нашим ботом [13].

Коли ми запускаємо веб-сервіс для нашого телеграм чат-бота на PythonAnywhere, спочатку ми переходимо до пункту "Web" в панелі керування PythonAnywhere. Потім ми обираємо директорію, яка буде ініціалізована на веб-сервері (рис. 3.37). Ця директорія міститиме наш код, включаючи файли програми бота та всі необхідні ресурси. Ініціалізація цієї директорії на веб-сервісі PythonAnywhere дозволить нам запустити наш телеграм чат-бот, який буде доступний для користувачів в будь-який час. Після ініціалізації директорії ми можемо налаштувати веб-сервер для запуску нашого коду, щоб обробляти запити від користувачів і відправляти їм відповіді через телеграм API. Після цього наш чат-бот буде готовий функціонувати в будь-який час, і користувачі зможуть спілкуватися з ним через телеграм, незалежно від часу та місцезнаходження.

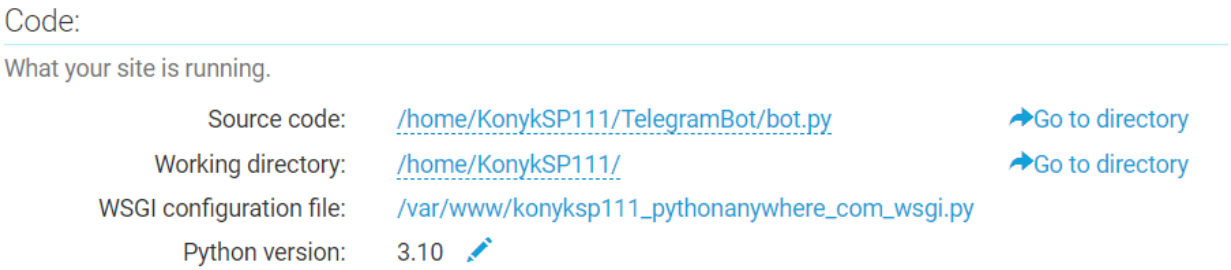


Рисунок 3.37 – Вибір директорії для запуску онлайн телеграм-чат боту

При користуванні безкоштовною версією сервісу PythonAnywhere важливо мати на увазі, що потрібно періодично оновлювати дані. За замовчуванням, користувачі безкоштовного плану отримують три місяці безкоштовного користування після останнього оновлення. Після цього періоду, якщо дані не оновлювати, вони можуть бути видалені з серверів.

Отже, щоб забезпечити безперебійну роботу телеграм чат-бота на PythonAnywhere, слід регулярно оновлювати дані, не розміщуючи на сервері ніякої діяльності протягом трьох місяців. Це можна зробити шляхом перезавантаження веб-сервера або вручним оновленням файлів у вашій директорії проекту. Такий підхід до забезпечення безперебійної роботи телеграм чат-бота на PythonAnywhere дійсно може бути ефективним. Регулярне оновлення даних без активної діяльності на сервері протягом трьох місяців допомагає підтримувати актуальність інформації, що обробляється чат-ботом. Одним із способів реалізації цього підходу є перезавантаження веб-сервера або виконання вручного оновлення файлів у директорії проекту (рис. 3.38).

Варто також вести облік дати останнього оновлення, щоб мати можливість своєчасно виконати оновлення та уникнути втрати даних або перерви в роботі вашого чат-бота.

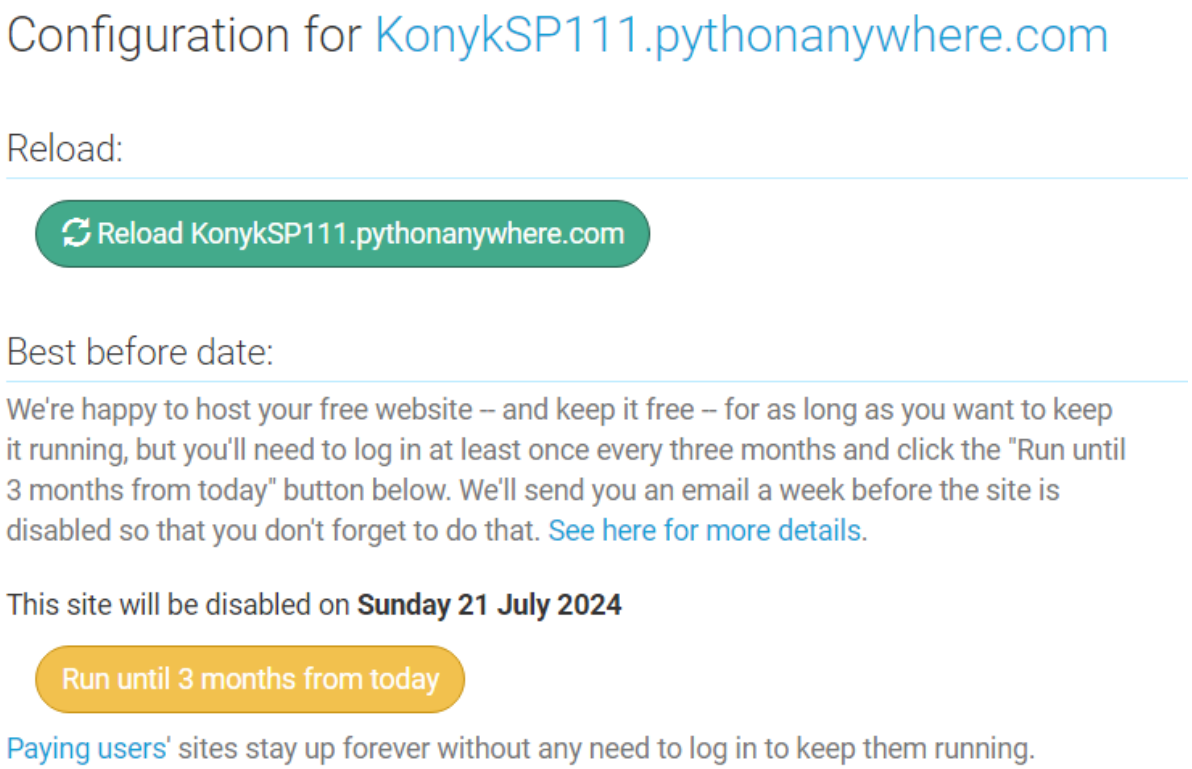


Рисунок 3.38 – Інформацію про хостинг телеграм-чат бота

3.2 Тестування та огляд роботи програмного забезпечення

Тестування програмного додатку відбувалося в ході розробки з метою виявлення та виправлення помилок, які систематично виникали. Основний акцент був зроблений на огляді інтерфейсу, зв’язку з базою даних, перевірці працездатності чат-бота, а також визначенні швидкості реакції на запити користувачів та правильності функціонування алгоритму.

Процес тестування включав:

1. Перевірка зручності та доступності інтерфейсу для користувача, впевненість у правильному відображенні елементів та їх взаємодії.
2. Виконання різних команд та взаємодія з чат-ботом для перевірки його функціональності та відповідності очікуваним результатам.
3. Вимірювання часу відправки та отримання відповіді від бота, щоб забезпечити швидку та безперебійну роботу.
4. Коректне введення інформації в базу даних SQLite.

Цей процес дозволив виявити та виправити помилки, а також підтвердити коректну роботу програмного додатку перед його публікацією.

Результат підтвердив успішність програми: головне меню з'явилося на екрані, як очікувалося. Хоча відповідь затрималася на декілька секунд, ймовірно, через обсяг коду, але це не вплинуло на коректність відображення. Все це свідчить про те, що бот працює так, як заплановано, і готовий до подальшої взаємодії. Це обнадійливий знак, що реалізація функціоналу кнопок працює правильно, а відповіді відповідають очікуванням користувача. Тепер можна продовжувати перевіряти функціональну частину програми на основі успішного тестування (рис. 3.39).

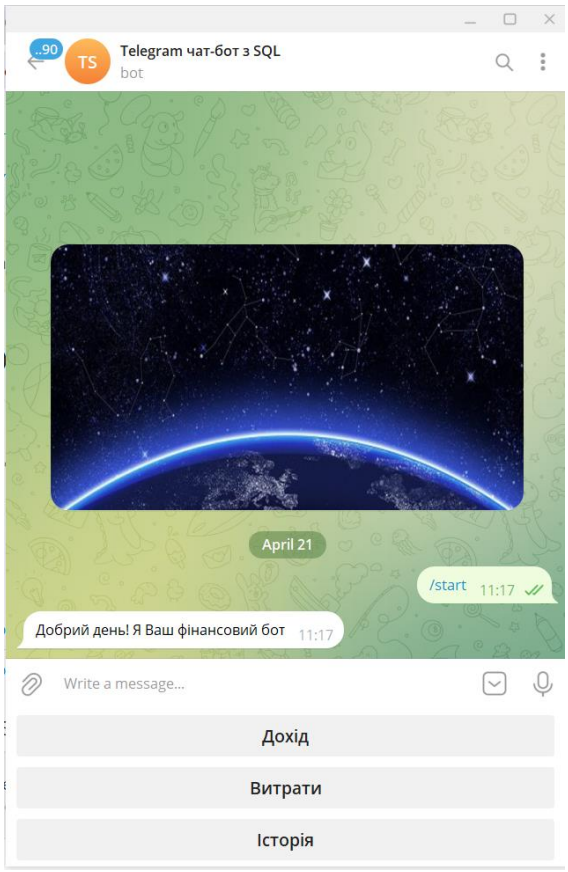


Рисунок 3.39 – Головне меню чат-бота

Головне меню було реалізоване у вигляді текстового блоку інформації, а також кілька кнопок для подальшої роботи з додатком. Це забезпечило зручний інтерфейс для користувачів і дозволило їм легко працювати з базою даних.

Аналізуючи час відправки повідомлень, можна побачити, що бот відповів на запит користувача в ту саму хвилину. Оскільки не спостерігається значної завантаженості користувачів, програмі вдається оперативно надавати результати. Проте, при збільшенні кількості запитів, час очікування обробки може збільшитися. У такому випадку, може знадобитися розгляд альтернативних рішень для оптимізації часу реакції бота на запити користувачів.

Наступний етап полягає в оцінці правильності функціонування кнопок та їх взаємодії з базою даних. Це важливий крок для забезпечення стабільності та надійності програми. Під час цього етапу було проведено перевірку наступних аспектів:

1. Перевіряється, чи виконують кнопки ті дії, які передбачені в програмі.

Наприклад, якщо кнопка "Додати" призначена для додавання даних у базу, вона дійсно має виконувати цю дію.

2. Впевнюємося, що дані, які вводяться через кнопки, правильно зберігаються у базі даних. Також перевіряємо, чи можна отримати дані з бази за допомогою відповідних кнопок.
3. Перевіряється, як програма реагує на непередбачувані ситуації, наприклад, якщо відбувається конфлікт при внесенні даних.
4. Важливо переконатися, що взаємодія з базою даних через кнопки відбувається ефективно, без зайвого затримання.

Перевіримо нашу внесену інформацію в базі даних від користувача з прикладу (рис. 3.40). Якщо база даних функціонує коректно то кнопка Історія повинна витянути нашу інформацію (рис. 3.41).

Рисунок 3.40 – Функціонування кнопок та запис інформації в базу даних

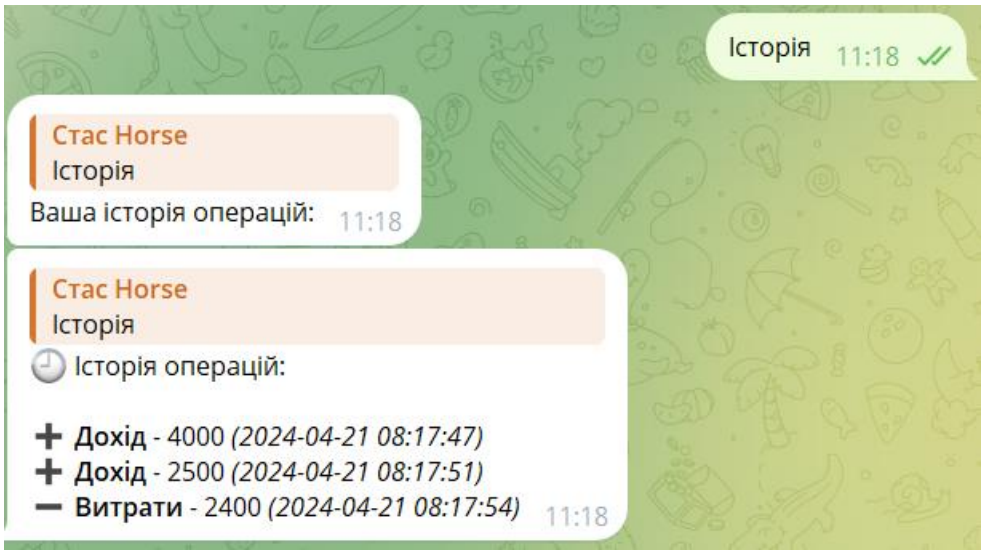


Рисунок 3.41 – Перевірка кнопки Історія та її конект з базою даних

Зберігання даних, що вводяться через інтерфейс користувача, у базі даних є ключовою складовою для забезпечення цілісності та надійності інформації. Перевірка успішності цього процесу може бути визначена через кілька критеріїв. Перш за все, необхідно переконатися, що дані правильно відображаються в базі даних і відповідають введеним користувачем. Далі, слід переконатися, що дані зберігаються вірно та без втрати інформації. Крім того, важливо перевірити, чи виконані всі необхідні правила та обмеження бази даних, такі як унікальність ключів або відповідність типам даних. Якщо всі ці критерії виконані, можна стверджувати, що мета забезпечення цілісності та надійності даних виконана. Таким чином, можна зробити висновок, що перевірка пройдена успішно, оскільки дані зберігаються належним чином та відповідають вимогам системи.

ВИСНОВКИ

Проведено аналіз чат ботів, виявлено роль яку відіграють боти в сучасному світі. Їх використання, поширення як серед звичайних користувачів для повсякденних потреб, так і серед бізнесу для автоматизації процесів та оптимізації ресурсів.

Досліджено соціальні мережі та мобільні додатки з інструментами для інтеграції ботів для вибору оптимального варіанту для розробки конкретного чат-бота.

Реалізація проекту відбулася з використанням мови програмування Python та інтегрованого середовища розробки PyCharm. Для забезпечення зв'язку з месенджером Telegram використано пакети Aiogram, Sqlite та інтерфейс зв’язку з додатком Telegram Bot API. Бот був офіційно зареєстрований у Telegram з унікальним ідентифікатором пошуку @TgSQL_bot.

У процесі розробки Telegram чат-бота на мові програмування Python було віддано перевагу використанню бази даних SQLite. База даних SQLite інтегрована безпосередньо у програмний код бота, що дозволило забезпечити простоту встановлення та використання. Розроблена БД відрізняється легкістю використання та налаштування. Вбудованість бази даних дозволяє легко інтегрувати БД без необхідності запуску окремого сервера, що робить процес розгортання бота швидшим і простішим. База даних ефективно зберігає та управляє різноманітною інформацією, яка використовується у функціонуванні бота. Включає в себе дані користувачів, історію взаємодії з ботом, статистику використання та інші важливі дані, які допомагають оптимізувати роботу чат-бота та покращувати користувацький досвід.

Проведено тестування як у процесі розробки, так і після завершення, включаючи ручну перевірку функціонування, швидкості відповідей програми для перевірки коректності роботи бота.

У результаті вдалої реалізації проекту були досягнуті головні цілі та очікування. Чат-бот став ефективним інструментом для контроля фінансових

витрат через месенджер Telegram, забезпечивши користувачам зручний доступ до необхідної інформації та можливість взаємодії з ним у режимі реального часу.

ПЕРЕЛІК ПОСИЛАНЬ

1. Що таке чат-бот: секрети використання та основні переваги для бізнесу. [Електронний ресурс] - https://helpcrunch.com/blog/uk/shcho-take-chat-bot/

2. Для 50,6% читачів основним месенджером є Telegram. Результати опитування AIN.UA [Електронний ресурс] - https://ain.ua/2023/03/09/telegram-osnovnyj-mesendzher-opytuvannya/

3. Чат-бот. Переваги, засоби використання та як створити бота. [Електронний ресурс] - https://gerabot.com/article/detalno_pro_chatboti

4. Кращі IDE для Python в 2023 році. [Електронний ресурс] - https://mate.academy/blog/python/ide-for-python-2023/

5. Створюємо Telegram бота на Python. Частина 1 [Електронний ресурс] - https://devzone.org.ua/post/stvoriuyemo-telegram-bota-na-python-chastyna-1

6. SQL база даних Для чого призначена база даних? [Електронний ресурс] - https://www.ukraine.com.ua/uk/blog/programming/sql-baza-dannih-dlya-chego-prednaznachena-baza-dannih.html

7. За границей Hello World: полный гайд по разработке Telegram ботов с помощью Python и Aiogram 3. [Електронний ресурс] - https://habr.com/ru/articles/732136/

8. TOP 10 PYCHARM SHORTCUTS FOR PYTHON LEARNERS. [Електронний ресурс] - https://www.mariakhalusova.com/posts/2017-07-26-top10-pycharm-shortcuts/

9. Менеджер баз данных SQLiteStudio. [Електронний ресурс] - https://progtips.ru/bazy-dannyx/menedzher-baz-dannyx-sqlitestudio.html

10. Встановлення модулів Python¶. [Електронний ресурс] - https://docs.python.org/uk/3/installing/index.html

11. Що таке pip в Python? [Електронний ресурс] - https://itproger.com/ua/spravka/python/pip

12. Асинхронный Telegram бот на языке Python 3 с использованием библиотеки Aiogram. Урок 5. Клавиатуры и кнопки [Електронний ресурс] - https://surik00.gitbooks.io/aiogram-lessons/content/chapter5.html

13. Python in the Cloud: Let's Explore PythonAnywhere and Other Alternatives [Електронний ресурс] - https://www.codemotion.com/magazine/languages/python-in-the-cloud-lets-pythonanywhere-and-other-alternatives/

14. BotFather. Можливості, команди та функціонал. [Електронний ресурс] - https://gerabot.com/article/botfather_mozhlivosti_ta_funkcional

15. Будуємо телеграм чат-бот на Java: від ідеї до деплою. Частина 1. [Електронний ресурс] - https://dou.ua/forums/topic/38358/

16. Using Python on PythonAnywhere. [Електронний ресурс] - https://www.py4e.com/software-pyaw.php

17. Що таке API: навіщо використовується програмістами та базові основи роботи з ним. [Електронний ресурс] - https://cloud.itstep.org/blog_3/what-is-an-api-why-is-it-used-by-programmers-and-the-basics-of-working-with-it

18. #4 – SQLite3. Подключение к базе данных [Електронний ресурс] - https://itproger.com/ua/course/telegram-bot/4

19. Справочник по Bot API [Електронний ресурс] - https://tlgrm.ru/docs/bots/api

20. МЕСЕНДЖЕРИ ДОВІРИ [Електронний ресурс] - https://reputation.contractors/mediatrust2023

21. Как установить аватар бота в Telegram [Електронний ресурс] - https://chat2desk.com/baza-znanij/nastrojka-messendzherov/kak-ustanovit-avatar-bota-v-telegram

ДОДАТОК А. ФРАГМЕНТИ ПРОГРАМНОГО КОДУ

bot.py

```
 from aiogram import executor
from dispatcher import dp
import handlers

from db import BotDB
BotDB = BotDB('DataBase.db')

if __name__ == "__main__":
    executor.start_polling(dp, skip_updates=True)
```

db.py

```
import sqlite3

class BotDB:

    def __init__(self, db_file):
        self.conn = sqlite3.connect(db_file)
        self.cursor = self.conn.cursor()

    def user_exists(self, user_id):
        """Перевірка чи є користувач в базі"""
        result = self.cursor.execute("SELECT `id` FROM `users` WHERE `user_id` = ?",
(user_id,))
        return bool(len(result.fetchall())
```

```
    def get_user_id(self, user_id):
        """Достаєм id користувача в базі по його user_id"""
        result = self.cursor.execute("SELECT `id` FROM `users` WHERE `user_id` = ?",
(user_id,))
        return result.fetchone()[0]

    def add_user(self, user_id):
        """Добавляємо користувача в базу"""
        self.cursor.execute("INSERT INTO `users` (`user_id`) VALUES (?)", (user_id,))
        return self.conn.commit()

    def add_record(self, user_id, operation, value):
        """Створення запису про дохід/витрати"""
        self.cursor.execute("INSERT INTO `records` (`users_id`, `operation`, `value`)
VALUES (?, ?, ?)",
            (self.get_user_id(user_id),
            operation == "+",
            value))
        return self.conn.commit()

    def get_records(self, user_id, within = "all"):
        """Отримуємо історию про дохід/витрати"""

        if within == "day":
            result = self.cursor.execute("SELECT * FROM `records` WHERE `users_id` =
? AND `date` BETWEEN datetime('now', 'start of day') AND datetime('now',
'localtime') ORDER BY `date`",
                (self.get_user_id(user_id),))
        else
```

```
            result = self.cursor.execute("SELECT * FROM `records` WHERE `users_id` =
? ORDER BY `date`",
                (self.get_user_id(user_id),))

        return result.fetchall()

    def close(self):
        """Закриваєм звязок з БД"""
        self.connection.close()
```

dispatcher.py

```
import logging
from aiogram import Bot, Dispatcher
from filters import IsOwnerFilter, IsAdminFilter, MemberCanRestrictFilter
import config

# Configure logging
logging.basicConfig(level=logging.INFO)

# prerequisites
if not config.BOT_TOKEN:
    exit("No token provided")

# init
bot = Bot(token=config.BOT_TOKEN, parse_mode="HTML")
dp = Dispatcher(bot)

# activate filters
```

```
dp.filters_factory.bind(IsOwnerFilter)
dp.filters_factory.bind(IsAdminFilter)
dp.filters_factory.bind(MemberCanRestrictFilter)
```

config.py

```
BOT_TOKEN="7068892783:AAE4PghbvLX-GqOGXfP8VOZ1ibhEvTvJaPg"
```

personal_actions.py

```
from aiogram import types
from dispatcher import dp
import config
import re
from bot import BotDB

@dp.message_handler(commands="start")
async def start(message: types.Message):
    if not BotDB.user_exists(message.from_user.id):
        BotDB.add_user(message.from_user.id)

    keyboard = types.ReplyKeyboardMarkup(resize_keyboard=True)
    keyboard.add(types.KeyboardButton("Дохід"))
    keyboard.add(types.KeyboardButton("Витрати"))
    keyboard.add(types.KeyboardButton("Історія"))

    await message.bot.send_message(message.from_user.id, "Добрий день! Я Ваш фінансовий бот", reply_markup=keyboar
```

```
@dp.message_handler(lambda message: message.text in ("Дохід", "Витрати"))
async def operation_handler(message: types.Message):
    operation_type = "+" if message.text == "Дохід" else "-"
    await message.reply(f"Введіть суму {message.text.lower()}:")

@dp.message_handler(lambda message: message.text == "Історія")
async def history_handler(message: types.Message):
    await message.reply("Ваша історія операцій:")
    # Отримуємо історію операцій з бази даних
    records = BotDB.get_records(message.from_user.id, "day")

    if len(records) > 0:
        # Формуємо повідомлення з історією операцій
        answer = "🕘 Історія операцій:\n\n"
        for record in records:
            answer += "<b>" + ("➖ Витрати" if not record[2] else "➕ Дохід") + "</b>"
            answer += f" - {record[3]}"
            answer += f" <i>({record[4]})</i>\n"
        await message.reply(answer)
    else:
        await message.reply("Записів не виявлено!")

@dp.message_handler(lambda message: re.match(r"^[+-]?\d{1,3}(?:[,
]?\d{3})*(?:\.\d+)?$", message.text))
async def amount_handler(message: types.Message):
    if message.text.startswith("+"):
        operation_type = "+"
        operation_text = "доходу"
    elif message.text.startswith("-"):
```

```
        operation_type = "-"
        operation_text = "витрати"
    else:
        # Якщо користувач ввів неправильний формат
        await message.reply("Неправильний формат. Введіть суму ще раз.")
        return

    value = float(re.sub(r"[^\d.]", "", message.text))  # Видаляємо всі символи, крім цифр і крапки
    BotDB.add_record(message.from_user.id, operation_type, value)

    if operation_type == "-":
        await message.reply("✅ Запис про витрати успішно внесено!")
    else:
        await message.reply("✅ Запис про дохід успішно внесено!")
```

init.py

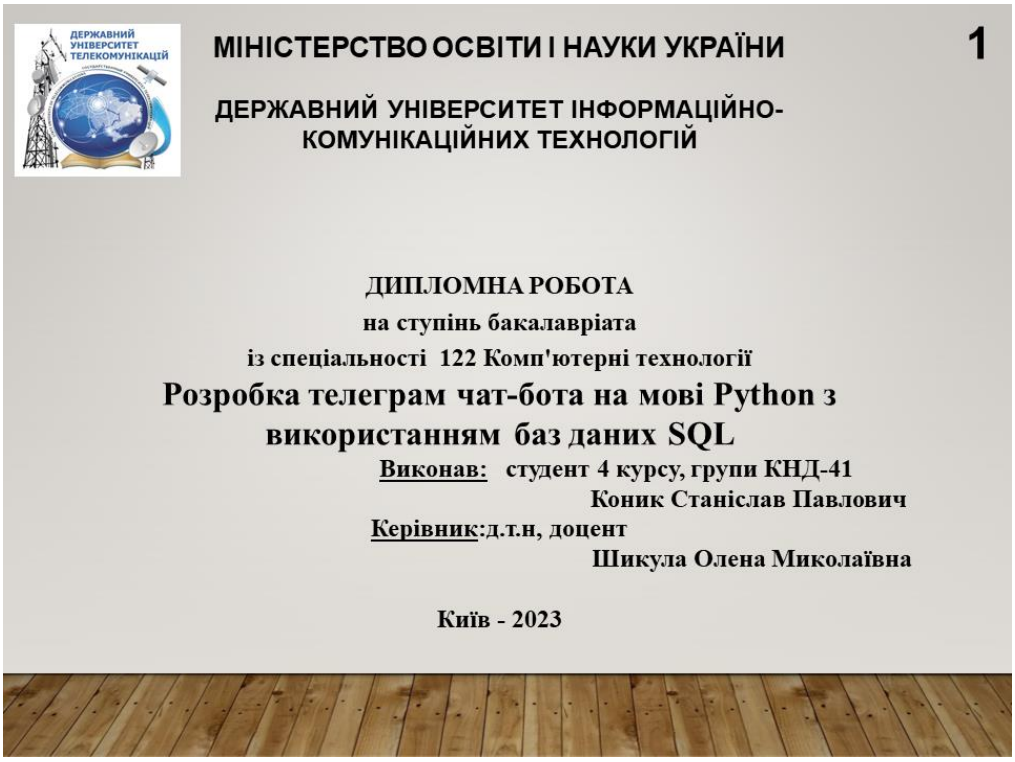
```
from . import personal_actions
```

filters.py

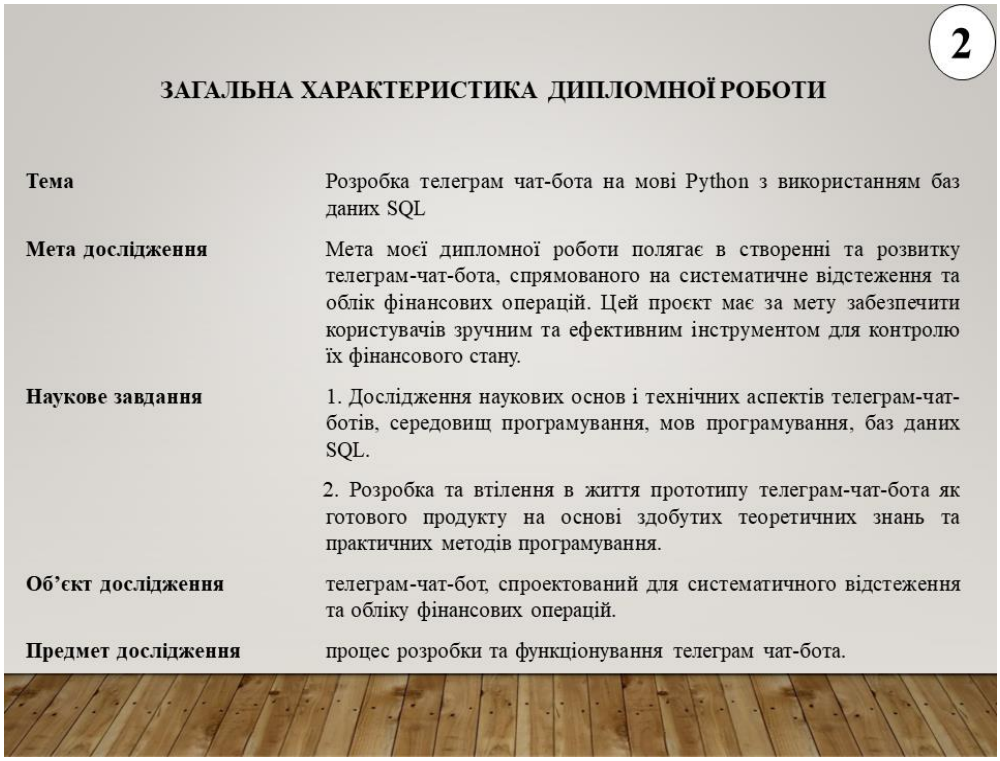
```
import config

class IsOwnerFilter(BoundFilter):

    key = "is_owner"

    def __init__(self, is_owner):
```

```
        self.is_owner = is_owner

    async def check(self, message: types.Message):
        return message.from_user.id == config.BOT_OWNER

class IsAdminFilter(BoundFilter):

    key = "is_admin"

    def __init__(self, is_admin: bool):
        self.is_admin = is_admin

    async def check(self, message: types.Message):
        member = await message.bot.get_chat_member(message.chat.id,
message.from_user.id)
        return member.is_chat_admin() == self.is_admin

class MemberCanRestrictFilter(BoundFilter):
    key = 'member_can_restrict'
    def __init__(self, member_can_restrict: bool):
        self.member_can_restrict = member_can_restrict

    async def check(self, message: types.Message):
        member = await message.bot.get_chat_member(message.chat.id,
message.from_user.id)
        return (member.is_chat_creator() or member.can_restrict_members) ==
self.member_can_restrict
```

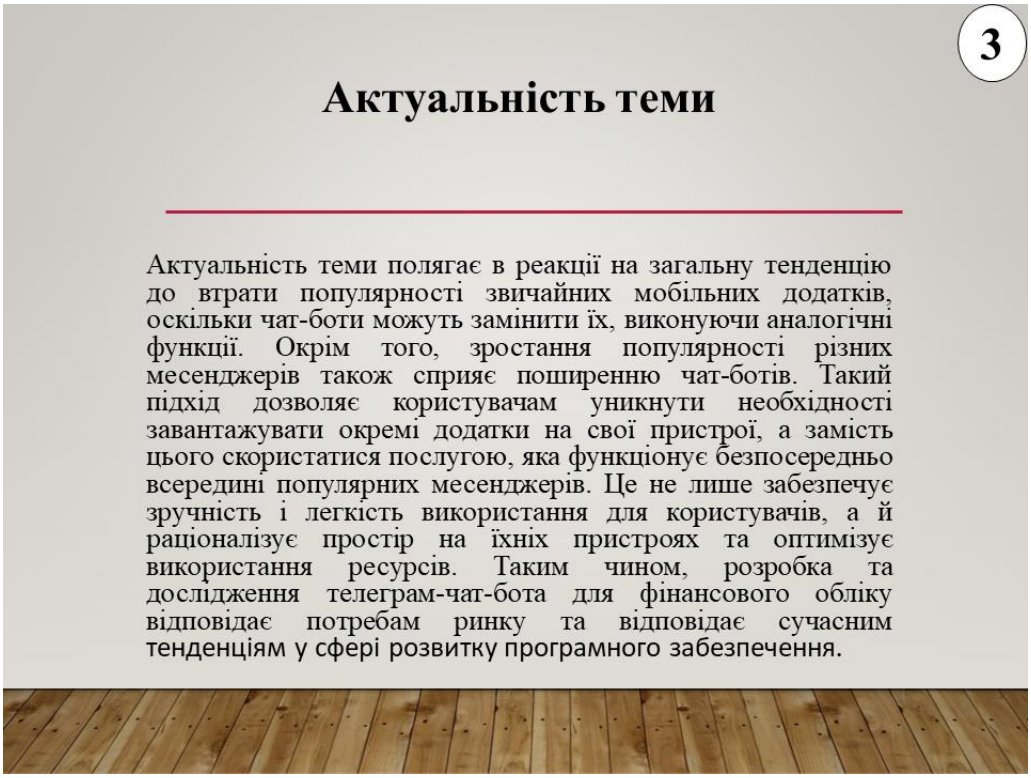
ДОДАТОК Б ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація)



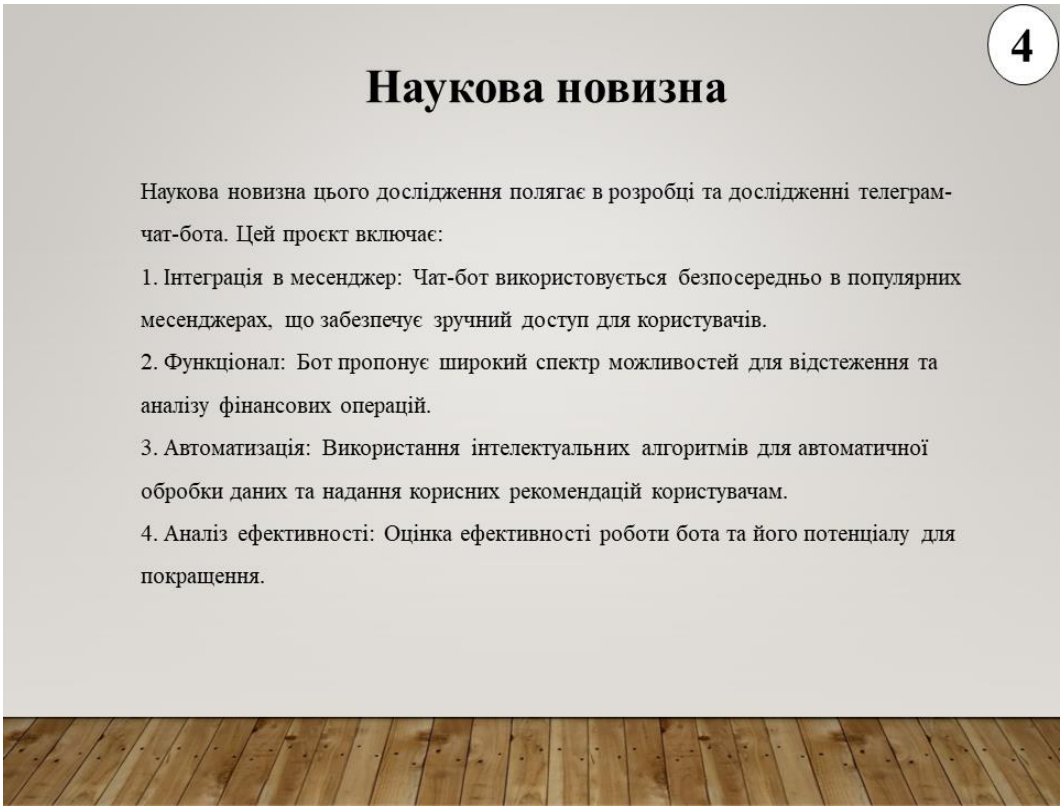
Слайд 1



Слайд



Слайд 3



Слайд 4

5

Теоретичні основи

У цьому проекті ми використовуємо мову програмування Python як основну мову для розробки телеграм-чат-бота. Python відомий своєю простотою та ефективністю, що робить його ідеальним вибором для створення таких проєктів.

Для збереження та керування даними ми використовуємо базу даних SQLite. SQLite - це легка, вбудована база даних, яка працює відмінно для малих до середніх розмірів додатків. Вона ідеально підходить для наших потреб зі зберігання та отримання даних у нашому телеграм-чат-боті.

Слайд 5

6

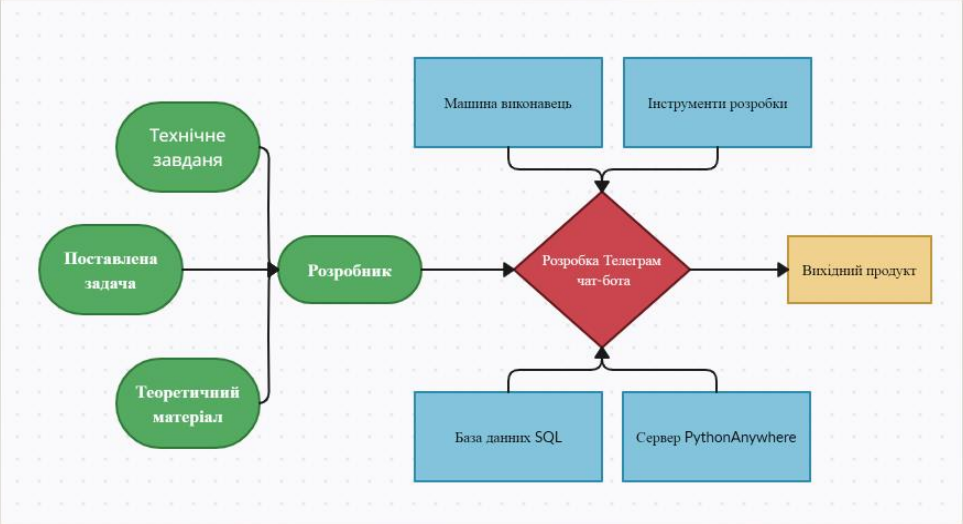
Архітектура та функціонал чат-бота

Архітектура розробленого телеграм-чат-бота базується на моделі "клієнт-сервер".
Основні складові:
•**Користувач**: Взаємодія з ботом через інтерфейс месенджера Telegram.
•**Сервер (бот)**: Обробка запитів користувачів та відправлення відповідей.
Функціонал:
1.Отримання та обробка фінансових даних.
2.Відображення статистики доходів та витрат.
3.Аналіз та генерація звітів за заданими періодами.
Цей функціонал забезпечує користувачам зручний та ефективний інструмент для керування їхніми фінансами безпосередньо через месенджер Telegram.

Слайд 6

7

Архітектура та функціонал чат-бота



Слайд 7

8

Тестування та валідація

1. Функціональне тестування: Перевірка роботи основного функціоналу бота, включаючи відстеження доходів та витрат, аналіз статистики, та нагадування про платежі.
2. Валідація даних: Перевірка коректності та цілісності даних у базі даних SQLite.
3. Тестування взаємодії з користувачем: Перевірка реакції бота на різні введені команди та сценарії користувача.
4. Тестування відповідності вимогам: Перевірка, чи відповідає функціонал бота встановленим вимогам та очікуванням користувачів.
5. Тестування забезпечення безпеки: Перевірка наявності заходів забезпечення конфіденційності та цілісності даних користувачів.
6. Тестування завантаження: Перевірка реакції бота на велику кількість запитів та обробка даних у межах прийнятних часових рамок.

Проведення цих видів тестування дозволило підтвердити функціональність, надійність та безпеку розробленого чат-бота.

Слайд 8

9

Процес розробки

1. Аналіз вимог: Оцінка потреб користувачів та визначення функціональних вимог до чат-бота.
2. Проектування: Створення структури бази даних, розроблення архітектури бота та визначення основних функцій.
3. Реалізація: Написання коду на мові Python для логіки бота та взаємодії з базою даних SQLite.
4. Інтеграція з Telegram API: Підключення бота до Telegram API для взаємодії з користувачами.
5. Тестування: Перевірка працездатності та відповідності функціоналу вимогам.
6. Налаштування та оптимізація: Вдосконалення швидкодії та стабільності роботи, виправлення помилок.

Цей процес забезпечив успішне створення та реалізацію функціоналу телеграм-чат-бота для ефективного управління фінансами користувачів.

Слайд 9

10

Перспективи

- Можливість розширення функціоналу бота для включення додаткових корисних опцій та можливостей.
- Подальше удосконалення і оптимізація алгоритмів для підвищення швидкодії та продуктивності бота.
- Планування інтеграції з іншими сервісами та програмами для розширення функціоналу та зручності використання.

Слайд 10

11

ВИСНОВКИ

- Проведено аналіз чат ботів, виявлено роль яку відіграють боти в сучасному світі. Їх використання, поширення як серед звичайних користувачів для повсякденних потреб, так і серед бізнесу для автоматизації процесів та оптимізації ресурсів.
- Досліджено соціальні мережі та мобільні додатки з інструментами для інтеграції ботів для вибору оптимального варіанту для розробки конкретного чат-бота.
- Реалізація проекту відбулася з використанням мови програмування Python та інтегрованого середовища розробки PyCharm. Для забезпечення зв'язку з месенджером Telegram використано пакети Aiogram, Sqlite та інтерфейс зв’язку з додатком Telegram Bot API.

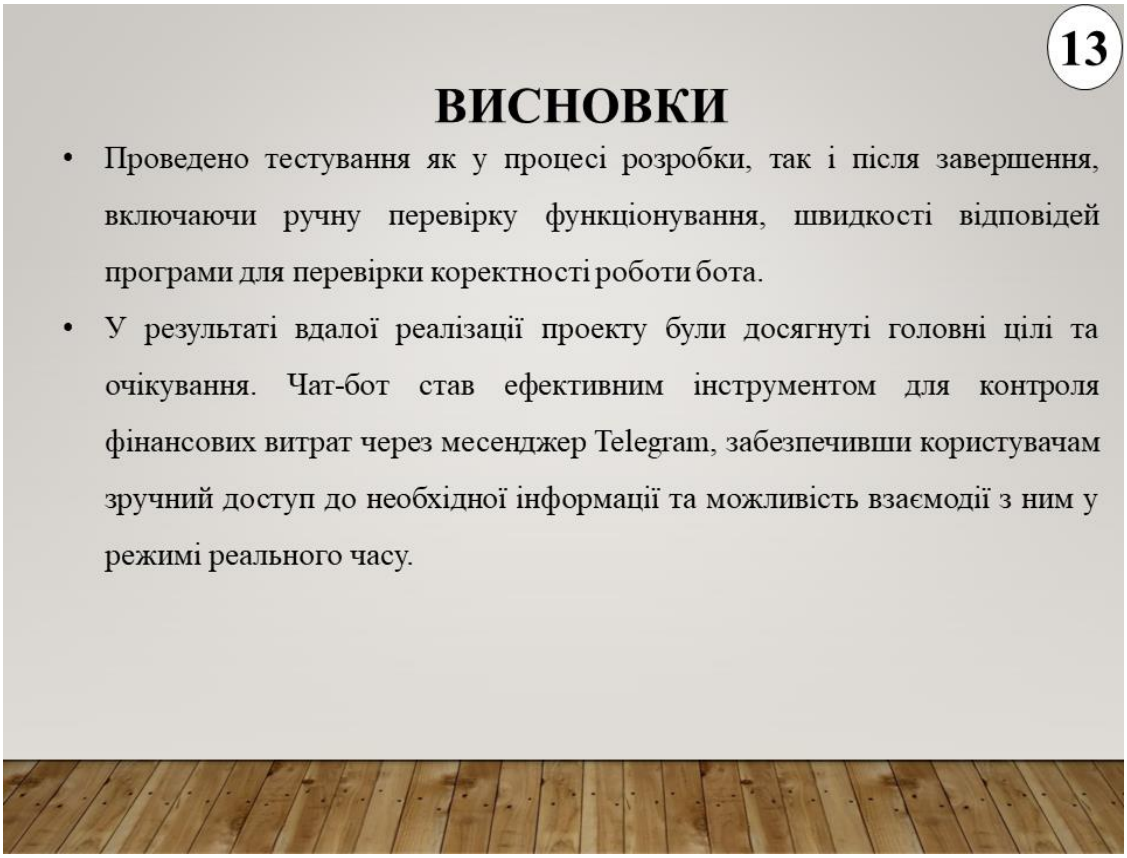
Слайд 11

12

ВИСНОВКИ

- Розроблена БД відрізняється легкістю використання та налаштування. Вбудованість бази даних дозволяє легко інтегрувати БД без необхідності запуску окремого сервера, що робить процес розгортання бота швидшим і простішим. База даних ефективно зберігає та управляє різноманітною інформацією, яка використовується у функціонуванні бота. Включає в себе дані користувачів, історію взаємодії з ботом, статистику використання та інші важливі дані, які допомагають оптимізувати роботу чат-бота та покращувати користувацький досвід.
- У процесі розробки Telegram чат-бота на мові програмування Python було віддано перевагу використанню бази д\аних SQLite. База даних SQLite інтегрована безпосередньо у програмний код бота.

Слайд 12


13
ВИСНОВКИ
• Проведено тестування як у процесі розробки, так і після завершення, включаючи ручну перевірку функціонування, швидкості відповідей програми для перевірки коректності роботи бота.
• У результаті вдалої реалізації проекту були досягнуті головні цілі та очікування. Чат-бот став ефективним інструментом для контроля фінансових витрат через месенджер Telegram, забезпечивши користувачам зручний доступ до необхідної інформації та можливість взаємодії з ним у режимі реального часу.

Слайд 13