МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет

Центр заочної, дистанційної та вечірньої форм навчання

Кафедра комп'ютених наук

«До захисту допущено»

Завідувач кафедри

_____ Ігор ШЕЛЕХОВ

(підпис)

___ _____ 20__р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 – Комп'ютерні науки,

Освітньо-професійної програми «Інформатика»

на тему: «Інформаційна система вибору та вивчення мови програмування із застосуванням чат-бота»

здобувачки групи ІНз-01с Шапар Анастасії Юріївни

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

_____ Анастасія ШАПАР

(підпис)

Керівник

кандидат фізико-математичних

наук, доцент

_____ Сергій ШАПОВАЛОВ

(підпис)

Суми – 2024

Сумський державний університет

Центр заочної, дистанційної та вечірньої форми навчання

Кафедра комп'ютерних наук

«Затверджую»

В.о. завідувача кафедри

_____ Ігор ШЕЛЕХОВ

(підпис)

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавра

зі спеціальності 122 - Комп'ютернi науки, освітньо-професійної програми «Інформатика» здобувачки групи ІНз-01с Шапар Анастасії Юріївни

1. Тема роботи: ««Інформаційна система вибору та вивчення мови програмування із застосуванням чат-бота» затверджую наказом по СумДУ від *«26» квітня 2024 р. № 0438-VI*
2. Термін здачі здобувачем кваліфікаційної роботи *до «05» червня 2024 року*
3. Вхідні дані до кваліфікаційної роботи __
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)

1) Аналіз видів і актуальність чат-ботів, порівняння їх можливостей 2) Проектування чат-бота та опис обраних програмних засобів 3) Налаштування інструментів для розробки чат-бота 4) Програмна реалізація застосунку чат-бота.

5. Перелік графічного матеріалу (із точним зазначенням обов'язкових креслень)
6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «___»_____20__р.

Завдання прийняв до виконання _____

(підпис)

Керівник _____

(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Аналіз видів та актуальність чат-ботів, порівняння їх можливостей*		
2	*Проектування чат-бота та опис обраних програмних засобів*		

3	*Налаштування інструментів для розробки чат-бота*		
4	*Програмна реалізація застосунку чат-бота*		
5	*Оформлення пояснювальної записки до кваліфікаційної роботи*		

Здобувачка вищої освіти _____ Керівник _____
(підпис) (підпис)

АНОТАЦІЯ

Записка: 60 стор., 42 рис., 2 табл., 1 додаток, 20 використаних джерел.

Обгрунтування актуальності теми роботи – тема чат-ботів досить актуальна, бо на сьогоднішній день популярність мобільних пристроїв висока, а основне спілкування за їх допомогою проходить у месенджерах. Даний чат-бот допоможе програмісту-початківцю вивчати мови програмування у будь який час і у будь-якому місці.

Об'єкт дослідження – процес навчання мовам програмування.

Мета роботи – створення чат-бота для вивчення мов програмування.

Методи дослідження – технології та алгоритми створення чат-ботів

Результат – розроблено чат-бот для вивчення мов програмування. Складено планування майбутніх оновлень нашого бота, а також покращення його взаємодії з користувачами.

ЧАТ-БОТ, ВИВЧЕННЯ МОВ ПРОГРАМУВАННЯ, ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ, TELEGRAM, PYTHON,TELEGRAM BOT API, SQLITE

ЗМІСТ

ВСТУП

На сьогоднішній день технології швидко розвиваються, тому важко уявити сучасну людину без смартфона. У невеликому гаджеті зосереджена різноманітна цікава і важлива інформація. Частіше за все ми користуємося телефоном за для перегляду соціальних мереж. В них ми спілкуємося з друзями та знайомими. Чат-бот – це робот, який здатний автоматично і швидко відповідати на повідомлення, які користувач вводить в чаті. Чат-боти мають багато можливостей, а саме: надання необхідної інформації, створення заміток і виконання решти повсякденних завдань. Ці боти в основному розміщуються в таких месенджерах як: Facebook Messenger, Viber, Telegram та інші. Завдяки багатофункціональності на чат-боти є попит у різних компаній по всьому світу. Це є вигідно, як для користувача, (він може отримати потрібну йому інформацію або виконувати інші дії більш зручним способом), так і для компанії (вона за допомогою чат-бота може розвивати бізнес, додавати нових клієнтів і просувати свій бренд).

Платформи для чат-ботів відкривають безмежні можливості для створення функціоналу, який раніше був доступний лише в мобільному додатку чи на веб-сайтах. Вони навіть можуть обслуговувати замовлення продукції, резервувати місця та навіть проводити банківські транзакції.

У епоху, коли інформаційні технології розвиваються з неймовірною швидкістю, попит на кваліфікованих розробників невпинно зростає. Разом із цим вдосконалюються інструменти та мови програмування. Це робить сферу IT однією з найвищими заробітними платами. Ось чому якісні IT-продукти оцінюються так високо, а оплата праці може досягати вражаючих цифр, приваблюючи все більше молодих талантів до професії розробника.

Існує велика кількість різноманітних вебінарів, курсів, занять, які можуть допомогти розвиватися в цій сфері. Однак через те що інформації занадто багато, школяру чи студенту важко обрати правильний вектор

розвитку і усвідомити те, що він має опанувати для досягнення бажаних результатів.

Telegram – це додаток, створений з використанням мови програмування C++, що дозволяє обмінюватися текстовими повідомленнями та файлами в широкому спектрі форматів. Його сервери використовують приватний код, забезпечуючи таким чином приватність та зашифрованість даних. Для шифрування даних використовуються декілька методів. Аутентифікація здійснюється за допомогою алгоритму RSA-2048, а для шифрування повідомлень застосовується AES з секретним ключем, доступним тільки серверу та користувачу.

Об'єкт дослідження – процес навчання мовам програмування.

Предмет дослідження – інформаційна система вибору та вивчення мови програмування із застосуванням чат-бота.

Гіпотеза дослідження полягає в тому, що розроблений чат-бот буде здатний структуровано подавати теоретичні відомості про мови програмування та закріпляти отримані користувачем знання за допомогою тестів.

Наукова новизна. Досліджено проблему навчання мовам програмування. Під час дослідження було створено чат-бот, який здатний навчати мовам програмування.

Структура. Кваліфікаційна бакалаврська робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел та додатку, викладених на 59 сторінках.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ВІДОМОСТІ ПРЕДМЕТНОЇ ОБЛАСТІ. АКТУАЛЬНІСТЬ БОТІВ ТА МОЖЛИВОСТЕЙ ЇХ РЕАЛІЗАЦІЇ

1.1 Формування мети проекту

Ще двадцять років тому монітори займали половину робочого місця користувача, а саме поняття “комп’ютер” викликало нерозуміння і сумніви щодо перспектив його популярності і необхідності в майбутньому. У ті часи кожна п’ята людина мріяла стати інженером, а зараз – спеціалістом в ІТ сфері. Які цьому причини перелічимо нижче.

- Висока заробітна плата: «Середня заробітна плата розробника програмного забезпечення складає від 20 тисяч гривень» і саме тому більшість українського населення вважають таку вакансію мрією. Для людини, яка працює у цій сфері – це звичайна пропозиція, зі звичайною зарплатою.
- Можливість кар’єрного розвитку: почавши працювати в сфері інформаційних технологій, можна отримати не тільки практичні навички, а й дорости через декілька років до ментора, або навіть розробити щось своє і стати новим Стівом Джобсом.
- Дистанційна робота: програмісту для його роботи потрібний лише ноутбук і доступ до інтернету. Він може працювати як в офісі, так і з будь-якої точки світу в зручний для нього час.

Здається, що в ІТ сферу просто потрапити. Для цього потрібно всього лише вивчити мови програмування. Але перша проблема з якою стикаються ті, хто хоче почати вивчати програмування – це елементарне незнання із чого почати. Нинішні працівники сфери ІТ також можуть слабо орієнтуватися у напрямках програмування і їх областях застосування.

Тому завданням дипломного проекту стало створення боту для надання користувачем конкретної, зрозумілої інформації і можливості

ознайомитися з синтаксисом та аспектами мов програмування, отримати чітке бачення подальшого розвитку у даній області.

1.2 Актуальність ботів

Для початку потрібно розібратися, що ж таке чат-бот? Він являє собою комп'ютерну програму, розроблену на основі машинного навчання, яка автоматизує спілкування з користувачем. Вона використовується від створення простого exo-бота до виконання різноманітних завдань. За останні роки боти почали набувати нечувану популярність. Це пов’язано із карантином через COVID-19. Із самого початку карантину багато організацій і компаній зіткнулися із проблемою дистанційного надання послуг та вирішення запитів клієнтів. Як виявилося, простіше за все це було зробити за допомогою чат-ботів. На даний час майже всі великі компанії використовують ботів для вирішення своїх проблем і покращення роботи з користувачами. Відповідно до даних sendpulse із січня до серпня 2022 року, щомісяця активність Telegram ботів зростає, що позитивно впливає на бізнес і покращує його дохідність.

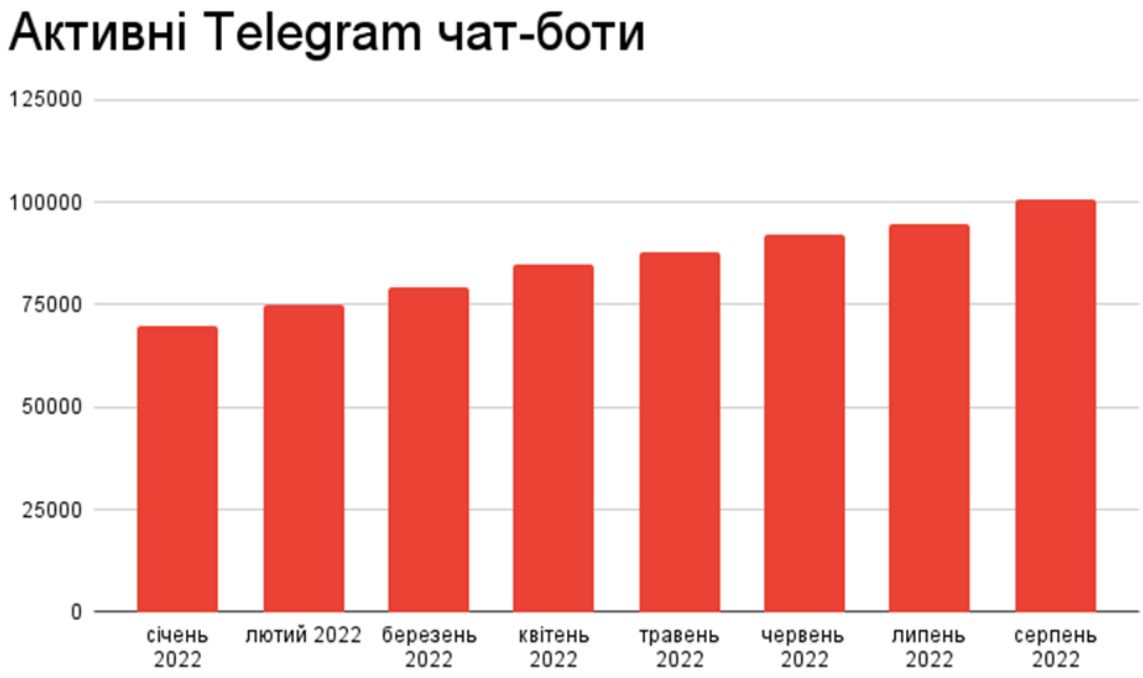

Рисунок 1.1 – Прибуток чат-ботів з січня по серпень 2022 року

1.3 Порівняльний аналіз платформ для інтеграції бота

У сучасному світі важко уявити людину, яка не користується гаджетами. Згідно статистичних даних із 2021 по 2022 рік в середньому людина проводить у соціальних мережах та месенджерах від 3 годин на день. Виходячи з таких даних, ми можемо зробити висновок, що вони є доцільними для реалізації даного проекту. Тепер потрібно підібрати месенджер, який оптимально буде вирішувати поставлені нами задачі. Для цього, склавши таблицю, проведемо аналіз найпопулярніших додатків.

Таблиця 1.1 – Порівняння мобільних додатків станом на 2021 рік

Функціонал	**WhatsApp**	**Viber**	**F b Messenger**	**Telegram**
Хмарне сховище	+	-	+	+
Охоплення 50+% населення України	-	+	+	+
Відкритий API	-	+	+	+
Двостороння відмова	-	-	-	+
Шифрування, анонімність	-	-	-	+
Браузер у додатку	-	-	-	+
Версія для комп’ютера	-	-	+	+

Провівши аналіз та дослідивши таблицю, можна зробити висновок, що очевидним фаворитом є Telegram, оскільки він, на відміну від усіх інших

додатків, має хмарне сховище (віртуальне місце збереження даних користувача), завдяки якому можна відновлювати або переносити дані на новий пристрій. Ще однією перевагою даного месенджера є високий рівень охоплення жителів України, бо чим більше людей користуються додатком, тим більший шанс що вони натраплять на нашого бота і він їх зацікавить.

Ще до переваг Telegram також можна відноситься наявність відкритого API. Він надає можливість безкоштовно користуватися сервісами і взаємодіяти з додатком. Із вище перерахованих месенджерів тільки Whatsapp має закритий API, за який потрібно буде платити, а так як наш бот не несе комерційної мети, то ми будемо обирати з додатків, які мають відкритий API.

Крім того є така функція, як двостороння відмова – це означає, що ніхто не може написати нам повідомлення без нашого дозволу. Ми не маємо на меті, щоб наш бот нав’язував послуги. Він буде діяти лише при згоді користувача, сам мотивувати його на навчання.

Telegram – є одним із найкращих додатків із шифрування повідомлень, тому його користувачі можуть не хвилюватися за розголошення своїх повідомлень. Це ще одна із очевидних переваг даного месенджера.

Останній але не менш важливий плюс на користь Telegram – це відсутність нав’язливої реклами та орієнтованість на користувачів, що в свою чергу підвищує зручність використання та збільшує варіанти для впровадження нашого продукту в додаток.

Виходячи із таблиці та її детального аналізу ми вирішили зупинитися на Telegram, бо він є одним із найбільш орієнтованих на користувачів і розробників мобільним додатком.

1.4 Огляд існуючих рішень

Проаналізувавши всі можливі додатки і ресурси, мені не вдалося знайти бота, який би міг задовольнити зазначені нами потреби та бути орієнтованим на вивчення мов програмування. Проте існує безліч ресурсів для вивчення мов програмування:

- довідники про різні мови програмування;
- онлайн курси та вебінари;
- інтерактивні курси;
- IT-конференції;
- онлайн ресурси з отримання диплома про закінчення курсу з мови програмування;
- відео-гіди та презентації.

Таким чином, ми дійшли висновку, що хорошими ресурсами для вивчення мов програмування є сайти, на яких заплативши деяку суму грошей, можна придбати готовий курс. Але це може бути не достатньо зручним методом вивчення, так як не всі люди знають, що таке програмування і взагалі не чули про ООП та IDE.

РОЗДІЛ 2
ВИБІР ПРОГРАМНИХ ЗАСОБІВ. ПРОЕКТУВАННЯ БОТА

Отже, після того як ми пройшли всі вище перераховані етапи, обрали платформу для розробки та визначилися із їх роллю, настав час для того щоб визначитися з програмними засобами реалізації.

2.1 Мови програмування

Для початку роботи потрібно обрати мову програмування. Вона створена для взаємодії з комп'ютером шляхом написання команд і їх виконання безпосередньо на машині.

Нашого бота в Telegrami можна написати різними мовами програмування. Ми обрали Python. Він є універсальним інструментом програмування, бо за допомогою нього можна створювати сайти, ігри, мобільні додатки та багато іншого. Python володіє майже безмежними можливостями. Одна з її переваг– це синтаксис і код. Довести це ми можемо за допомогою порівняння прикладів написання привітання декількома мовами - Java, Node JS, Python.

Так виглядає код з простим привітанням, написаний на **Java**:

```
class App{
        public static void main(String[] argos){
                System.out.println(“Hello World!”);
        }
}
```

Так виглядає на **Node JS:**

```
const http = require('node:http');
 const hostname = '127.0.0.1';
 const port = 3000;
 const server = http.createServer((req, res) => {
   res.statusCode = 200;
```

```
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World\n');
 });
 server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
 });
```

Привітання на **Python**

```
Print("Hello World")
```

Порівнявши код на різних мовах програмування, можна зробити очевидний висновок: код на мові програмування Python виглядає більш зручно і зрозуміло, ніж на Java і Node JS.

Ще однією перевагою є те що Python підтримує велику кількість бібліотек та додаткових модулів, які можуть нам знадобитися і спростять та удосконалять процес написання коду.

2.2 Середовище розробки

Що ж таке середовище розробки і для чого воно потрібне?

Integrated Development Environment (IDE, середовище розробки) – це програмне забезпечення, яке дає можливість писати код та автоматизувати програми.

Обираючи середовище для розробки проекту потрібно обов'язково врахувати його орієнтованість на мову програмування. Адже, чим більший функціонал надає IDE, тим більше можливостей в реалізації нашого бота і спрощення та написання самого коду. Нижче розглянемо різні середовища розробки.

- Sublime text 4 – це легкий і зручний у використання текстовий редактор в простому дизайні. Його написано частково на Python, тому він підтримує бібліотеки цієї мови програмування. Але деякі можливості обмежені (кожну бібліотеку треба завантажувати і встановлювати), а також

він потребує в додаткове налаштування (скачування додаткових плагінів для підтримки синтаксису конкретної мови). Це забирає багато часу і не кожний зможе його налаштувати під себе.

- VSCode – це розробка компанії Microsoft. Дане середовище програмування підтримує велику кількість мов програмування. Воно є редактором коду для розробки веб додатків та хмарних додатків. Нажаль VSCode не має інтерпретатора Python і потребує також додаткового налаштування.
- PyCharm. Із самої назви даного середовища розробки стає зрозуміло, що він є Python-орієнтованим. PyCharm підтримує розробку на всіх фреймворках Pythona, також він має повну безкоштовну версію, що також є плюсом. Ще це середовище розробки володіє найбільшим функціоналом суміжних додатків та інструментів для обраної нами мови програмування.

2.3 Створення бота у Telegram

Перед безпосереднім написанням коду і подальшій розробці нашого чат-бота, нам потрібно його зареєструвати у Telegram за допомогою “BotFather” бота. Спочатку переходимо у пошукове поле додатка, потім пишемо @botfather і звертаємося до цього бота командою /newbot. Після цього штучний інтелект проводить реєстрацію нашого бота. Це необхідно було зробити для отримання токену (ключа доступу до Telegram Bot API для управління ботом через середовище розробки), та затвердження власного тегу для доступу у додатку.

Нижче, в таблиці 2.1, можна ознайомитися із функціями управління ботом, які надає “BotFather”.

Таблиця 2.1 – Функції для управління ботом

Функції	Характеристика
/help	Виводить список всіх доступних команд
/setname	Змінює ім’я бота у додатку
/setuserpic	Встановллює зображення у іконку чат-бота
/mybots	Надає можливість змінити бота
/setprivacy	Вмикає приватний режим у чаті

Перераховані вище команди являють собою набір інструментів в середовищі Telegram. Вони дають можливість різними способами взаємодіяти з ботом для додаткових налаштувань, видалення, зміни даних у разі втрати доступу та ін.

Отже, виходячи із усього вище сказаного можна зробити висновок, що Telegram – це найкраща платформа для розміщення нашого проекту.

2.4 Вибір серверної платформи для доступу до бота

Для того, щоб наш бот справно працював і міг одразу взаємодіяти із користувачами необхідно розмістити його на певному сервері. Для цього можна використати ресурс Heroku. Даний сервіс володіє власною системою контролю версій Git, а також надає доступ до хмарних обчислень.

Щоб запустити бота на сервері обраного нами сервісу потрібно встановити HerokuCLI і ввійти в особистий кабінет за допомогою команди «herokulogin». Після цього вводимо ще кілька команд для перенесення деяких файлів із сервера на комп’ютер. Якщо зв’язок між сервером та виконавчою машиною встановиться, то у терміналі PyCharm з’явиться

повідомлення про вдале підключення. У результаті цього користувачі бота будуть швидко отримувати відповіді на свої запити.

2.5 Додаткові компоненти для роботи

Після того, як ми обрали мову програмування, середовище розробки, серверну частину і зареєстрували бота, нам залишилося тільки вирішити останнє – обрати суміжні компоненти, модулі та бібліотеки для правильної побудови логіки програми.

Бібліотека telebot – це ніби зовнішня оправа для мови Python для взаємодії з API Telegram. Вона відповідає за запити, та вбудовується для спрощення роботи і функціонування коду.

Тепер можна розібрати, Telegram Bot API є http-інтерфейсом за допомогою якого можна взаємодіяти з ботом в телеграмі. Бот реєструється у додатку, йому надається унікальний токен типу: 123456:DEF-WBA7434HeIkv-qnm79J6k3u111kq1. Записи до API мають вигляд посилань, в яких йде протокол передачі гіпертекстових посилань. Потім звернення до Telegram. У ньому обов’язково має бути вказаний токен бота і назва методу. Відповіді приходять у вигляді JSON-об’єкту, який має текстове поле description. У ньому міститься опис результату запису, а також поле зі значенням ok. Це означає успішність обробки запиту. Якщо виконання запиту невдале, то це поле буде мати значення false. Там можна буде побачити причини помилки.

2.6 Проектування бота на основі стандартів IDEF0 та UML

Почнемо з того, що таке IDEF0. Це методологія функціонального моделювання та графічна нотація, призначена для формалізації та опису бізнес-процесів. Відмінною рисою IDEF0 є її акцент на супідрядність об’єктів. У IDEF0 розглядаються логічні відносини між роботами, а чи не їх тимчасова послідовність.

Стандарт IDEF0 представляє собою організацію як набір модулів. Тут існує правило – найважливіша функція знаходиться у верхньому лівому кутку. Ще є правила сторі:

· верхні стрілки – це елементи управління;

· ліва сторона – це вхідне значення, яке провокує початок процесу;

· права сторона – це вихідні значення;

· нижня сторона – це механізми, які використовуються для виконання завданого процесу.

Нижче, на рис. 2.2 зображено вигляд контекстної діаграми процесу розробки бота в Telegram.

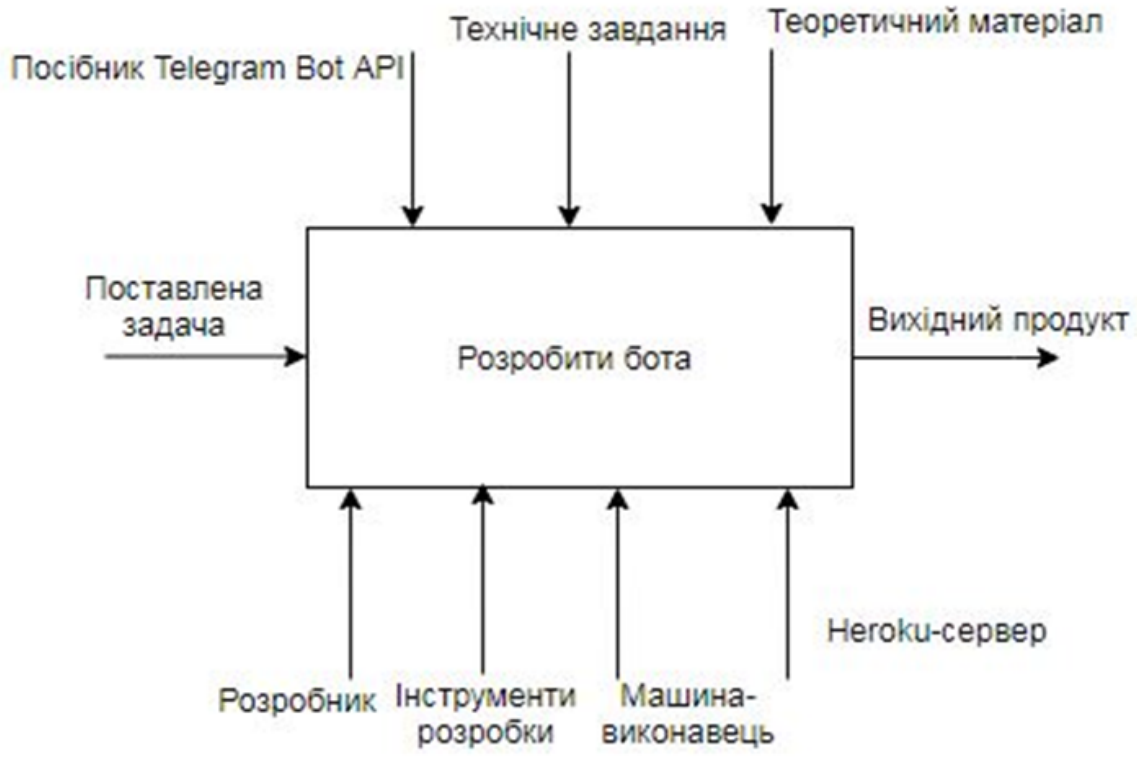

Рисунок 2.2 – Контекстна діаграма процесу розробки бота

Відповідно до контекстної діаграми маємо створити діаграму декомпозиції моделі. Діаграма декомпозиції потрібна для детальнішого розгляду процесу, зазначеного в контекстній діаграмі, за рахунок розділення головного процесу на додаткові. Складання діаграми декомпозиції дає

можливість детально описати процес розроблення бота і зрозуміти порядок виконання дій.

Розглянемо основні етапи розроблення бота:

1. Реєстрація бота у Telegram.
2. Процес розробки.
3. Тестування бота.

Останнім етапом йде тестування і після його завершення робота вважається виконаною. Це все допоможе у майбутньому в процесі розробки і визначенні терміну виконання завдання.

Так як у контекстній діаграмі показаний процес створення бота, то діаграма декомпозиції буде описувати як певні фактори та механізми впливають на різні процеси в ході виконання роботи. Діаграма декомпозиції зображена на рис. 2.3.

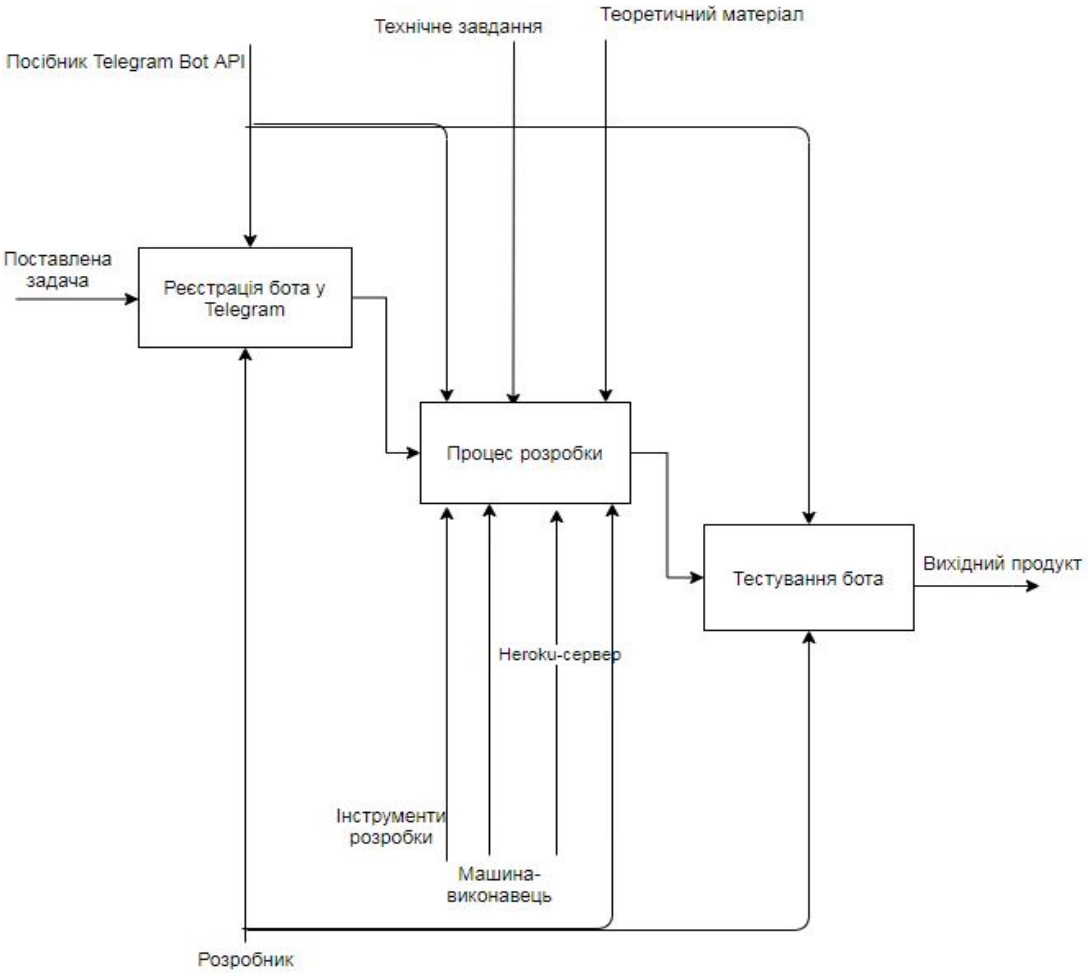

Рисунок 2.3 – Діаграма декомпозиції розроблення бота

Діаграма варіантів використання демонструє учасників, які діють у системі та функції, які вони можуть використовувати. У результаті встановлюється зв’язок функціоналу інтерфейсу бота із його користувачами. Це, в свою чергу, дає можливість у процесі розробки розуміти , яким чином вибудувати навігацію і встановлювати обмеження.

Добудована діаграма зображена на рисунку 2.4.

Рисунок 2.4 – Діаграма варіантів використання у системі

Як бачимо, дозволених типів користування лише два. Адміністратору доступні усі функції звичайного користувача, але крім цього він також отримує інформацію про взаємодії користувача в системі.

Наступна діаграма показує нам, як бот перевіряє наявність нових зв’язків у системі. Для того, щоб бот відповів на повідомлення користувача йому потрібно періодично звертатися до інтерфейсу та перевіряти нові запити користувачів.

Діаграма перевірки оновлень бота зображена на рис. 2.5.

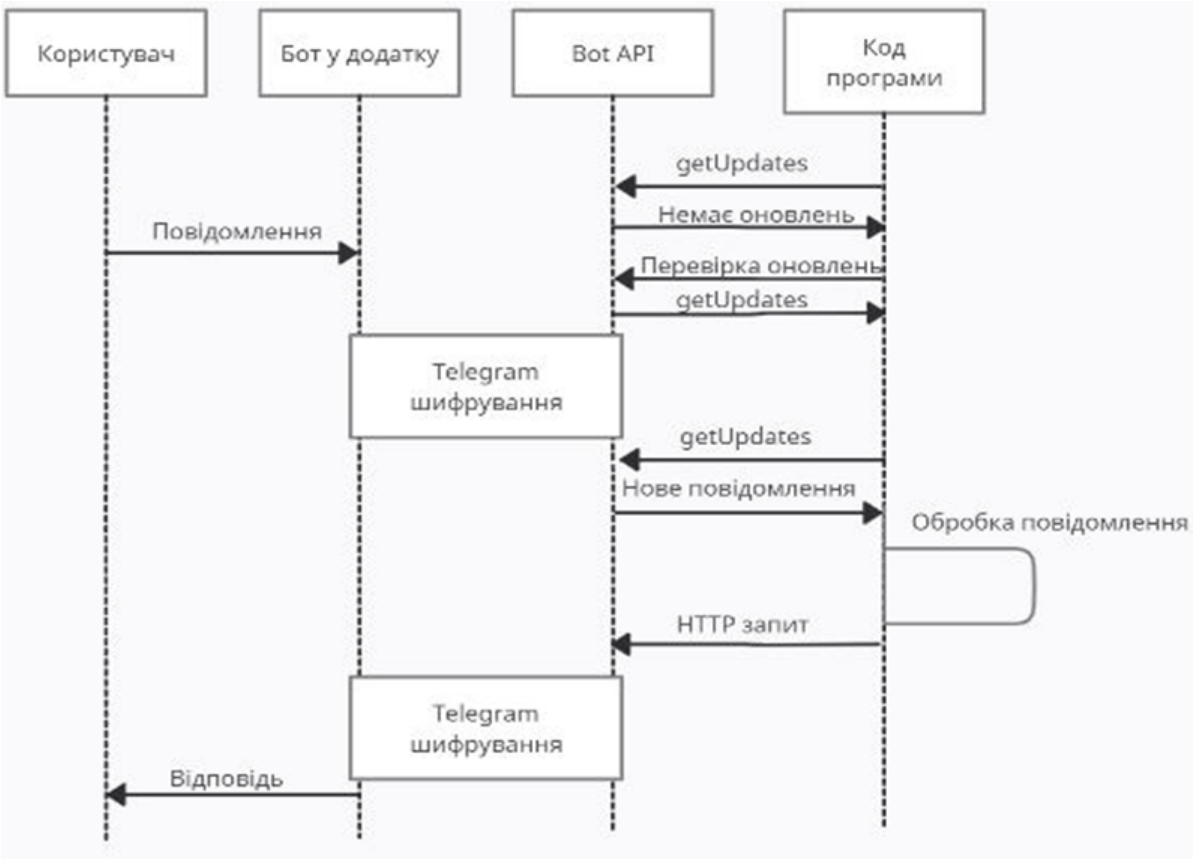

Рисунок 2.5 – Перевірка нових повідомлень

Перевірка нових повідомлень здійснюється за допомогою методу getUpdates. Цей метод вбудований у бібліотеку Telegram Bot API. Він потрібен для перевірки стану діалогового інтерфейсу між користувачем та ботом та виглядає наступним чином (рис. 2.6).

```
MethodGetUpdates = 'https://api.telegram.org/bot{token}/getUpdates'.format(token=TOKEN)


while True:
    response = requests.post(MethodGetUpdates)
    result = response.json()
    print result
```

Рисунок 2.6 – Лістинг коду отримання запиту від getUpdates

РОЗДІЛ 3
НАЛАШТУВАННЯ ІНСТРУМЕКНТІВ ТА РОЗРОБЛЕННЯ БОТА

3.1 Реєстрація бота у Telegram

Тепер, коли ми обрали технології і програмні засоби для реалізації проекту, можна приступити до його виконання. Почнемо з детального розгляду процесу реєстрації нашого бота. Для цього потрібно в рядку пошуку прописати @botfathery і перейти до діалогу з ботом (рис. 3.1).

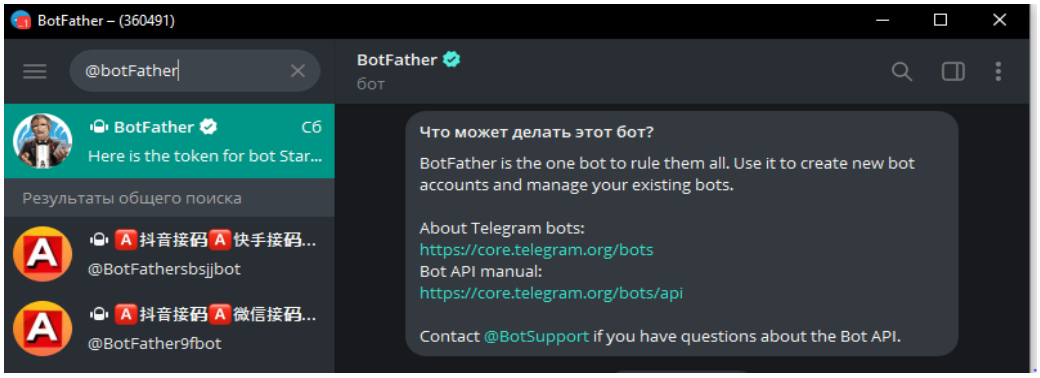

Рисунок 3.1 – Початок реєстрації бота у Telegram додатку

Набираємо команду “/start” і відправляємо, щоб бот дав відповідь. Він надішле нам меню функцій, які він може виконувати (рис. 3.2).

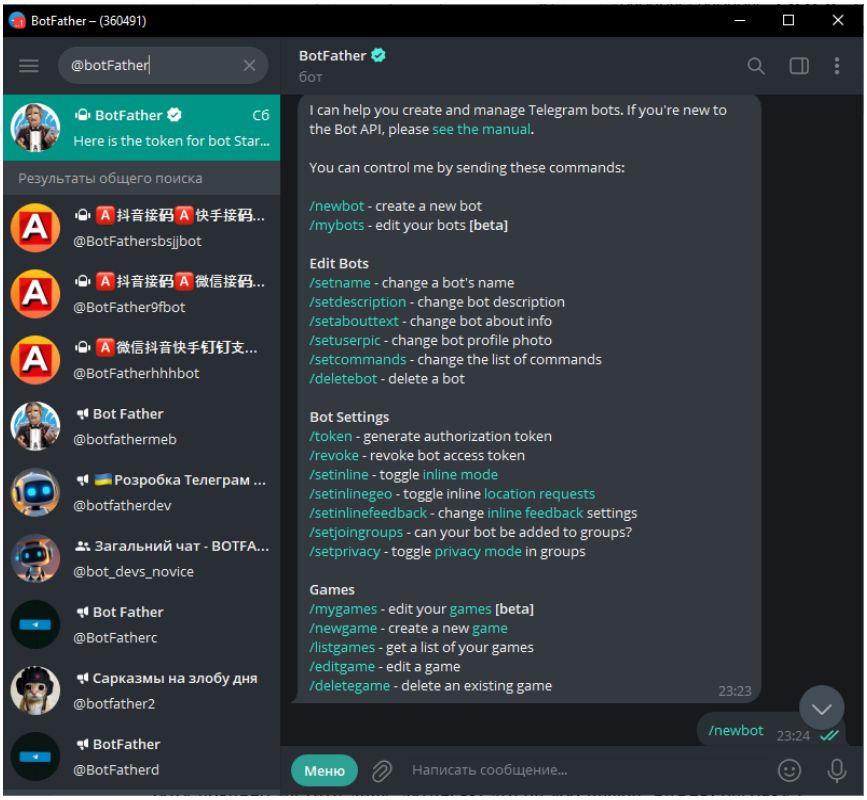

Рисунок 3.2 – Функції, що надає BotFather

Для створення бота у додатку потрібно ввести команду “/newbot”. Далі треба написати назву нашого бота. Вона потрібна лише для того, щоб користувач мав якесь уявлення про те, що це за бот і які його функції. Ім’я бота повинно містити лише латинські літери або цифри, нижню рисочку і не має повторюватися. Крім того воно повинно закінчуватися на bot.

Після того, як ми закінчили попередній процес, нам в чат приходить сповіщення про завершення реєстрації нашого бота. У ньому є токен для доступу до нашого бота по HTTPAPI, а також посилання на бота в додатку Telegram (рис. 3.3).

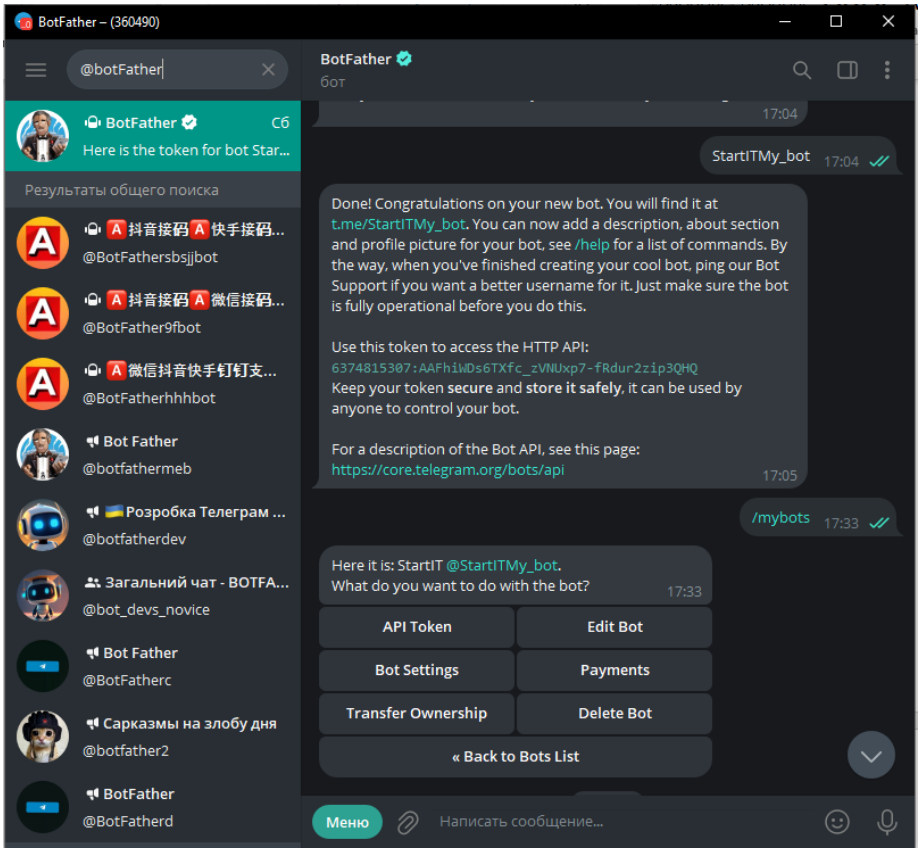

Рисунок 3.3 – Повідомлення із ключем доступу та посилання на бота

Після цього можна перейти за посиланням, де нам відкриється наш бот, який ми будемо доповнювати командами і доводити його до готового виду. Поки що він повністю порожній і готовий до експериментів (рис. 3.4).

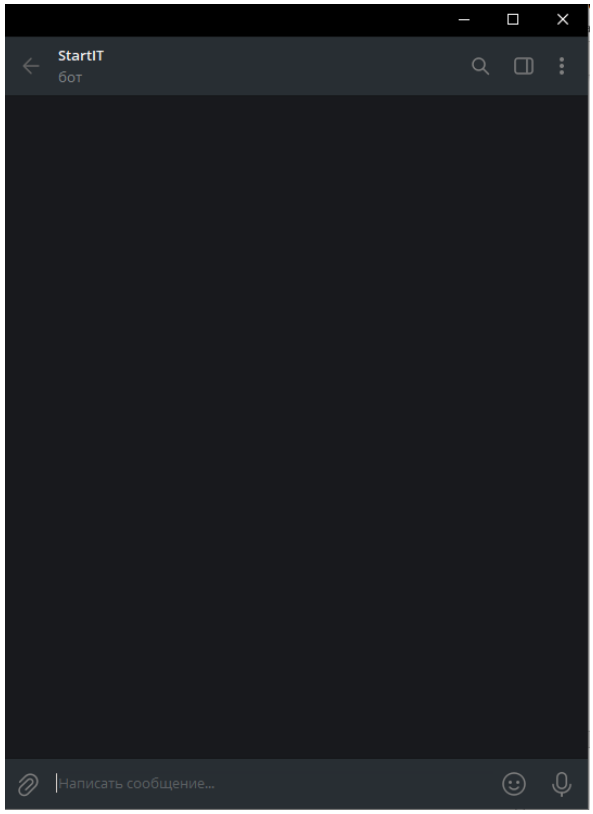

Рисунок 3.4 – Вікно чату із зареєстрованим ботом

Також ми можемо натиснути на іконку зображення нашого бота і переглянути інформацію профіля. Він виглядає стандартно (рис. 3.5).

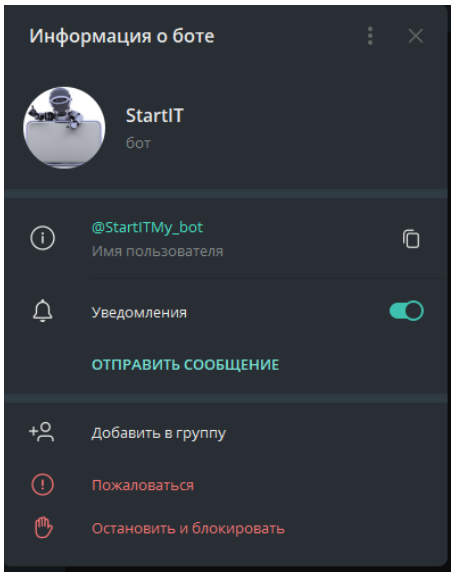

Рисунок 3.5 – Вікно додаткової інформації користувача

Для того щоб почати роботу з ботом, потрібно написати команду “/start”. Нажаль, наш бот не дасть нам відповідь на цей запит, так як він ще

не вміє відповідати на http запити. Цього ми навчимо його згодом за допомогою коду, який буде керувати нашим ботом.

Якщо ми подивимося на цей інформаційний блок, то зрозуміємо, що цей чат майже ніяк не відрізняється від чату спілкування із живою людиною. Він слугує чистим листом для роботи, тому що весь код, який ми напишемо у середовищі розробки, буде виконуватися і виводитися у вигляді кнопок, зображень та інформаційних вікон у текстовому форматі. Це можна назвати, як інтерфейс для взаємодії з програмою. Також усі зв'язки та шифрування Телеграму при такому виконанні не будуть відноситися до протоколу шифрування. Для отримання відповіді на запит користувача, вони передаються по протоколу HTTPS. Форма вигляду цих запитів повинна мати такий вигляд: https://api.telegram.org/bot'token'/назва_метода.

Також досить важливим є оновлення зображення бота, так як візуальне оформлення може зіграти позитивну роль в заохоченні користувачів натиснути на бота та почати із ним діалог. Для того щоб оновити зображення нашого бота, потрібно перейти до BotFather і відправити запит "/setuserpic", а вже потім відправити потрібне зображення в чат (рис. 3.6). Далі перейдемо до нашого бота і перевіримо чи встановлено нове зображення профілю (рис. 3.7).

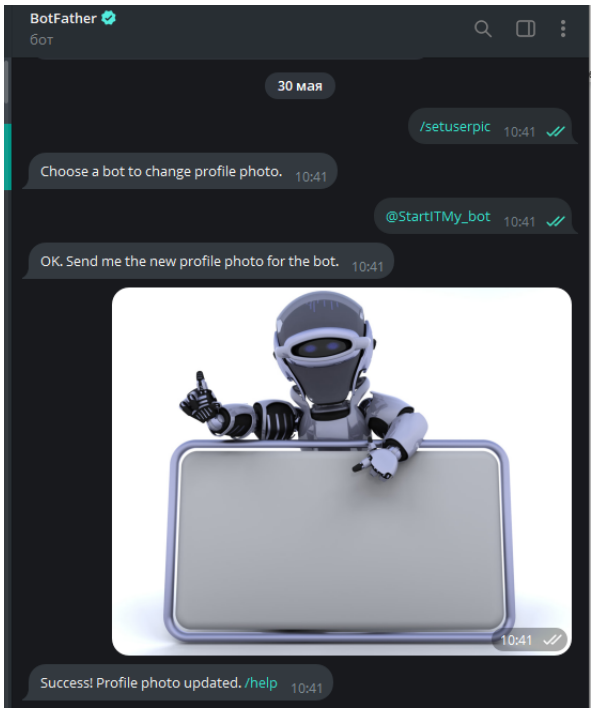

Рисунок 3.6 – Оновлення зображення для бота

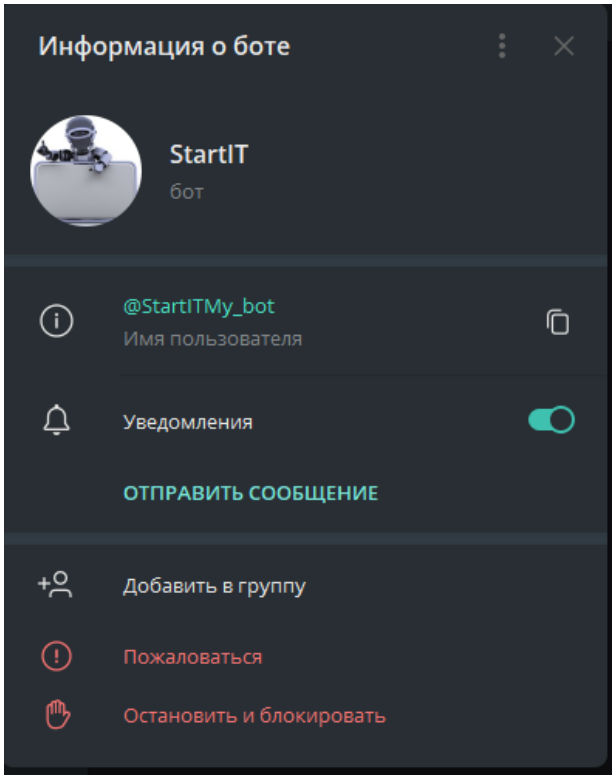

Рисунок 3.7 – Нове зображення профіля бота

3.2 Налаштування середовища розробки

Тепер можемо приступити до налаштування середовища розробки та підключення усіх необхідних ресурсів для комфортної роботи. Почати потрібно буде із встановлення останньої версії Python на наш комп’ютер. Завантажимо інсталятор із офіційного сайту. Варто звернути увагу на стабільність версії і її сумісність з операційною системою. Остання версія Python є 3.12.3. Вона підтримується майже всіма операційними системами. Перед встановлення останньої версії треба ознайомитися із документацією цієї версії. Це може спростити нам написання коду (рис. 3.8).

Рисунок 3.8 – Оновлення версії 3.12.3 від 2 жовтня 2023

Після завантаження та встановлення останньої версії Python у вікні натискаємо кнопку Update Now та очікуємо повного встановлення (рис. 3.9).

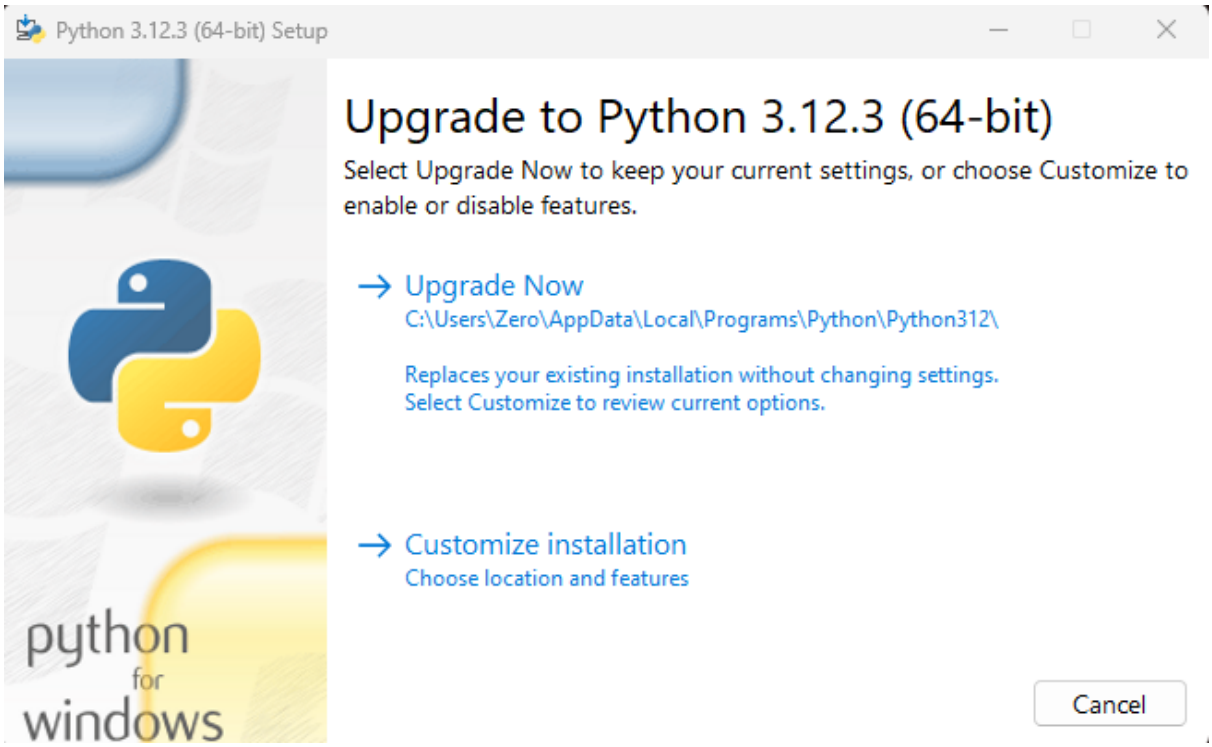

Рисунок 3.9 – Встановлення оптимальної версії Python

Після встановлення Python і PyCharm можна перейти до його налаштування. Відкриваємо середовище розробки PyCharm і створюємо новий проект. Для цього необхідно натиснути File - NewProject (рис. 3.10), далі обрати папку в якій буде зберігатися наш проект та середовище серед запропонованих:

- virtualenv;
- pipenv;
- conda.

На даний час середовище розробки PyCharm пропонує три віртуальних середовища. Якщо ми оберемо середовище virtualenv, то там все одно потрібно буде звертатися до pip, так як буде необхідно встановити та завантажити бібліотеки, а вже потім звернутися до pip freeze для збереження встановлених додатків.

Для початку розглянемо pipenv, який був створений через те, що virtualenv мав багато недоліків. Завдяки цьому середовищу можна завантажувати потрібні пакети не тільки в папку проекту, а і до файлу pipenv. Це дає нам змогу отримувати повторний доступ до всіх підключених раніше пакетів за допомогою зберігання їх в pip-файл.

Середовище “Conda” має чітке орієнтування на використання NumPy і SciPy пакетів, які використовуються для роботи з масивами, функціями, науковими та інженерно-технічними розрахунками.

Виходячи з вище сказаного, наш вибір падає на pipenv за рахунок того, що воно являє собою удосконалену версію іншого віртуального середовища.

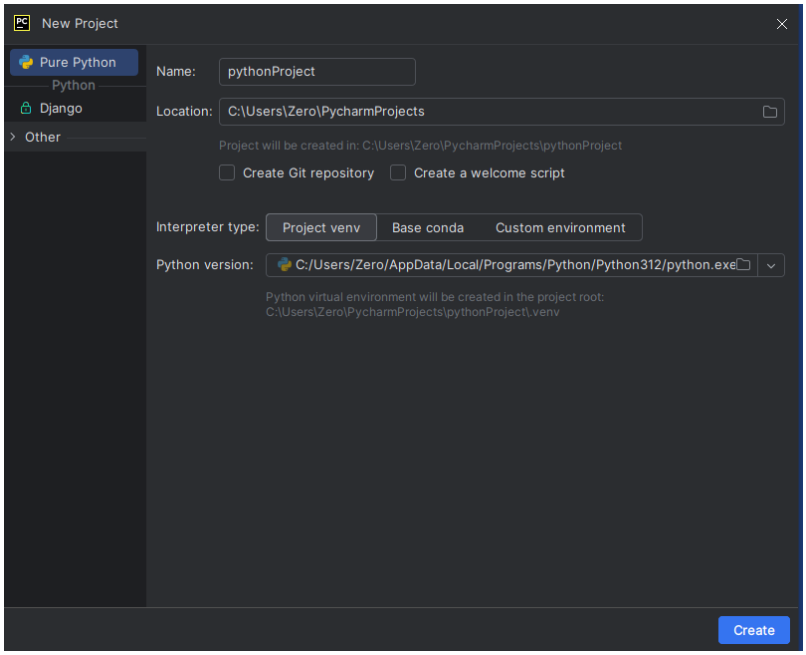

Рисунок 3.10 – Налаштування проекту в середовищі розробки

Що таке інтерпретатор? Інтерпретатор – це програма, яка потрібна для виконання програм написаних у середовищі розробки. Він здійснює обробку всього коду та перетворює написану логіку у машинний код, який виконує програма.

Ми обрали базовий інтерпретатор. За замовчуванням, середовище розробки пропонує останню версію.

Частіше за все складні програми, написані мовами високого рівня, потім перетворюються в машинний код для виконання процесором комп’ютера. У процесі створення програмного забезпечення, розробник може постійно змінювати та редагувати початковий код. При використанні компілятора, кожного разу він компонує усі файли разом, перед тим як виконати програму. Отже, якщо код програми є досить великим, то час на його збірку у компіляторі може бути досить тривалим. Найцікавіше те, що інтерптетатору не потрібно бачити увесь код відразу. Йому треба лише той код, який виконується саме зараз. Це займе менше часу на виконання програми. Його принцип роботи було спроектовано і відображено на рис. 3.11.

Рисунок 3.11 – Принцип роботи інтерпретатора

Тепер розберемось, що таке pip. Pip – це система керування пакетами, що слугує для встановлення і управління програмними пакетами, написаними мовою програмування Python. Після того, як ми налаштували наше робоче середовище, наступним кроком є завантаження всіх обраних нами бібліотек. Якщо розібратися більш детально, то починаючи із Python 3.4 система pip встановлюється разом із інтерпретатором. Отже, оскільки ми встановили Python 3.12.3, то окремо інсталювати pip не потрібно. Далі розберемося із основними командами pip:

- piphelp – допомога згідно доступних команд;
- piplist – список встановлених додатків;
- pipsearch – пошук пакетів за назвою;
- pipinstallpackage_name – встановлення бібліотек та пакетів;
- pipuninstallpackage_name – видалення бібліотек та пакетів за їх іменем;
- pipshowpackage_name – надання інформації про встановлену команду;
- pipinstall-U – оновлення обраних бібліотек.

При запуску середовища розробки, ми відкрили термінал. Тепер розберемося, що він собою являє. Термінал або консоль – це текстовий інтерфейс для роботи з операційною системою. Іноді використовують командний рядок операційної системи, але він має менший функціонал,

ніж його аналог. Тому часто радять проводити завантаження пакетів та бібліотек саме через інтерпретатор, який знаходиться в самому середовищі розробки.

Отже, щоб встановити пакети за допомогою системи керування пакетами, потрібно відкрити термінал та написати команду: pipinstall –”назва пакету” (без лапок). Як пройшло встановлення бібліотеки TelegramBotAPI у наш проект показано на зображенні нижче (рис. 3.12).

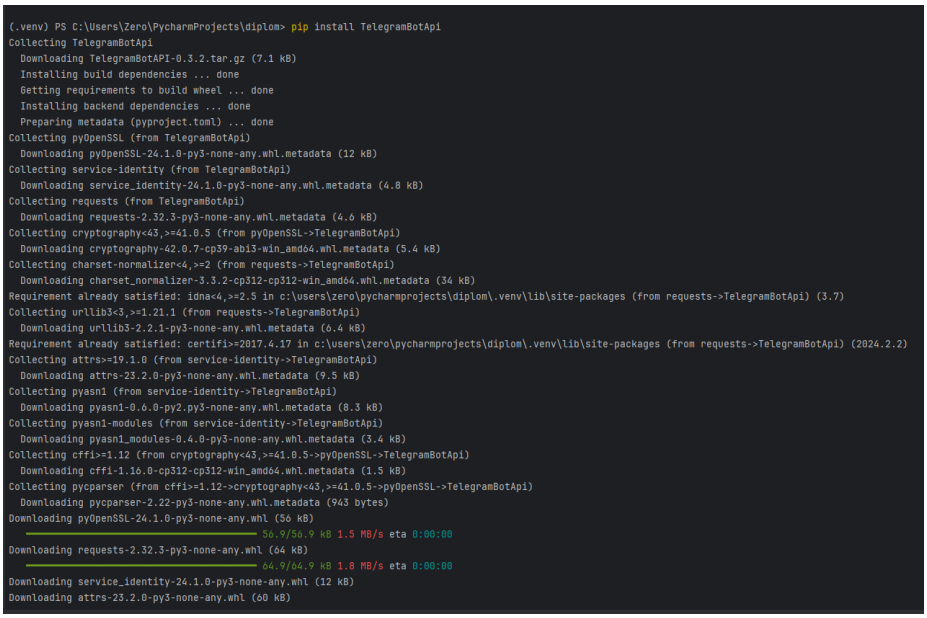

Рисунок 3.12 – Встановлення бібліотеки до проекту

TelegramBotAPI – це спеціальний модуль, створений на базі Телеграму для написання ботів. Його використання досить просте. Нам не потрібно буде розбиратися як працює протокол шифрування Телеграму. Ми будемо підтримувати зв’язок з нашим додатком за допомогою https запиту на проміжний сервер. Звичайно, краще і легше використовувати уже створені https запити чи сторонні бібліотеки, ніж писати власноруч https запити.

Взагалі існує два типи отримання оновлень від нашого бота. Перший – метод getUpdate та другий – веб-куків. Вхідні дані будуть зберігатися на сервері на протязі 24 годин, доки їх не буде видалено.

Об’єкт Update вважається вхідним оновленням. Таке оновлення має на увазі, що це будь-яка взаємодія користувача з нашим ботом. Нижче представлені варіанти. Тільки один із показаних елементів може існувати в

одному оновленні. Йде повторення проходить кожного разу, коли користувачі проводять певну взаємодію з ботом в інтерфейсі чату.

Поле	Тип	Опис
update_id	Integer	Унікальний ідентифікатор оновлень.
message	Message	Нове вхідне повідомлення будь-якого типу: текст, фото, стікер, і т.д.
inline_query	InlineQuery	Новий інлайн-запит
chosen_inline_result	ChosenInlineResult	Результат будь-якого запиту, що був відправлений користувачем в чат

Таблиця 3.1 – Отримання оновлень в чаті

Тепер поговоримо про метод getUpdate. Він використовується для отримання оновлень через технологію, яка отримує інформацію про нові дії за допомогою довгих запитів. Відповідь на такі запити приходе у вигляді масиву об’єктів Update.

Поле	Тип	Опис
offset	Integer	Ідентифікатор першого повернутого оновлення. Повинен бути більшим ніж самий великий ідентифікатор в попередніх оновленнях.
limit	Integer	Ліміт кількості оновлень. Дозволені числа від 1 до 100.
timeout	Integer	Перерва до запиту в секундах. За замовчуванням 0

Таблиця 3.2 – Команди для вище описаного методу

Типи, які дозволені в модулі є JSON-об’єктами. Далі будуть рисунки, на яких зображені об’єкти керування ботом (рис. 3.13–3.15).

User

Цей об'єкт зображає бота чи користувача Telegram.

Поле	Тип	Опис
id	Integer	Ідентифікатор користувача чи бота
first_name	String	Ім'я бота
username	String	. Username користувача чи бота

Рисунок 3.13 – Об’єкт User та його команди

Даний об’єкт дає можливість дізнатися про ідентифікатор користувача або бота, його ім’я у чаті. Також важливо відразу відмітити, що

усі об’єкти, які залежать від регістра, повинні знаходитися у форматі кодування UTF-8.

Chat

Об'єкт чат - інтерфейс зв'язку між ботом та користувачем.

Поле	Тип	Опис
id	Integer	Ідентифікатор чату. Абсолютне значення не перевищує 1e13
type	Enum	Тип чату: “private”, “group”, “supergroup” чи “channel”
title	String	Назва для канала чи групи
first_name	String	. Ім'я користувача в чаті
last_name	String	Прізвище користувача в чаті
all_members_are_administrators	Boolean	.True, якщо усі користувачі являються адміністраторами

Рисунок 3.14 – Об’єкт Chat та його команди

Отже, об’єкт, зображений на Рисунку 3.14 дає команди керування інтерфейсом. Тобто це зв’язок між ботом і користувачем. Команди дають можливість змінювати тип чату, його опис, ідентифікувати ім’я користувача в чаті та ін.

Message

Об'єкт повідомлень у чаті.

Поле	Тип	Опис
message_id	Integer	Ідентифікатор повідомлення
from	User	Відправник повідомлення
date	Integer	Дата відправки повідомлення
chat	Chat	Діалог в якому було відправлено повідомлення
forward_from	User	Хто є відправником оригінального повідомлення, у разі пересланих повідомлень
forward_date	Integer	Дата відправки оригінального повідомлення, у разі пересланих повідомлень
text	String	Для текстових повідомлень: текст повідомлення, 0-4096 символів
contact	Contact	Інформація про відправлений контакт
location	Location	Інформація про місцезнаходження
group_chat_created	True	Повідомлення про створення групового чату
pinned_message	Message	Вказане повідомлення було прикріплено до шапки чату

Рисунок 3.15 – Об’єкт Message та його команди

Даний об’єкт відповідає за повідомлення у чаті, а саме – за інформацію про місцезнаходження, відправлений контакт, дату відправки повідомлень та інше.

Існують бібліотеки, які допоможуть нам у процесі створення бота:

- aiogram;
- telebot;
- python-telegram-bot;

Бібліотека telebot підійде для ботів, які не мають великого трафіку, складних інтеграцій і зв’язків зі сторонніми ресурсами. Тому він підходить для поставленої нами задачі. Взагалі, усі бібліотеки мають свої переваги. Так, наприклад, aiogram є асинхронною бібліотекою. Вона підходить для реалізації складних проектів з інтеграцією додаткових ресурсів. Як проходе процес встановлення бібліотеки показано нижче (рис. 3.16).

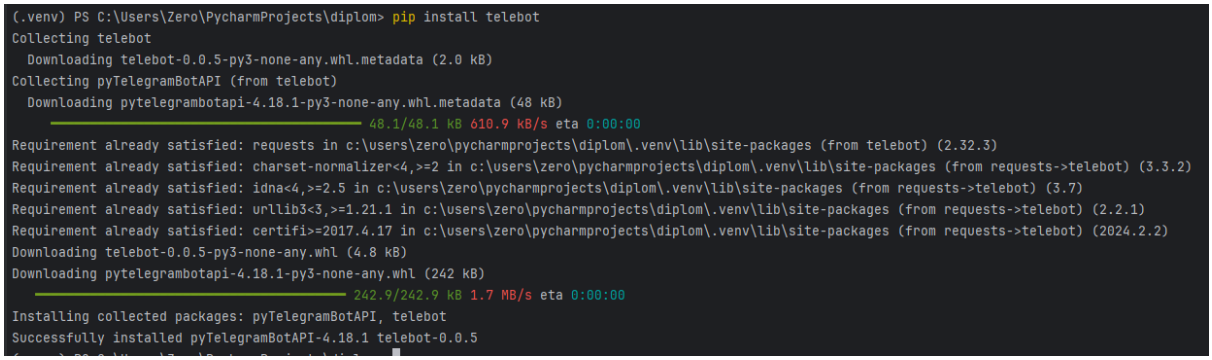

Рисунок 3.16 – Встановлення бібліотеки до нашого проекту

Ще існує така бібліотека як requests. Вона є стандартним інструментом для створення http-запитів. Завдяки програмному інтерфейсу значно покращується процес створення запитів, що дає можливість зосередитися на використанні даних та взаємодії зі службами.

Get метод бібліотеки є одним із найпопулярніших методів. Він існує для того щоб вказувати на те, що з ресурсу проводиться спроба отримати дані. Відповідь на цей запит майже завжди несе в собі інформацію, яка знаходиться у тілі повідомлення та має назву payload.

Об’єкт response використовується для отримання відповіді. Він являє собою інструмент, який аналізує результат запитів. Об’єкт response виконується для вивчення даних, отриманих у результаті запиту.

Тепер розглянемо коди станів про статус запиту. Якщо отримуємо код статусу запиту 200, тобто відповідь на нього, то це означає, що запит виконаний успішно. Щоб отримати статус, потрібно написати команду представлену нижче (рис. 3.17).

Рисунок 3.17 – Команда коду станів та відповідь на неї

Щоб отримати зміст запиту в байтах потрібно використовувати .content. Ця команда надає нам доступ до будь-яких даних у тілі запиту. Але це зазвичай потребує конвертації даної інформації в поширеному стандарті кодування. Отримані відповіді приходять нам у форматі JSON. Самі коди

станів дають нам багато можливостей, але перед їх використанням потрібно ознайомитися з базовими поняттями протоколу шифрування.

3.3 Створення головного файлу

Після того, як ми повністю налаштували наш робочий простір і додали бібліотеки, створимо головний файл у форматі “.py”. У цьому файлі ми імпортуємо бібліотеки та модулі. Це потрібно для того, щоб їх компоненти та команди працювали коректно і саме середовище розробки не знаходило помилок у нашому коді. Імпорт самих модулів та бібліотек відбувається у вигляді звичайного коду (рис. 3.18).

```
import logging
from telegram import Update, ReplyKeyboardMarkup
from telegram.ext import Application, CommandHandler, MessageHandler, ContextTypes, filters
from handlers.learning import handle_learning_message, start_learning
from handlers.tests import handle_tests_message, start_tests

```

Рисунок 3.18 – Імпортування бібліотек у код проекту

Для початку потрібно створити команду “/start”. Вона потрібна для того, щоб бот міг реагувати на повідомлення в чаті і пропонувати свої послуги. Після цієї команди бот пропонує меню з вибором інформації. До цього потрібно додати токен, який видав BotFather раніше. Це служить для того, щоб саме потрібний бот отримував команди, що генеруються у коді програми (рис. 3.19).

```
1 usage
def main():
    # Створення Application та передача токену бота
    application = Application.builder().token("6374815307:AAFhiWDsбTXfc_zVNUxp7-fRdur2zip3QHQ").build()
```

Рисунок 3.19 – Команда на додавання токену у програму

Після того, як ми надали ключ, звертаємось до бота напряму за допомогою команд, які є у завантажених у віртуальне середовище проекту бібліотеках. Для прикладу команда зображена на рис. 3.20 дає можливість боту розпізнавати контент текстового формату.

```
# Функція для обробки текстових повідомлень
1 usage
async def handle_message(update: Update, context: ContextTypes.DEFAULT_TYPE):
    text = update.message.text
    if text == 'Навчання':
        await start_learning(update, context)
    elif text == 'Тести':
        await start_tests(update, context)
    elif text == 'Назад':
        await back(update, context)
    elif text == 'Головна':
        await main_menu(update, context)
    else:
        handled = await handle_learning_message(update, context) or await handle_tests_message(update, context)
        if not handled:
            await update.message.reply_text( text: "Будь ласка, оберіть 'Навчання' або 'Тести' з меню.", reply_markup=main_menu_markup)
```

Рисунок 3.20 – Команда розпізнавання тексту

Після того як ми зрозуміли, як працює кнопка “/start”, наступним кроком буде формування загального вигляду роботи бота. Проаналізувавши різні варіанти оформлення інтерфейсу вікна чату, було вирішено додати кнопки для зручного переходу від одного блоку до іншого. Кнопки дають можливість створити якісну навігацію в ботові для користувачів. Перше що потрібно буде зробити – це з’ясувати різницю між видами кнопок у додатку Телеграм.

Розпочати можна з Reply Keyboard Markup. Це шаблони повідомлень. Завдяки ним бот задає питання у вигляді текстових повідомлень і пропонує варіанти відповіді для подальшого переходу в архітектуру бота. Користувач має можливість як самостійно набирати текст з віртуальної клавіатури, так і натискати готові кнопки в інтерфейсі бота. Такий формат кнопок не може містити ніякої інформації. І все, що відбудеться при натисненні безпосередньо на кнопку – відправка того змісту, який на ній написаний. Приклад створення такої кнопки зображено нижче (рис. 3.21).

```
# Додавання обробників команд
application.add_handler(CommandHandler( command: "start", start))
application.add_handler(MessageHandler(filters.TEXT & ~filters.COMMAND, handle_message))
```

```
# Функція для команди /start
1 usage
async def start(update: Update, context: ContextTypes.DEFAULT_TYPE):
    await update.message.reply_text(
        text: "Привіт! Я бот, який  допомогає навчатися  програмувати та складати тести.",
        reply_markup=main_menu_markup
    )
```

Рисунок 3.21 – Приклад написання шаблонних кнопок

У цьому випадку показано простий приклад написання шаблонної кнопки із використанням іншої бібліотеки, але її принцип незмінний. Ця

кнопка може лише виводити текст вказаний на ній в чат і слугує більше простим ботам, де йде виконання простих функцій, наприклад спілкування з користувачем (рис. 3.22).

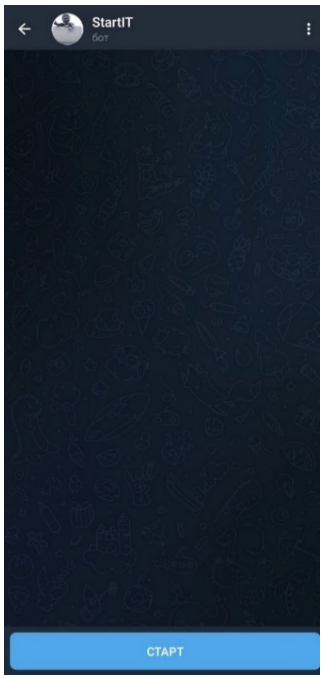

Рисунок 3.22 – Зовнішній вигляд шаблонної кнопки

Дана кнопка виглядає як звичайна клавіатура і знаходиться у меню керування.

Тепер можна більш детально розібрати Reply Keyboard Markup. Створення клавіатури буде відбуватися за допомогою main_menu_keyboard, він містить масив кнопок, що будуть відображатися в інтерфейсі нашого Телеграм бота. У нашому випадку вона містить дві кнопки «Навчання» і «Тести». Після цього у нас йде функція обробки натискань на кнопки «**handle_message**». Вона перевіряє текст повідомлення, яке надійшло від користувача. Якщо текст відповідає одному з визначених варіантів ("Навчання", "Тести", "Назад", "Головна"), виконується відповідна дія (наприклад, запуск функції **start_learning** або **start_tests**). Це дозволяє користувачеві взаємодіяти з ботом, просто натискаючи на кнопки, замість введення тексту вручну. Нижче приведений приклад розробки кнопки (рис. 3.23).

```python
# Функція для обробки текстових повідомлень
1 usage
async def handle_message(update: Update, context: ContextTypes.DEFAULT_TYPE):
    text = update.message.text
    if text == 'Навчання':
        await start_learning(update, context)
    elif text == 'Тести':
        await start_tests(update, context)
    elif text == 'Назад':
        await back(update, context)
    elif text == 'Головна':
        await main_menu(update, context)
    else:
        handled = await handle_learning_message(update, context) or await handle_tests_message(update, context)
        if not handled:
            await update.message.reply_text( text: "Будь ласка, оберіть 'Навчання' або 'Тести' з меню.", reply_markup=main_menu_markup)
```

Рисунок 3.23 – Приклад коду кнопок

Як видно з назви кнопки вона знаходиться прямо у вікні чат-бота. Єдина відмінність – форма (одна заокруглена, а друга прямокутна). Приклад вигляду такої кнопки зображено на рис. 3.24.

Рисунок 3.24 – Зовнішній вигляд кнопок

Після успішного завершення вибору програмного забезпечення та способів реалізації, можна приступити до опису функціонала нашого бота.

Наші дії під час розроблення бота:

- дослідження різних ресурсів для збору і структурування інформації для якісної реалізації курсів;
- створення кнопки «/start» (виклик головного меню бота);
- створення головного навігаційного меню (доступ до різних структурованих курсів);
- створення додаткових навігаційних кнопок для покращення взаємодії з ботом (кнопка повернення на крок назад і вихід на головне меню бота);
- розробка тестів для закріплення знань після проходження уроків по програмуванню.

РОЗДІЛ 4
РЕАЛІЗАЦІЯ ТА ОГЛЯД ФУНКЦІОНАЛЬНОЇ ЧАСТИНИ. РОБОТА ЧАТ-БОТА

Тепер ми можемо приступити до завершального етапу – тестування нашого бота в роботі та його опису.

4.1 Перевірка кнопок навігації та логіки команд в інтерфейсі бота

Почнемо із перевірки кнопок навігації та логіки команд в інтерфейсі нашого бота. Для цього ми відкриваємо бот і напишемо команду /start для початку спілкування із ним (рис. 4.1).

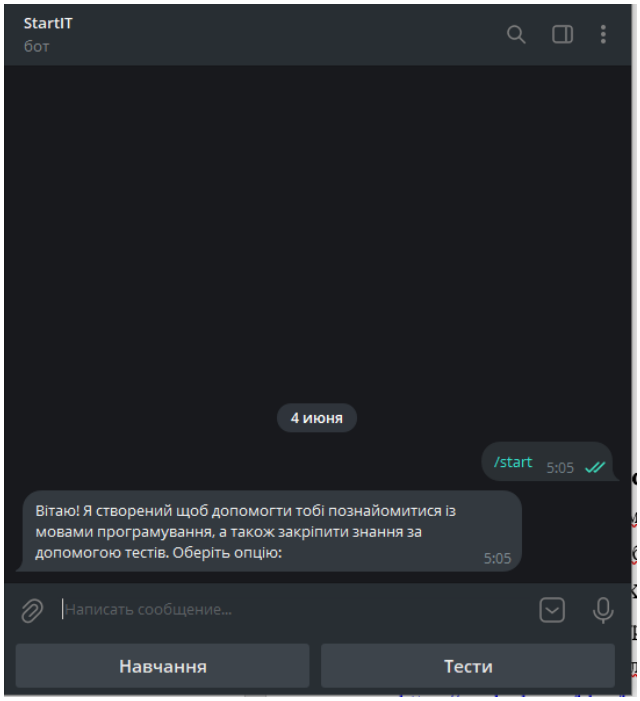

Рисунок 4.1 – Початок спілкування з ботом

Одразу потрібно буде подивитися, як у нас виконується логування. На рис. 4.2 представлена частина коду, яка відповідає за логування (процес конфігурації механізму введення логів; він дозволяє відстежувати та записувати різноманітні події.

```
# Установка логування
logging.basicConfig(
    format='%(asctime)s - %(name)s - %(levelень)s - %(повідомлення)s',
    level=logging.INFO
)
```

Рисунок 4.2 – Логування

Як ми бачимо на рис. 4.2 у нас представлена функція для налаштування базового конфігураційного процесу логування. Level=logging.INFO – для нормального ходу виконання програми.

Далі можна розглянути меню вибору категорій навчання, яке показано на рис. 4.3. **category_menu_keyboard** – список для вибору категорії навчання.

```
# Меню вибору категорій навчання
category_menu_keyboard = [['Програмування на Java', 'Програмування на PHP'], ['Назад', 'Головна']]
category_menu_markup = ReplyKeyboardMarkup(category_menu_keyboard, one_time_keyboard=True, resize_keyboard=True)
```

Рисунок 4.3 – меню вибору категорій навчання

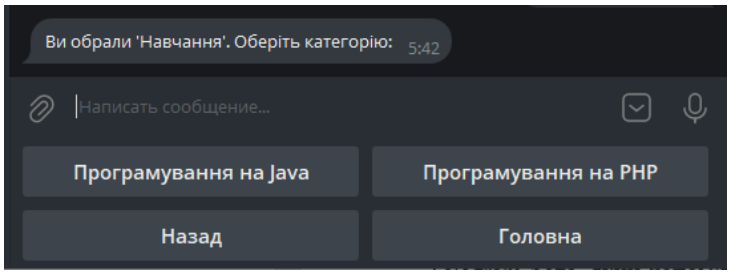

Рисунок 4.4 – Зовнішній вигляд кнопок навчання

Наступним кроком буде підменю навчання з кнопками. Сюди також можна віднести кнопки для покрокового навчання в обраній категорії. На рисунку 4.5 показано зовнішній вигляд, а на рис. 4.6 код підменю кнопок.

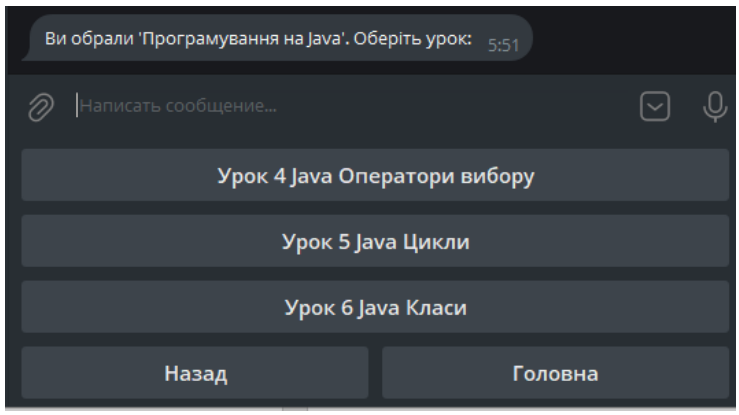

Рисунок 4.5 – Навчання на java

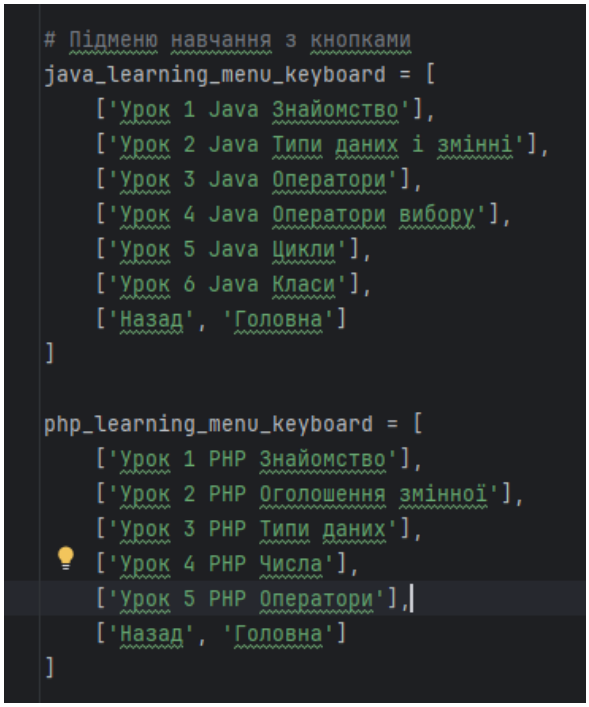

Рисунок 4.6 – Код кнопок для вибору уроку

4.2 База даних SQLite

Тепер можна розглянути функцію для отримання уроків із бази даних. У нашому проекті використовується база даних SQLite, яка не потребує окремого сервера і може використовуватися безпосередньо в проекті.

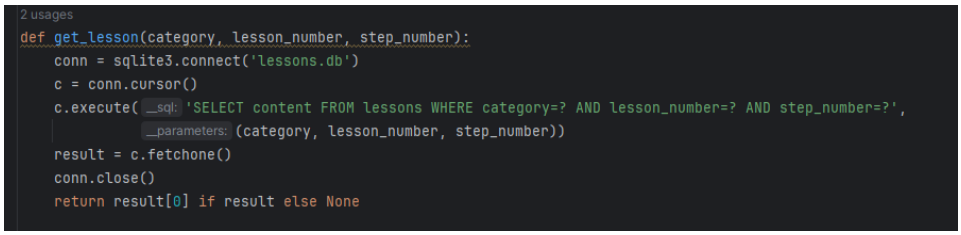

Рисунок 4.7 – Отримання уроків із бази даних

SQLite – це зручне і компактне рішення для зберігання та керування даними. Особливо вона зручна у мобільних додатках, де немає необхідності в складних серверах. SQLite дає змогу розробникам ефективно керувати даними, водночас скорочуючи та забезпечуючи їхню цілісність, зберігаючи простоту використання та низькі накладні витрати. SQLite повністю підтримує повний набір SQL-команд.

4.3 Асинхронні функції

Далі розглянемо асинхронні функції, які використовуються в нашому коді. Асинхронність в python – це концепція програмування, яка дає змогу виконувати завдання незалежно одне від одного, що в свою чергу пришвидшить відповідь нашого бота. Замість того, щоб блокувати виконання програми в очікуванні, асинхронність дозволяє перемикатися між різними завданнями, не перериваючи виконання основного потоку. У Python асинхронність реалізується з використанням модуля `asyncio`, який надає засоби для створення асинхронних програм. Приклад використання асинхронних функцій показано на рис. 4.8. Асинхронні функції позначаються ключовим словом `async`, а всередині таких функцій використовуються корутини та подієвий цикл для керування асинхронними операціями.

```
3 usages
async def start_learning(update: Update, context: ContextTypes.DEFAULT_TYPE):
    await update.message.reply_text( text: "Ви обрали 'Навчання'. Оберіть категорію:", reply_markup=category_menu_markup)

2 usages
async def handle_learning_message(update: Update, context: ContextTypes.DEFAULT_TYPE):
    text = update.message.text
    user_id = update.message.from_user.id

    if text == 'Програмування на Java':
        await update.message.reply_text( text: "Ви обрали 'Програмування на Java'. Оберіть урок:",
                                         reply_markup=java_learning_menu_markup)
        return True
    elif text == 'Програмування на PHP':
        await update.message.reply_text( text: "Ви обрали 'Програмування на PHP'. Оберіть урок:",
                                        reply_markup=php_learning_menu_markup)
        return True
    elif 'Урок' in text and 'Java' in text:
        lesson_number = int(text.split(' ')[1])
        await start_java_learning(update, context, lesson_number)
        return True
    elif 'Урок' in text and 'PHP' in text:
        lesson_number = int(text.split(' ')[1])
        await start_php_learning(update, context, lesson_number)
        return True
    elif text == 'Наступний крок' and user_id in user_learning_step:
        if user_learning_lesson[user_id][0] == 'Java':
            await continue_java_learning(update, context)
        else:
            await continue_php_learning(update, context)
        return True
    elif text == 'Назад':
        await start_learning(update, context)
        return True
    elif text == 'Головна':
        await go_to_main_menu(update, context)
        return True
    return False
```

Рисунок 4.8 – Використання асинхронних функцій

Далі розглянемо використання функцій для вивчення уроків і їх кроки. На рис. 4.9 показано як проходять уроки по вивченню мови програмування на java. На рис. 4.10 зображено код програми для виконання

запуску уроку і його завершення, якщо урок пройдено до кінця. Аналогічний вигляд буде і у інших уроків із різних мов програмування.

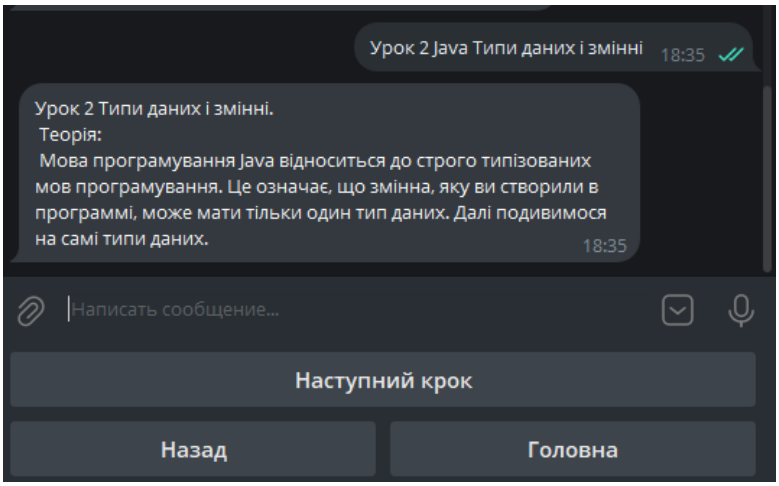

Рисунок 4.9 – Уроки програмування на Java

```
1 usage
async def start_java_learning(update: Update, context: ContextTypes.DEFAULT_TYPE, lesson_number):
    user_id = update.message.from_user.id
    user_learning_step[user_id] = 0
    user_learning_lesson[user_id] = ('Java', lesson_number)
    await java_learning(update, context, lesson_number)

1 usage
async def continue_java_learning(update: Update, context: ContextTypes.DEFAULT_TYPE):
    user_id = update.message.from_user.id
    lesson_number = user_learning_lesson[user_id][1]
    await java_learning(update, context, lesson_number)

2 usages
async def java_learning(update: Update, context: ContextTypes.DEFAULT_TYPE, lesson_number):
    user_id = update.message.from_user.id
    step = user_learning_step[user_id]
    content = get_lesson( category: 'Java', lesson_number, step)
    if content:
        await update.message.reply_text(content, reply_markup=learning_steps_markup)
        user_learning_step[user_id] += 1
    else:
        await update.message.reply_text( text: f"Ви завершили Урок {lesson_number} по Java.",
                                        reply_markup=java_learning_menu_markup)
        user_learning_step[user_id] = 0
```

Рисунок 4.10 – Код програми по проходженню уроків

4.4 Проходження тестів

Наступним пунктом розглянемо проходження тестів. На рисунку 4.11 показано їх проходження і вивід кінцевого результату. На рис. 4.12 продемонстровано код, який запускає проходження тестів і принцип вибору правильної відповіді. Як ми бачимо всі функції, які написані в нашому чат-боті є асинхронні. Аналогічні функції використовуються для всіх наших тестів.

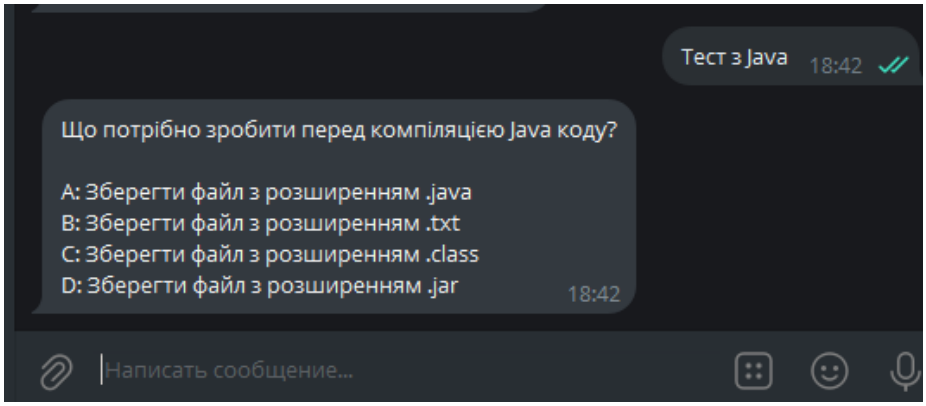

Рисунок 4.11 – Зовнішній вигляд проходження тестів

```python
# Меню вибору категорій тестів
test_category_menu_keyboard = [['Тест з Java', 'Тест з PHP'], ['Назад', 'Головна']]
test_category_menu_markup = ReplyKeyboardMarkup(test_category_menu_keyboard, one_time_keyboard=True, resize_keyboard=True)

# Індекси питань тестів
user_test_step = {}
user_test_category = {}
user_test_score = {}

2 usages
async def start_tests(update: Update, context: ContextTypes.DEFAULT_TYPE):
    await update.message.reply_text( text: "Ви обрали 'Тести'. Оберіть категорію:", reply_markup=test_category_menu_markup)

2 usages
async def handle_tests_message(update: Update, context: ContextTypes.DEFAULT_TYPE):
    text = update.message.text
    user_id = update.message.from_user.id
    if text == 'Тест з Java':
        await start_java_test(update, context)
        return True
    elif text == 'Тест з PHP':
        await start_php_test(update, context)
        return True
    elif text in ['A', 'B', 'C', 'D'] and user_id in user_test_step:
        await check_answer(update, context, text)
        return True
    return False
```

Рисунок 4.12 – Код, який відповідає за початок тестування

ВИСНОВКИ

У процесі аналізу ринку IT розробок у сфері навчання було вирішено створити чат-бота, який допомагав би користувачам освоювати програмування. Він доступний у будь-який час зі смартфона чи ПК. Оскільки додаток повністю безкоштовний, то потреба у платних курсах зникає. Користувач може ознайомитися з мовою програмування, пройти навчання і якщо вона йому не сподобається – обрати іншу і також її освоїти.

У процесі створення додатку було проведено аналіз і компонування теоретичних матеріалів із різних джерел для більш кращого і коректного підбору курсів для вивчення мов програмування.

Після створення чат-бота було заплановано майбутні оновлення та різні рішення для покращення його роботи, а саме – додавання відео та аудіо матеріалів, створення нових курсів на інші мови програмування і тестів для їх закріплення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Мінухін С. В. Методи і моделі проектування на основі сучасних CASE-засобів. Навчальний посібник / С. В. Мінухін, О. М. Беседовський, С. В. Знахур. — Харків: Вид. ХНЕУ, 2008. — 272 с.
2. Кейс: як розробити конструктор Telegram чат-ботів та вирости до 100 000 активних ботів у 2022 [Електронний ресурс]. Режим доступу до ресурсу: https://sendpulse.ua/blog/how-sendpulse-created-a-telegram-chatbot-builder
3. Логування: поняття, вимоги, рівні [Електронний ресурс] – Режим доступу до ресурсу: https://training.qatestlab.com/blog/technical-articles/logging-in-concepts-requirements-levels/
4. Основи Pycharm [Електронний ресурс] – Режим доступу до ресурсу: https://w3schoolsua.github.io/hyperskill/pycharm-basics.html#gsc.tab=0
5. Перша програма на PHP [Електронний ресурс] – Режим доступу до ресурсу:https://programer.in.ua/index.php/prohramuvannia/mova-prohramuvannia-php/173-urok-1-persha-prohrama-na-php
6. Рейтинг мобільних додатків за 2022 рік [Електронний ресурс] – Режим доступу до ресурсу: https://www.kantar.com/ua/inspiration/advertising-media/mobile-app-ranking_april-2022
7. Середовище розробки для Python: що це, які вони бувають і як їх використовувати [Електронний ресурс] – Режим доступу до ресурсу: https://foxminded.ua/seredovyshche-rozrobky-python/
8. Створюємо Telegram бота на Python. Частина 1 [Електронний ресурс] – Режим доступу до ресурсу: https://devzone.org.ua/post/stvoriujemo-telegram-bota-na-python-castina-1
9. Що таке асинхронність у Python і як її правильно застосовувати [Електронний ресурс] – Режим доступу до ресурсу: https://foxminded.ua/asynkhronnist-python/

10. Що таке SQLite, для чого і як його використовувати [Електронний ресурс] – Режим доступу до ресурсу: https://foxminded.ua/prohramuvannia-baz-danykh-na-sqlite/
11. Чат-бот [Електронний ресурс] – Режим доступу до ресурсу: https://sendpulse.ua/support/glossary/chatbot
12. Що таке PHP? [Електронний ресурс] – Режим доступу до ресурсу: https://w3schoolsua.github.io/php/index.html#gsc.tab=0
13. Як створити Telegram-бота за допомогою Python [Електронний ресурс] – Режим доступу до ресурсу: https://www.freecodecamp.org/ukrainian/news/yak-stvoryty-telehram-bota-za-dopomohoyu-python/
14. Heroku та Blazor: як безкоштовно створити інтерактивний додаток [Електронний ресурс] – Режим доступу до ресурсу: https://dou.ua/forums/topic/33927/
15. Introduction to Node.js [Електронний ресурс] – Режим доступу до ресурсу: https://nodejs.org/en/learn/getting-started/introduction-to-nodejs
16. Mark Lutz Learning Python, 5th Edition O'Reilly Media, Incorporated, 2013 1643 p.
17. Python Introduction [Електронний ресурс] – Режим доступу до ресурсу: https://www.w3schools.com/python/python_intro.asp
18. Python-telegram-bot [Електронний ресурс] – Режим доступу до ресурсу: https://pypi.org/project/python-telegram-bot/
19. Telegram Bot API [Електронний ресурс] – Режим доступу до ресурсу: https://core.telegram.org/bots/api
20. What is Pycharm [Електронний ресурс] – Режим доступу до ресурсу: https://intellipaat.com/blog/what-is-pycharm/

ДОДАТОК А

Main.py – Основний файл для запуску чат-бота

```
import logging
from telegram import Update, ReplyKeyboardMarkup
from telegram.ext import Application, CommandHandler, MessageHandler,
ContextTypes, filters
from handlers.learning import handle_learning_message, start_learning,
go_to_main_menu
from handlers.tests import handle_tests_message, start_tests

# Установка логування
logging.basicConfig(
    format='%(asctime)s - %(name)s - %(levelень)s - %(повідомлення)s',
    level=logging.INFO
)

# Головне меню з кнопками
main_menu_keyboard = [['Навчання', 'Тести']]
main_menu_markup = ReplyKeyboardMarkup(main_menu_keyboard,
one_time_keyboard=True, resize_keyboard=True)

# Функція для команди /start
async def start(update: Update, context: ContextTypes.DEFAULT_TYPE):
    await update.message.reply_text("Вітаю! Я створений щоб допомогти тобі
познайомитися із мовами програмування, а також закріпити знання за
допомогою тестів. Оберіть опцію:", reply_markup=main_menu_markup)

# Головна функція для запуску бота
def main():
```

```
    app = Application.builder().token("6374815307:AAFhiWDs6TXfc_zVNUxp7-
fRdur2zip3QHQ").build()
    app.add_handler(CommandHandler("start", start))
    app.add_handler(MessageHandler(filters.TEXT & ~filters.COMMAND,
handle_user_message))

    # Запуск бота
    app.run_polling()

# Функція для обробки текстових повідомлень
async def handle_user_message(update: Update, context:
ContextTypes.DEFAULT_TYPE):
    text = update.message.text

    if text == 'Навчання':
        await start_learning(update, context)
    elif text == 'Тести':
        await start_tests(update, context)
    elif await handle_learning_message(update, context):
        pass
    elif await handle_tests_message(update, context):
        pass
    else:
        await update.message.reply_text("Виберіть опцію з меню.")
if __name__ == '__main__':
    main()

learning.py – файл для обробки уроків
import sqlite3
import logging
```

```
from telegram import Update, ReplyKeyboardMarkup
from telegram.ext import ContextTypes
logger = logging.getLogger(__name__)

# Меню вибору категорій навчання
category_menu_keyboard = [['Програмування на Java', 'Програмування на PHP'],
['Назад', 'Головна']]
category_menu_markup = ReplyKeyboardMarkup(category_menu_keyboard,
one_time_keyboard=True, resize_keyboard=True)

# Підменю навчання з кнопками
java_learning_menu_keyboard = [
    ['Урок 1 Java Знайомство'],
    ['Урок 2 Java Типи даних і змінні'],
    ['Урок 3 Java Оператори'],
    ['Урок 4 Java Оператори вибору'],
    ['Урок 5 Java Цикли'],
    ['Урок 6 Java Класи'],
    ['Назад', 'Головна']
]

php_learning_menu_keyboard = [
    ['Урок 1 PHP Знайомство'],
    ['Урок 2 PHP Оголошення змінної'],
    ['Урок 3 PHP Типи даних'],
    ['Урок 4 PHP Числа'],
    ['Урок 5 PHP Оператори'],
    ['Назад', 'Головна']
]
java_learning_menu_markup =
```

```
ReplyKeyboardMarkup(java_learning_menu_keyboard, one_time_keyboard=True,
resize_keyboard=True)
php_learning_menu_markup =
ReplyKeyboardMarkup(php_learning_menu_keyboard, one_time_keyboard=True,
resize_keyboard=True)

# Кнопки для кроків навчання
learning_steps_keyboard = [['Наступний крок'], ['Назад', 'Головна']]
learning_steps_markup = ReplyKeyboardMarkup(learning_steps_keyboard,
one_time_keyboard=True, resize_keyboard=True)

# Індекси кроків навчання
user_learning_step = {}
user_learning_lesson = {}
def get_lesson(category, lesson_number, step_number):
    conn = sqlite3.connect('lessons.db')
    c = conn.cursor()
    c.execute('SELECT content FROM lessons WHERE category=? AND
lesson_number=? AND step_number=?',
              (category, lesson_number, step_number))
    result = c.fetchone()
    conn.close()
    return result[0] if result else None
async def start_learning(update: Update, context: ContextTypes.DEFAULT_TYPE):
    await update.message.reply_text("Ви обрали 'Навчання'. Оберіть категорію:",
reply_markup=category_menu_markup)
async def handle_learning_message(update: Update, context:
ContextTypes.DEFAULT_TYPE):
    text = update.message.text
    user_id = update.message.from_user.id
```

```
    if text == 'Програмування на Java':
        await update.message.reply_text("Ви обрали 'Програмування на Java'.
Оберіть урок:",
                                        reply_markup=java_learning_menu_markup)
        return True
    elif text == 'Програмування на PHP':
        await update.message.reply_text("Ви обрали 'Програмування на PHP'.
Оберіть урок:",
                                        reply_markup=php_learning_menu_markup)
        return True
    elif 'Урок' in text and 'Java' in text:
        lesson_number = int(text.split(' ')[1])
        await start_java_learning(update, context, lesson_number)
        return True
    elif 'Урок' in text and 'PHP' in text:
        lesson_number = int(text.split(' ')[1])
        await start_php_learning(update, context, lesson_number)
        return True
    elif text == 'Наступний крок' and user_id in user_learning_step:
        if user_learning_lesson[user_id][0] == 'Java':
            await continue_java_learning(update, context)
        else:
            await continue_php_learning(update, context)
        return True
    elif text == 'Назад':
        await start_learning(update, context)
        return True
    elif text == 'Головна':
        await go_to_main_menu(update, context)
```

```
        return True
    return False

async def start_java_learning(update: Update, context:
ContextTypes.DEFAULT_TYPE, lesson_number):
    user_id = update.message.from_user.id
    user_learning_step[user_id] = 0
    user_learning_lesson[user_id] = ('Java', lesson_number)
    await java_learning(update, context, lesson_number)
async def continue_java_learning(update: Update, context:
ContextTypes.DEFAULT_TYPE):
    user_id = update.message.from_user.id
    lesson_number = user_learning_lesson[user_id][1]
    await java_learning(update, context, lesson_number)
async def java_learning(update: Update, context: ContextTypes.DEFAULT_TYPE,
lesson_number):
    user_id = update.message.from_user.id
    step = user_learning_step[user_id]
    content = get_lesson('Java', lesson_number, step)
    if content:
        await update.message.reply_text(content,
reply_markup=learning_steps_markup)
        user_learning_step[user_id] += 1
    else:
        await update.message.reply_text(f"Ви завершили Урок {lesson_number} по
Java.",
                                        reply_markup=java_learning_menu_markup)
        user_learning_step[user_id] = 0
async def start_php_learning(update: Update, context:
ContextTypes.DEFAULT_TYPE, lesson_number):
```

```
    user_id = update.message.from_user.id
    user_learning_step[user_id] = 0
    user_learning_lesson[user_id] = ('PHP', lesson_number)
    await php_learning(update, context, lesson_number)
async def continue_php_learning(update: Update, context:
ContextTypes.DEFAULT_TYPE):
    user_id = update.message.from_user.id
    lesson_number = user_learning_lesson[user_id][1]
    await php_learning(update, context, lesson_number)
async def php_learning(update: Update, context: ContextTypes.DEFAULT_TYPE,
lesson_number):
    user_id = update.message.from_user.id
    step = user_learning_step[user_id]
    content = get_lesson('PHP', lesson_number, step)
    if content:
        await update.message.reply_text(content,
reply_markup=learning_steps_markup)
        user_learning_step[user_id] += 1
    else:
        await update.message.reply_text(f"Ви завершили Урок {lesson_number} по
PHP.",
                                        reply_markup=php_learning_menu_markup)
        user_learning_step[user_id] = 0
async def go_to_main_menu(update: Update, context:
ContextTypes.DEFAULT_TYPE):
    main_menu_keyboard = [['Навчання', 'Тести']]
    main_menu_markup = ReplyKeyboardMarkup(main_menu_keyboard,
one_time_keyboard=True, resize_keyboard=True)
    await update.message.reply_text("Оберіть із меню:",
reply_markup=main_menu_markup)
```

tests.py – файл відповідає за тести

```
import logging
from telegram import Update, ReplyKeyboardMarkup
from telegram.ext import ContextTypes
from data.java_lessons import java_tests
from data.php_lessons import php_tests

logger = logging.getLogger(__name__)

# Меню вибору категорій тестів
test_category_menu_keyboard = [['Тест з Java', 'Тест з PHP'], ['Назад', 'Головна']]
test_category_menu_markup =
ReplyKeyboardMarkup(test_category_menu_keyboard, one_time_keyboard=True,
resize_keyboard=True)

# Індекси питань тестів
user_test_step = {}
user_test_category = {}
user_test_score = {}

async def start_tests(update: Update, context: ContextTypes.DEFAULT_TYPE):
    await update.message.reply_text("Ви обрали 'Тести'. Оберіть категорію:",
reply_markup=test_category_menu_markup)

async def handle_tests_message(update: Update, context:
ContextTypes.DEFAULT_TYPE):
    text = update.message.text
    user_id = update.message.from_user.id
    if text == 'Тест з Java':
```

```
        await start_java_test(update, context)
        return True
    elif text == 'Тест з PHP':
        await start_php_test(update, context)
        return True
    elif text in ['A', 'B', 'C', 'D'] and user_id in user_test_step:
        await check_answer(update, context, text)
        return True
    return False

async def start_java_test(update: Update, context: ContextTypes.DEFAULT_TYPE):
    user_id = update.message.from_user.id
    user_test_step[user_id] = 0
    user_test_category[user_id] = 'java'
    user_test_score[user_id] = 0
    await send_java_question(update, context, user_test_step[user_id])

async def send_java_question(update: Update, context:
ContextTypes.DEFAULT_TYPE, question_index):
    question = java_tests[question_index]
    question_text = question['question']
    options = question['options']
    options_text = "\n".join([f"{chr(65 + i)}: {option}" for i, option in
enumerate(options)])
    await update.message.reply_text(f"{question_text}\n\n{options_text}")

async def check_answer(update: Update, context: ContextTypes.DEFAULT_TYPE,
answer):
    user_id = update.message.from_user.id
    question_index = user_test_step[user_id]
```

```
    correct_answer = java_tests[question_index]['answer']
    selected_option = java_tests[question_index]['options'][ord(answer) - 65]

    if selected_option == correct_answer:
        user_test_score[user_id] += 1

    user_test_step[user_id] += 1

    if user_test_step[user_id] < len(java_tests):
        await send_java_question(update, context, user_test_step[user_id])
    else:
        score = user_test_score[user_id]
        await update.message.reply_text(f"Тест завершено! Ваш результат: {score} з
{len(java_tests)}")
        user_test_step.pop(user_id)
        user_test_category.pop(user_id)
        user_test_score.pop(user_id)
async def start_php_test(update: Update, context: ContextTypes.DEFAULT_TYPE):
    user_id = update.message.from_user.id
    user_test_step[user_id] = 0
    user_test_category[user_id] = 'php'
    user_test_score[user_id] = 0
    await send_php_question(update, context, user_test_step[user_id])

async def send_php_question(update: Update, context:
ContextTypes.DEFAULT_TYPE, question_index):
    question = php_tests[question_index]
    question_text = question['question']
    options = question['options']
```

```
    options_text = "\n".join([f"{chr(65 + i)}: {option}" for i, option in
enumerate(options)])
    await update.message.reply_text(f"{question_text}\n\n{options_text}")

async def check_answer(update: Update, context: ContextTypes.DEFAULT_TYPE,
answer):
    user_id = update.message.from_user.id
    question_index = user_test_step[user_id]
    if user_test_category[user_id] == 'java':
        correct_answer = java_tests[question_index]['answer']
        selected_option = java_tests[question_index]['options'][ord(answer) - 65]
    else:
        correct_answer = php_tests[question_index]['answer']
        selected_option = php_tests[question_index]['options'][ord(answer) - 65]

    if selected_option == correct_answer:
        user_test_score[user_id] += 1

    user_test_step[user_id] += 1

    if user_test_category[user_id] == 'java' and user_test_step[user_id] <
len(java_tests):
        await send_java_question(update, context, user_test_step[user_id])
    elif user_test_category[user_id] == 'php' and user_test_step[user_id] <
len(php_tests):
        await send_php_question(update, context, user_test_step[user_id])
    else:
        score = user_test_score[user_id]
        total_questions = len(java_tests) if user_test_category[user_id] == 'java' else
len(php_tests)
```

```
        await update.message.reply_text(f"Тест завершено! Ваш результат: {score} з
{total_questions}")
        user_test_step.pop(user_id)
        user_test_category.pop(user_id)
        user_test_score.pop(user_id)
```