

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук та технологій
(повне найменування факультету)

Кафедра комп'ютерних систем та мереж
(повне найменування кафедри)

Пояснювальна записка
до дипломного проєкту (роботи)
бакалавра
(ступінь вищої освіти (освітній ступінь))

на тему **РОЗРОБКА КОМП'ЮТЕРНОЇ СИСТЕМИ ІНФОРМАЦІЙНОЇ**
ДОПОМОГИ АБІТУРІЄНТАМ

Виконав: студент 4 курсу, групи КНТ-511сп
спеціальності _____

123 «Комп'ютерна інженерія»
(код і найменування спеціальності)

Освітня програма (спеціалізація) _____
«Комп'ютерна інженерія»

ШМИРКО О. С.

(ПРІЗВИЩЕ та ініціали)

Керівник ХОХЛОВ М.М.
(ПРІЗВИЩЕ та ініціали)

Рецензент КОЗІНА Г.Л.
(ПРІЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування вищого навчального закладу)

Факультет Комп'ютерних наук і технологій
Кафедра «Комп'ютерні системи та мережі»
Ступінь вищої освіти бакалаврський
Спеціальність 123 Комп'ютерна інженерія
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерна інженерія
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ
Завідувач кафедри КУДЕРМЕТОВ Р.К.
“ ” 2024 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

ШМИРКО Олександра Сергійовича
(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка комп'ютерної системи інформаційної допомоги абітурієнтам

керівник проєкту (роботи) ст викладач, ХОХЛОВ Миколай Миколайович
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом вищого навчального закладу від “01” квітня 2024 року № 105

2. Строк подання студентом проєкту (роботи) 20 травня 2024 року
3. Вихідні дані до проєкту (роботи) наявна безперебійно працююча система інформативної допомоги абітурієнтам методом створення телеграм бота

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1) Аналіз технічного завдання;
2) Опис роботи системи;
3) Поетапна розробка системи.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Плакат-1: актуальність використання чат-ботів

Плакат-2: виконання основних команд чат-бота

Плакат-3: вивантаження чат-бота на віртуальний сервер

Креслення-1: алгоритм обробки команд системи

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-3	ХОХЛОВ М. М., ст. викл		
нормоконтроль	ЩЕРБАК Н.В., ст. викл.		

7. Дата видачі завдання 15.04.2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Аналіз наявних методів. Вибір середовища та додаткових бібліотек та фреймворків. Аналіз недоліків та переваг обраних методів.	15.04.2024 р.	
2	Опис структури системи та її робота.	15.04.2024 р.	
3	Опис роботи команд системи.	20.04.2024 р.	
4	Створення оболонки системи.	01.05.2024 р.	
5	Розробка програмної частини системи.	10.05.2024 р.	
6	Тестування та вивантаження на віртуальний сервер.	15.05.2024 р.	
8	Оформлення отриманих результатів у ПЗ.	20.05.2024 р.	
9	Оформлення допоміжного матеріалу.	20.05.2024 р.	

Студент (ка)


 (підпис)
Олександр ШМИРКО

(Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи)


 (підпис)
Микола ХОХЛОВ

(Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

ПЗ: 54с., 44 рис., 18 джерел, 1 додаток

API, BOT, ID, PYTHON, TELEGR, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ,
ПОЯСНЮВАЛЬНА ЗАПИСКА, ШТУЧНИЙ ІНТЕЛЕКТ, КОМАНДИ,
ДОДАТОК, СЕРВЕР, ХОСТИНГ.

Тема роботи: «Розробка комп'ютерної системи інформаційної
допомоги абітурієнтам»

Мета роботи: засвоєння навичок у створенні системи на базі ШІ на
відповідну тему з використанням інструментів web-застосунка telegram з
базою даних.

У проекті проведено опис поетапного аналізу, розробка та побудована
системи інформаційної допомоги абітурієнтам

До отриманих результатів належить розроблена, повністю
працездатна, протестована система допомоги абітурієнтам у телеграмі.

Об'єкт дослідження- системи інформаційної допомоги абітурієнтам.

ЗМІСТ

Вступ.....	7
1 Розділ Аналіз технічного завдання.....	8
1.1 Аналіз бібліотек та допоміжних інструментів	16
1.2 Аналіз хостингу бота	19
2 Розділ Опис роботи системи	22
2.1 Опис головних команд бота	23
2.1 Опис додаткових команд бота	28
3 Поетапна розробка системи	30
3.1 Створення оболонки бота.....	30
3.2 Встановлення необхідного ПЗ	33
3.3 Розробка програмної частини бота.....	35
3.4 Тестування на хостинг бота	44
Висновки	49
Перелік джерел посилання	50
Додаток А	52
Додаток Б	53

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

БД	–	База даних
ПЗ	–	програмне забезпечення
ПК	–	персональний комп'ютер
Токін	–	набору символів для зв'язку між ботом та сервером месенджера
ШІ	–	штучний інтелект
API	–	інтерфейс програмування додатка application programming interface
BOT	–	скорочено від англ. Robot. спеціальна програма, що виконує задані алгоритми
CB	–	скорочення callback, повернення інформації, використовується у написанні синтаксису команд бота
CSS	–	таблиця каскадних стилів
HTTP	–	протокол передачі даних, HyperText Transfer Protocol
HTML	–	мова гіпертекстової розмітки сторінки, HyperText Markup Language
ID	–	індивідуальний ідентифікатор користувача, identifier
IDE	–	інтегроване середовище розробки, Integrated Development Environment
JS	–	мова програмування, Java Script
VS	–	інтегроване середовище розробки Visual Studio
.PY	–	розширення мови програмування Python
PIP	–	система керування пакунками на мові Python, Python Index Pacakage
TG	–	скорочення від англ. telegram

ВСТУП

Через велику кількість документів та не знання де шукати актуальну інформації для вступу до університету було проведено аналіз для створення комп'ютерної системи інформаційної допомоги абітурієнтам. Цей процес автоматизований методом створення чат-бота за допомогою інструментів у web-додатку telegram.

Метою цієї роботи є аналіз комп'ютерної системи інформаційної допомоги абітурієнтам та підбір оптимальних інструментів для розробки системи. Система повинна надавати користувачам можливість зручно та ефективно шукати потрібної інформації на різних пристроях, таких як комп'ютери, смартфони, планшети тощо, що підтримує соцмережу телеграм.

Основні функції системи:

- надання повної інформації для вступу: користувач повинен мати можливість швидко знайти необхідні документи, номери, електронні скриньки тощо;
- зворотній зв'язок: користувач повинен мати змогу задавати питання та отримувати відповіді на свої запитання в цієї системи;
- підтримка різних пристроїв: система має підтримувати різні пристрої

У процесі для розробки інформаційної допомоги абітурієнтам проведено аналіз технологій та інструментів:

- мова програмування: вибір мови програмування впав саме на Python. Оскільки саме ця мова програмування є продуктивною та оптимальною для розробки чат-ботів;
- інтеграція зі сторонніми API: для отримання доступу до бібліотек web-додатка телеграм, та установка додаткової бібліотеки telebot;
- встановлення на хостинг для безперебійної роботи системи;
- застосування внутрішніх інструментів telegram.

1 АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ

Головна задача, це розробити комп'ютерну систему інформаційної допомоги абітурієнтам. Для створення цієї системи будемо розробляти чат-бота на базі соцмережі Telegram.

Чат-бот — це програма, яка імітує справжню розмову з користувачем. За допомогою чат-ботів можна спілкуватися текстовими та аудіо повідомленнями на сайті, в месенджері, мобільному додатку або по телефону.

Чат-боти переважно використовують штучний інтелект для спілкування з користувачами, надаючи їм релевантний контент та актуальні пропозиції. Вони працюють на основі набору інструкцій або використовують машинне навчання. Інструментальні чат-боти мають досить обмежену функціональність.

У більшості випадків вони призначені для того, щоб відповідати на фіксований набір запитань. У провідних компаніях світу чат ботів активно використовують для автоматизації роботи з клієнтами, наприклад відповіді на часті запитання, чи надсилення або заповнення форм та інше. Якщо дивитись на статистику, яка наведена на рисунку 1.1, можна побачити, що ботів активно використовують для багатьох задач, навіть для підвищення комунікації між робітниками компаній.



Рисунок 1.1 – Статистика від фахівців

Рівень інтелекту бота залежить виключно від того, як він запрограмований.

Чат-бот повинен мати в собі наступне:

- зручний інтерфейс та швидкий пошук інформації;
- можливість користувачу задавати питання;
- зворотній зв'язок;
- безперебійна робота бота.

Для розробці потрібна мова програмування, бота можна писати налюбій мові, яку знає розробник. Але, відштовхуючись від того, що бот буде створений для соцмережі телеграм, тому обираємо між Python та JavaScript.

Python - друга за популярністю мова програмування високого рівня у світі. Її використовують для розробки веб-додатків, програмного забезпечення та машинного навчання. Python використовують такі провідні компанії як Google, Instagram, Facebook, Telegram, IBM, NASA, Dropbox та Netflix для вирішення робочих завдань. Розробники цінують мову програмування за простоту вивчення, ефективність та багатоплатформність.

Python - це об'єктно-орієнтована мова програмування, програмне забезпечення якої розробляється у вигляді моделей, які збираються у вигляді пакетів. Тип і структуру кожного об'єкта можна дізнатися під час виконання програми. Для кожного об'єкта можна отримати всю інформацію про його внутрішню структуру. Крім того:

- Python має логічний синтаксис, що робить вихідний код легким для читання та розуміння;
- гнучкість та розширюваність Python дозволяє адаптувати високорівневу логіку та швидко розширювати складні додатки за потреби;
- розробка на Python майже завжди відбувається швидше, ніж на інших мовах програмування;
- Python є інтерпретованою мовою програмування що дає змогу писати і успішно виконати код у будь-якому текстовому файлі на будь-якій платформі;
- Python має величезну спільноту однодумців, тому знайти потрібну

документацію чи інформацію буде не складно.

Однак є і ряд особливостей, які можна вважати недоліками. Це повільність - ця мова програмування універсальна, але повільніша за інші мови, кількість ресурсів, необхідних для роботи з нею, та її "прив'язка" до системних бібліотек.

Крім того, пайтон має велику кількість додаткових бібліотек. Для виконання конкретних завдань використовуються різні бібліотеки. Наприклад, Matplotlib підходить для відображення даних у 2D або 3D графіці. Pandas - для зручної обробки даних. NumPy дозволяє створювати масиви та керувати ними. Requests - для веб-розробки. OpenCV-Python, відкриває можливості обробки зображень для оптимізації систем машинного зору. PyTelegramBot та aiogram – дозволяє створювати ботів на базі соцмережі telegram за допомогою API.

IDE чи інтегроване середовище розробки - це програмне забезпечення, яке надає розробникам інструменти, необхідні для написання, редагування, тестування та налаштування коду. Найпоширенішими IDE для розробки для Python є PyCharm, IDLE, Spyder та Atom.

JavaScript - це мова сценаріїв високого рівня, яка широко використовується для створення інтерактивних додатків і веб-сайтів. JS дозволяє нам додавати анімацію і спливаючі повідомлення, валідувати форми, оновлювати контент і робити інтерфейс інтерфейс і зробити його таким, що реагує на наші дії. JavaScript робить сторінки функціональними Вона має ряд своїх унікальних особливостей. Ось деякі з них:

- інтерпретованість. JavaScript виконується в реальному часі в середовищі браузера без необхідності попередньої компіляції. Розробники можуть вносити зміни в код JavaScript і одразу бачити результати;

- динамічна типізація. Типи змінних JavaScript визначаються автоматично під час виконання коду. Програмістам не потрібно їх попередньо призначати, що спрощує їх роботу;

- численні бібліотеки та фреймворки JavaScript. Готові рішення для різних завдань економлять час IT-фахівців;

- крос-платформенна сумісність; JavaScript-код може працювати на

різних платформах і пристроях;

- автоматичний збір сміття. Полегшує управління пам'яттю, автоматично видаляючи невикористовувані об'єкти;

- переносимість, функціональність та принципи ООП. JavaScript підтримує ключові парадигми програмування. Це дозволяє створювати складні структури даних, покращувати організацію коду, застосовувати функціональні концепції та писати точні процедури для досягнення цілей;

- асинхронність. За допомогою таких механізмів, як обіцянки та `async/await`, JavaScript ефективно обробляє операції, які не потребують негайного завершення, не блокуючи інші завдання;

- вбудована підтримка DOM. Для динамічної взаємодії з елементами веб-сторінки, обробки подій та модифікації;

- навколо JS сформувалася величезна спільнота розробників, які постійно створюють нові курси, навчальні посібники, фреймворки та бібліотеки JavaScript. Тому можна легко знайти відповіді на питання, які виникають в процесі роботи або навчання;

- відкритий вихідний код; JavaScript (як і Python) можна використовувати та змінювати без ліцензійних обмежень.

Яка люба мова програмування js має свої плюси та недоліки, які наведені нижче.

Переваги JavaScript:

- легкість у вивченні, зрозумілий і логічний синтаксис JavaScript робить його гарним вибором для початківців;

- універсальність, JavaScript чудово працює на будь-якому пристрої та платформі. Все, що вам потрібно - це браузер;

- гнучкість і сумісність. JavaScript легко інтегрується з іншими технологіями, такими як HTML і CSS, і підтримує різні парадигми програмування, асинхронну і динамічну верстку;

- багато корисних інструментів. JS-бібліотеки та фреймворки дозволяють вирішувати широкий спектр завдань, від створення веб-сайтів і додатків до розробки серверних рішень;

- популярність; мова програмування JavaScript використовується багатьма компаніями по всьому світу (Amazon, PayPal, Facebook, Twitter, Google, Airbnb і т.д.).

Деякі з недоліків JavaScript включають наступне:

- JS може бути менш продуктивною, ніж інші мови програмування (особливо в обчислювальних завданнях з високим навантаженням);
- JavaScript вразливий до певних кібератак, таких як міжсайтовий скриптинг (XSS). Однак проблеми кібербезпеки не є унікальними для JavaScript;
- у великих, довгострокових проектах динамічна верстка JS може викликати труднощі та збільшити вартість підтримки програмного забезпечення;
- не всі браузері обробляють JavaScript-код однаково, що впливає на сумісність;
- крім того, JavaScript може бути складним для початківців через свою мультипарадигмальну природу та асинхронну роботу. Однак, якщо ви будете наполегливо вчитися, ви зможете все це зрозуміти.

Якщо підвести підсумок Js більше підходить до розробки сайтів та веб-застосунків, а пайтон для розробки бекенду та для роботи з базами даних. Але для обох мов є додаткові бібліотеки які підійдуть для розробки бота. Тому порівнюємо обидві мови у контексті розробки чат бота.

Python та JavaScript - це популярні мови програмування, які використовуються для розробки ПЗ. Якщо порівнювати їх разом, то вони схожі. Наприклад схожість синтаксису, вони мають зрозумілий синтаксис, що дозволяє розробникам легко читати і писати код. Вони обидві є інтерпретованими, тобто компілюються рядок за рядком та обидві мови мають асинхронно працювати, що дає змогу використовувати API Telegram, що над знадобиться у розробці бота . Крім того ці дві мови мають бібліотеки для написання ботів, у пайтона це pyTelegramBot або aiogram, у Js свої «telegraf» або «nodeTelegramBotAPI».

Розглянемо недоліки Python та JavaScript.

Недоліки Python:

- Python може бути менш швидким за інші мови програмування, особливо для великих обсягів обробки даних або при великому навантаженні на сервер;
- асинхронність: Python, зокрема стандартна бібліотека, може бути менш підходящою для асинхронної розробки порівняно з іншими мовами, що може призвести до меншої ефективності у вирішенні завдань, які вимагають асинхронної обробки даних.

Недоліки JavaScript:

- велика кількість вкладених колбеків або обіцянок може призвести до `callback hell`, що робить код складним для розуміння та утримання;
- нестрога типізація.

Існує спосіб подолання деяких недоліків, таких як використання альтернативних інтерпретаторів Python, наприклад використання `PyPy`, для покращення продуктивності, або використання сучасних функцій.

Зважаючи усі за та проти була обрана наступна мова програмування, а саме Python. Все ж для розробки бота більше підійде пайтон, оскільки це є популярною мовою для написання ботів, має велику базу документацій як на саму мову так і на її фреймворки. Тому обираємо Python. Згідно статистики майже 50% ботів пишуться на Python, 21% на JavaScript, 12% на Java та останні на інших мовах, таких як PHP, Ruby, C# та інші.

Для написання коду, потрібно середовище програмування. Оскільки остаточний вибір з мови програмування був Python, тому середовище треба підібрати під відповідну мову.

З популярних IDE для пайтон є PyCharm, IDLE, Spyder та Atom, але ще є редактор коду VS Code, за який зараз поговоримо.

VS Code — це спрощений, але потужний редактор вихідного коду, який запускається на комп'ютері й доступний для Windows, macOS і Linux. Він підтримує JavaScript, TypeScript і Node.js і має багату екосистему розширень для інших мов (наприклад, C++, C#, Java, Python, PHP і Go) і середовищ виконання (наприклад, .NET і Unity). VS Code надає додаткові можливості

завдяки розширенням. Розширення VS Code можуть вносити додаткові функції до наявного набору. Після випуску цієї функції з'явилася можливість використовувати розширення VS Code для роботи із порталами Power Apps.

Крім того, можна встановити майже будь яке ПЗ та додаткові бібліотеки шляхом завантаження пакетів через консоль або у самому додатку, це можна побачити на рисунку 1.2.

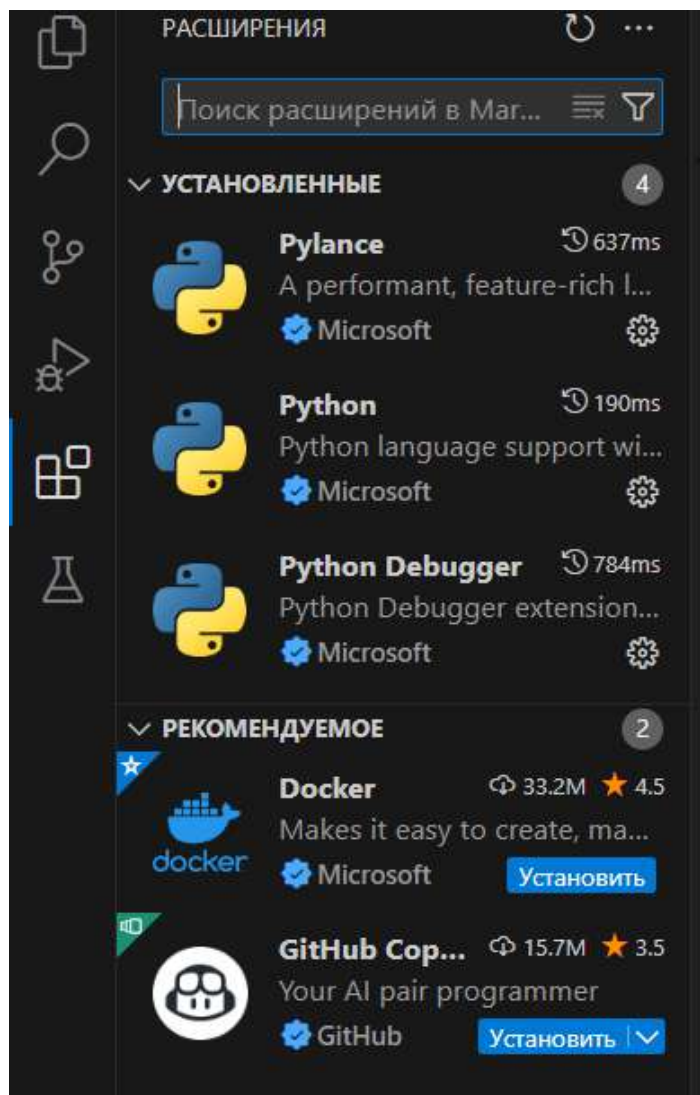


Рисунок 1.2 – Меню завантаження додаткового ПЗ

У самому додатку завантаження проходить буквально натискаючи кнопку «Встановити». Це IDE підтримує більше 50 мов програмування, тому це середовище дуже зручне для програмування на будь-якій мові.

Аналоги цього середовища є PyCharm, Spyder та Atom. Якщо їх порівнювати, то PyCharm: підтримує інші мови програмування, окрім Python.

Професійна версія також підтримує такі мови, як JavaScript, HTML та CSS. Він також має потужні інструменти для редагування коду та автозавершення для підвищення продуктивності програмістів. Крім того, його можна інтегрувати з повноцінними системами управління проектами та системами контролю версій, такими як Git, що нам знадобиться для вивантаження на хостинги. Найголовніше, він має добре організований інтерфейс з багатьма корисними інструментами та панелями.

Стосовно Spyder, він зосереджений переважно на наукових обчисленнях та аналізі даних, але також підтримує інші мови, такі як R. Spyder теж має хороші інструменти для редагування коду, але більше зосереджений на функціях аналізу даних. Він містить у собі базові функції управління проектами та підтримує інтеграцію з Git'ом, але менш просунутий, ніж PyCharm. Має гарний інтерфейс, спеціально розроблений для наукового програмування та аналізу даних.

Atom: може бути налаштований для роботи з багатьма мовами програмування, включаючи Python, але не має вбудованої функціональності для наукових розрахунків, таких як Spyder. Існує ряд плагінів, які можна налаштувати для покращення редагування коду та автозавершення. Для Atom потрібні додаткові плагіни для повноцінного управління проектами та інтеграції з Git'ом. Плюси: має дуже гнучкий і налаштовуваний інтерфейс, який можна налаштувати за допомогою плагінів, та велика база цих плагінів.

Якщо обирати між цими трьома середовищами, то краще обрати PyCharm. Він найбільш загально визнаний серед них, особливо для професійної розробки. Тому залишається вибір між PyCharm та VS Code. На відміну від PyCharm, VS Code має невеликий об'єм, який потрібен при встановленні додатка, він становить близько 500Mb, коли PyCharm може важити від 800 МБ до 1,5 ГБ в залежності від версії середовища. Але Пайчарм є безпосередньо IDE, а не редактором кода, тому встановлювати якійсь додаткові бібліотеки не потрібно, окрім бібліотеки для розробки бота, для VS Code потрібно встановити окрім плагінів на пайтон безпосередньо саму мову. Тому плюс PyCharm.

Але враховуючи, що чат-бот, якого будемо розроблювати, не потребує

дуже потужного інструменту, такий як PyCharm, я буду використовувати VS Code. Здебільшого, через те, що він займає мало місця, має велику базу плагінів для редагування тексту та я маю більше досвіду у роботі з цим середовищем, хоча PyCharm також добре підійде для цих завдань. А зараз перейдемо до бібліотеки, які потрібні для написання бота.

1.1 Аналіз бібліотек та допоміжних інструментів

Telegram - популярний месенджер, який дозволяє користувачам обмінюватися повідомленнями, відео, музикою та файлами. Блогери використовують його для публікації контенту, а компанії - для продажу своїх товарів та послуг. Крім того, Telegram дозволяє користувачам створювати власних ботів. Боти - це програми в месенджері, які автоматизують спілкування.

Кожен бот має унікальне ім'я, і його можна легко знайти в пошуковому рядку Telegram. Ботів також можна додавати до чатів і каналів, щоб вони виконували роль модераторів та адміністраторів. Віднедавна боти Telegram можуть ставати повноцінними веб-додатками з власним користувацьким інтерфейсом.

Можливості ботів обмежені лише фантазією та навичками їхніх розробників, тому компанії активно використовують ботів у своїй повсякденній діяльності.

Боти можуть використовуватися в будь-яких сферах. Наприклад, телеграм-боти можуть автоматизувати запис до салонів і лікарів, зменшуючи навантаження на менеджерів. Саме головне, боти можуть приймати заявки в будь-який час доби, включно з вихідними та державними святами, і переносити інформацію про зустрічі до календаря компанії. Мій бот буде на кшталт електронного довідника у телеграмі, з можливістю зворотнього зв'язку з фахівцями по питанням вступу.

Програмування необхідне, якщо сценарій використання бота нестандартний або якщо важливо мати повний контроль над поведінкою бота та обробкою даних, оскільки Telegram надає добре документований і відкритий API, що дозволяє розробляти ботів практично будь-якою мовою програмування. Хоча це можливо, але зручніше використовувати готові бібліотеки.

Оскільки розробники Telegram активно розвивають месенджер, важливо, щоб бібліотека оновлювалася якомога частіше і підтримувала всі нововведення сервісу. Боти здебільшого пишуться на Python через простоту та зручність мови.

Бібліотеки для ботів Telegram для Python включають pyTelegramBot або її аналог Aiogram.

Aiogram - це комплексна бібліотека з відмінною документацією, підтримкою та активною спільнотою. Існує велика спільнота, які готові допомогти з будь-якими труднощами і відповісти на питання. Бібліотека aiogram підтримує асинхронні автомати і призначена для розробки складних роботів, які використовуються багатьма користувачами щодня. Наразі існує дві активні версії aiogram, кращим вибором є версія 3.0, яка все ще знаходиться в бета-версії, але стабільно працює. Бібліотека вимагає поглиблених знань Python та досвіду програмування і може бути складною для початківців.

Telebot (pyTelegramBotAPI) - бібліотека для простих ботів з невеликою кількістю користувачів, яка підтримує основні методи Telegram API. telebot організовує асинхронність і складніший в управлінні великими навантаженнями, але для новачків він більш простіше і зручніше.

pyTelegramBot - надає простий та зручний спосіб створювати ботів для Telegram. Вона забезпечує доступ за допомоги Telegram Bot API через інтерфейс Python, що дозволяє легко створювати та управляти ботом.

Якщо порівнювати ці дві бібліотеки, то кращою буде aiogram. Але, як і з PyCharm, це є потужний інструмент для розробки, но все ж краще використовувати більш простіший спосіб розробки для оптимізації часу для розробки бота. Та по-друге, працювати з telebot зручніше, через велику

кількість офіційної документації від першоджерела, вона є актуальною, оскільки телеграм підтримує та оновлює цю бібліотеку.

Крім бібліотеки потрібно окремо зробити бота в BotFather. Він є у телеграмі під user_name @BotFather(див. рис. 1.3).



Рисунок 1.3 – BotFather у телеграм

Bot Father - є офіційним ботом Telegram, який дозволяє користувачу створювати своїх власних ботів шляхом надання їм унікального ідентифікатору (токену), які потім будемо використовувати для налаштування та управління ботами через API Telegram. Крім того BotFather надає можливість змінювати налаштування ботів, додавати команди, налаштовувати повідомлення-привітання, поставити аватар боту та багато іншого(див. рис. 1.4.)

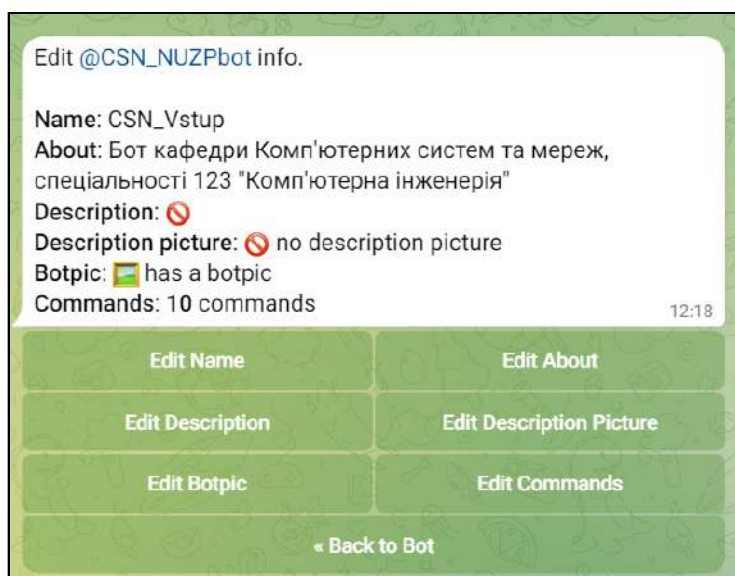


Рисунок 1.4 – Налаштування для вже створеного бота

Також має ряд функцій для створення ігор, додатків та іншого з

використанням телеграм. Але ми його будемо використовувати безпосередньо для створення бота. Головні команди, якими будемо користуватися можна побачити нижче у таблиці 1.1. Майже всі команди з переліку будуть використані для створення та редагування бота.

Таблиця 1.1 – Головні команди для створення та редагування бота

Команди	Значення
/newbot	Створює нового бота
/mybot	Дає змогу змінювати вже існуючих ботів
/token	Генерує унікальний токен аунтифікації бота
/setname	Дозволяє встановити ім'я боту
/setabouttext	Дозволяє встановити поле «Про себе»
/setdescription	Дозволяє встановити опис бота
/setuserpic	Дозволяє встановити аватар боту
/setcommands	Дозволяє встановити команди боту
/deletebot	Видалення бота

Мова програмування та додаткові бібліотеки для програмування будемо встановлювати за допомогою терміналу.

1.2 Аналіз хостингу бота

Для хостинг бота я обрав Replit. Дуже зручний хостинг, не потрібно щось додаткове завантажувати, чи встановлювати. Можна зареєструватись через такі платформи як GitHub, Google та інші. Має простий інтерфейс та головне він не дорогий.

Replit — це інтегроване онлайн-середовище розробки (IDE), яке можна використовувати з різними мовами програмування, включаючи JavaScript, Python, Go, C++, Node.js, Rust та будь-яку іншу мову, доступну в пакувальнику

Nix. Він використовує компонент редактора CodeMirror 6 той самий компонент редактора, який використовується на інших великих веб-сайтах, таких як CodePen.

Ключовою особливістю Replit є спільне програмування, можливість ділитися Repl з одним або декількома іншими користувачами та бачити зміни у файлах у реальному часі, надсилати повідомлення один одному та разом налагоджувати код.

Replit підтримує більше 50 мов програмування, але станом на 23 лютого 2023 року Replit використовує Nix у всіх Repl, що означає, що користувачі мають доступ до всієї бази даних пакетів Nixpkgs і можуть використовувати будь-яку мову програмування всередині. Нові репліки можна створювати за допомогою шаблонів офіційних мов або за допомогою конфігурації користувача Nix. Користувачі можуть налаштувати будь-що: від протоколу мовного сервера до підтримки відладчика для Repl

Крім вище перелічених плюсів хочу додати, як вигрузити на хостинг бота. Для цього потрібно лише встановити потрібні бібліотеки у консолі та просто у поле додати наш код. Усе бот буде готовий до запуску.

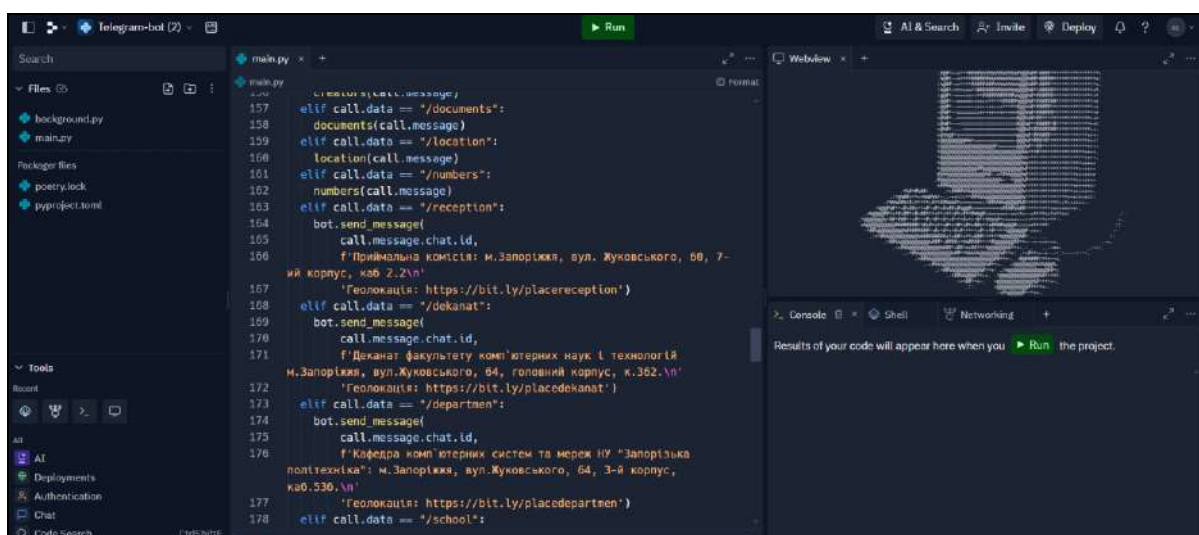


Рисунок 1.5 – Робоче меню Replit

Серед інших аналогів для хостингу є Hירוku.

Heroku - це платформа типу PaaS, яка дозволяє розгортати, керувати та масштабувати веб-додатки та сервіси безпосередньо з ресурсів хмарних

обчислень. З головних плюсів вона має безкоштовних термін роботи. Близько 15 днів, з наданням міцного WPS для розміщення бота, легкий деплой бота через Git. Треба декілька кроків для деплоя, зайти у Heroku через свій обліковий запис, та за допомоги Git завантажити на хостинг. Але з цього «безкоштовного» хостингу витікає наступне, він працює протягом 15 днів, що не підходить під наші умови. Якщо брати платний хостинг мінімальна ціна від 5\$ у місяць. А Replit надає платний хостинг лише за 1\$ у місяць від кількості використання ресурсі хоста.

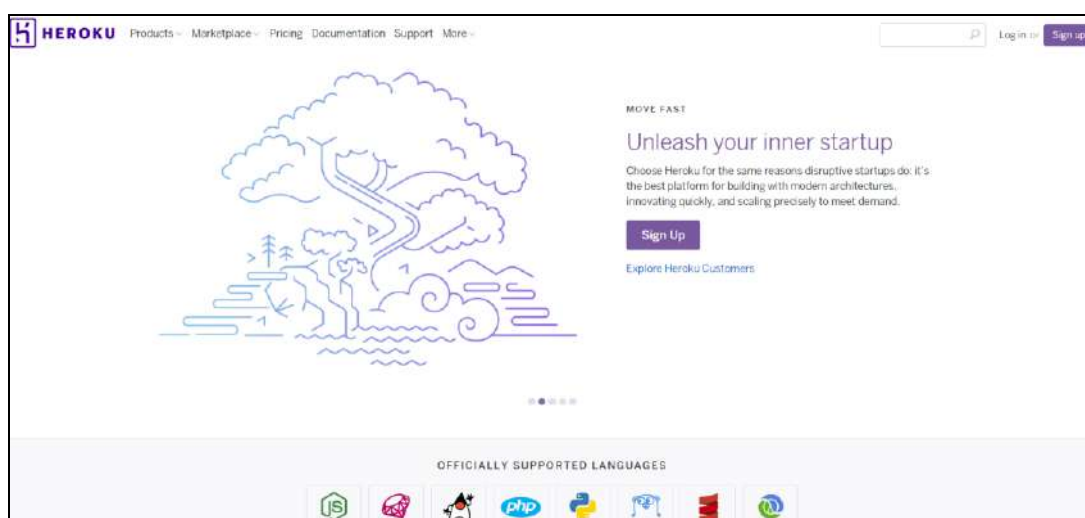


Рисунок 1.6 – Робоче поле Heroku

2 ПРОЕКТУВАННЯ РОБОТИ СИСТЕМИ

Як було описано висче, бот – це штучний інтелект, який виконує запрограмований йому алгоритм, тому треба продумати логіку бота та його команди.

Логіка буде наступною: спочатку користувач активує бота, за замовченням це команда /start, бот з ним вітається та пропонує розповісти про себе та його можливості, далі бот пропонує користувачу функціонал бота у вигляді кнопкового меню, яке буде реалізовано за допомогою бібліотек.

Далі користувач вже сам обирає, що саме йому треба, чи задати питання, чи подивитися список документів до вступу чи актуальні контактні данні кафедри, приймальної комісії чи деканату. Схему роботи бота можна побачити на рисунку 2.1

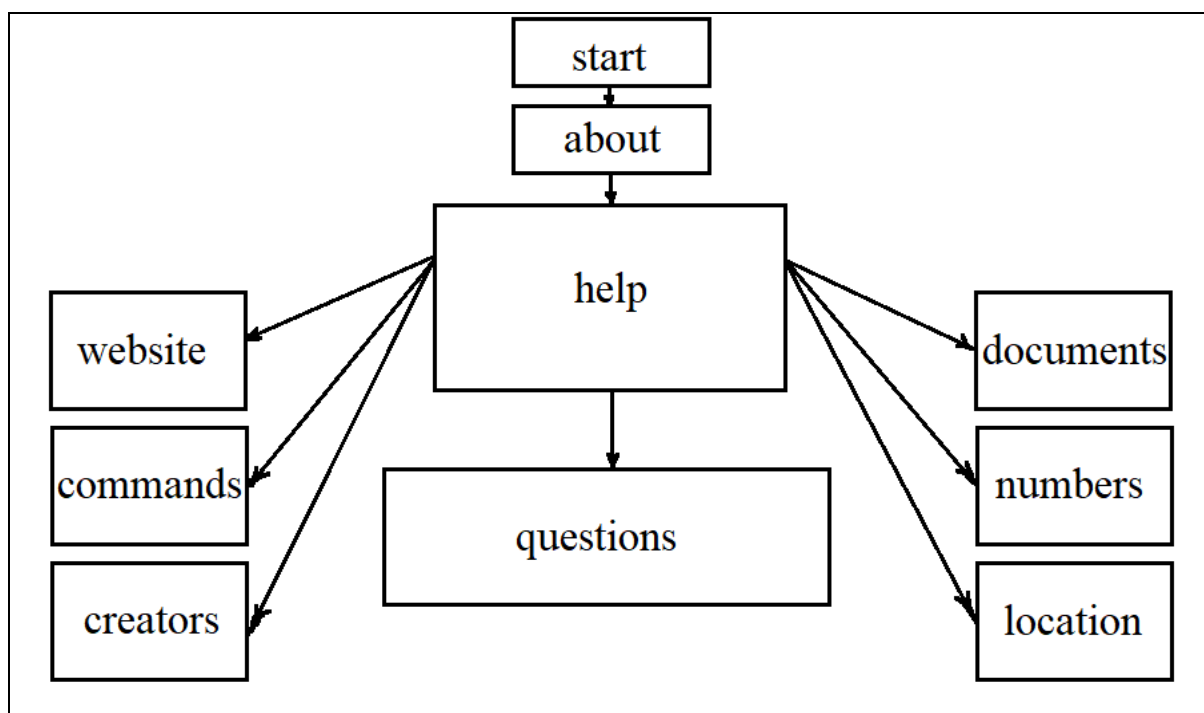


Рисунок 2.1 – Схема логіки роботи бота

2.1 Опис головних команд бота

Загально, бот буде використовувати 18 команд, з них 10 обов'язкових, з якими буде взаємодіяти користувач, 7 додаткових та одна текстова команда, яка буде перевіряти на правильне написання команд.

До основних команд відносимо усі команди з функціоналу бота, саму команду виводу функціоналу бота, загальні та додаткова команди, а саме: /start та /about. Робота усіх команд можна побачити на рисунках 2.2-2.14

Команда /start є обов'язковою, оскільки з неї починається діалог з користувачем, вона має містити текст вітання з користувачем, та запропонувати далі іншу команду, щоб користувач дізнався трішки про цього бота.

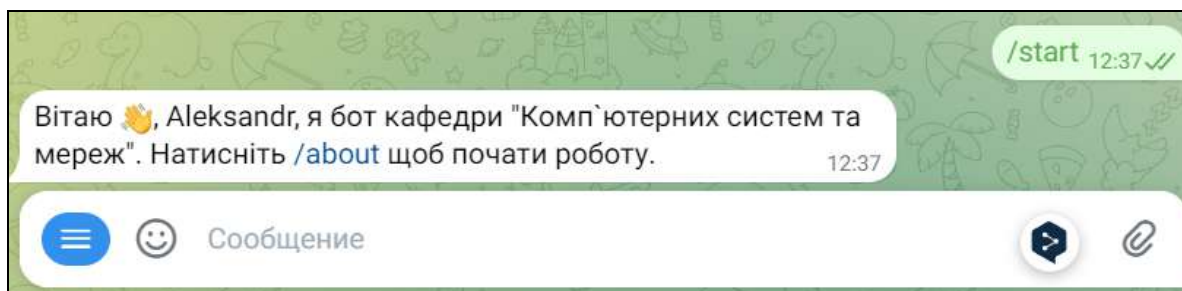


Рисунок 2.2 – Виконання команди /start

Далі йде команда /about, вона знайомить користувача з кафедрою та пропонує переглянути функціонал бота, де вже користувач може обрати, що йому потрібно.

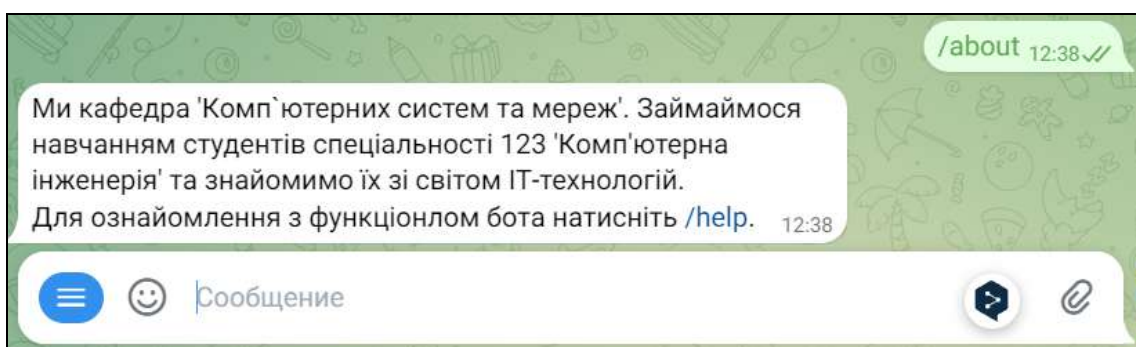


Рисунок 2.3 – Виконання команди /about

Команда /help – це головна команда бота, яка показує усю необхідну інформацію, яка знадобиться для вступників. Меню створено у вигляді кнопочового меню, до яких прив'язані команди. Воно містить в собі сайт кафедри, перелік команд, творців бота, перелік документів до вступу, номери усіх установ, які знадобляться вступнику та локації, де знаходяться та чи інша установа. Так користувач може задати питання боту, на яке згодом відповідь фахівець зі вступу особисто користувачу, або через бота.

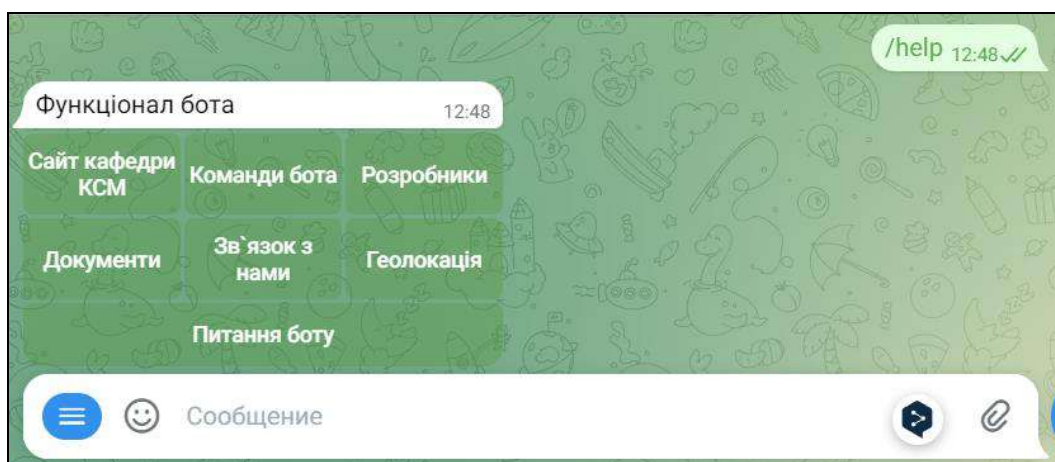


Рисунок 2.4 - Виконання команди /help

Далі пройдемося по командам у головному меню бота та почнемо з команди /website або «Сайт кафедри», ця функція показує кнопку, до якої прив'язане посилання на веб-сайт кафедри.



Рисунок 2.5 - Виконання команди /website

Команда `/command` або «Команди бота» використовується для переліку усіх головних команд бота, ці ж самі команди можна побачити у правому нижньому куті як плаваюче меню.

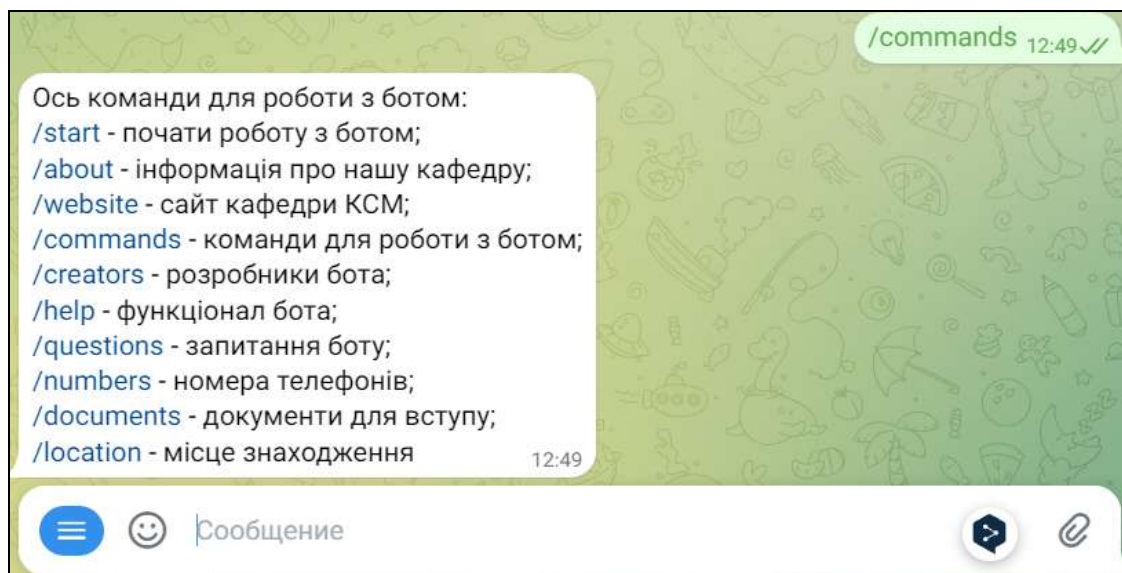


Рисунок 2.6 - Виконання команди `/command`

Команда `/creators` або «Розробники» показує розробника бота здебільшого для зворотнього зв'язку між користувачем та розробником для рішення технічних питань.



Рисунок 2.7 - Виконання команди `/creators`

`/numbers` або «Зв'язок з нами» – з назви можна зрозуміти, ця команда показує список номерів та електронних скриньок навчальних установ, кафедри, деканату та приймальної комісії.

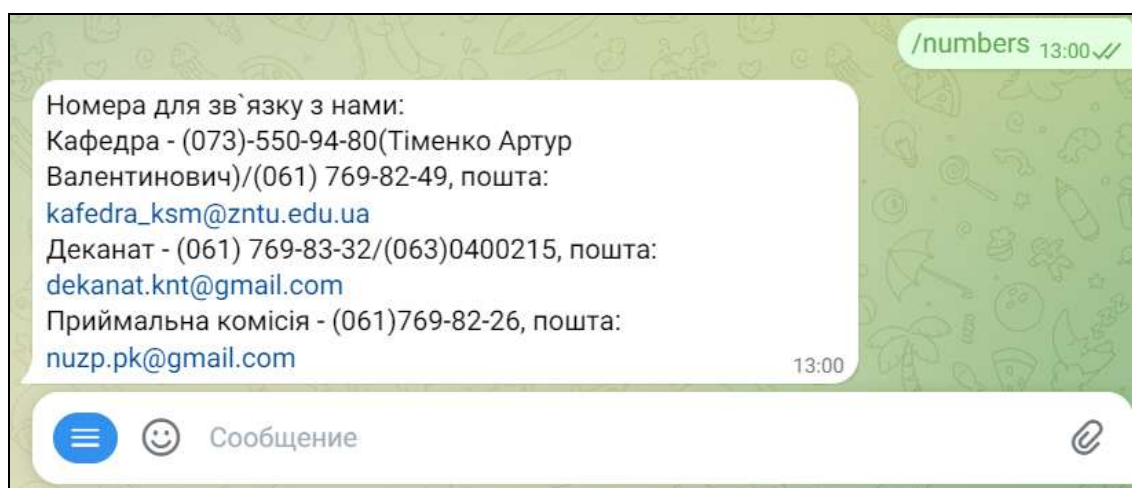


Рисунок 2.8 - Виконання команди /numbers

При визові команди /document або «Документи», користувач побачить меню з 3 кнопками. Він обирає, на базі чого буде поступати. Після того, як користувач обере куди йому поступати, він отримає перелік усіх документів шляхом натискання на кнопку, наприклад(див рис. 2.9) якщо він натисне «Після коледжу» він отримує перелік документів, які буде потрібні для вступу для отримання ступеню бакалавр на базі коледжу.

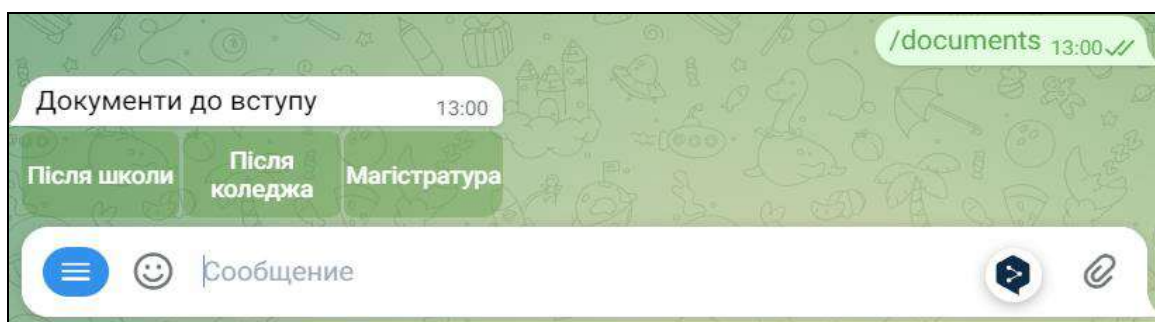


Рисунок 2.9 - Виконання команди /document

Команда /location «Геолокація» схожа на минулу команду /documents, але вона пропонує 3 кнопки в вибором освітніх установ, які знадобляться вступнику, а саме: кафедра, приймальна комісія, деканат.

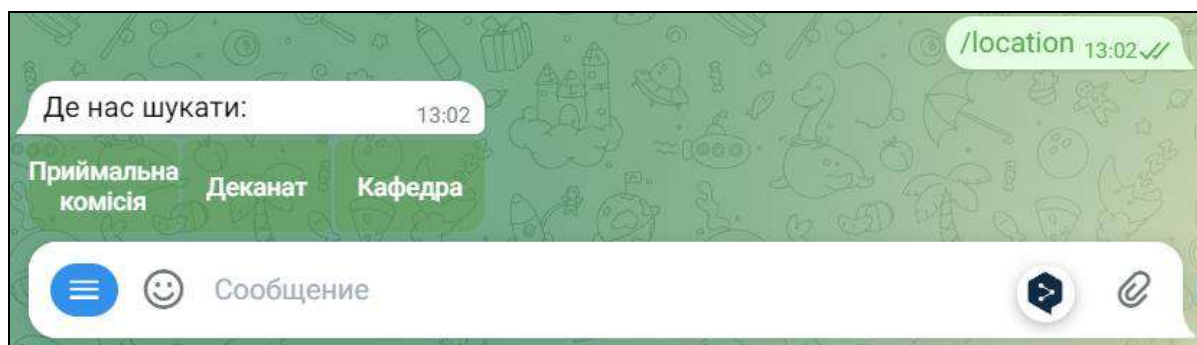


Рисунок 2.10 - Виконання команди /location

Та остання, одна з головних команд, команда /questions або «Питання боту» – ця команда потрібна для того, щоб користувач міг задати питання боту. Після виклику команди, користувач побачить меню з двох кнопок «Задати питання» та «Скасувати». Якщо користувач вибере «Задати питання» він зможе написати прямо у чат повідомлення, яке відправиться до фахівців, на яке згодом дадуть відповідь. Якщо користувач вибере «Скасувати», він скасує запит.

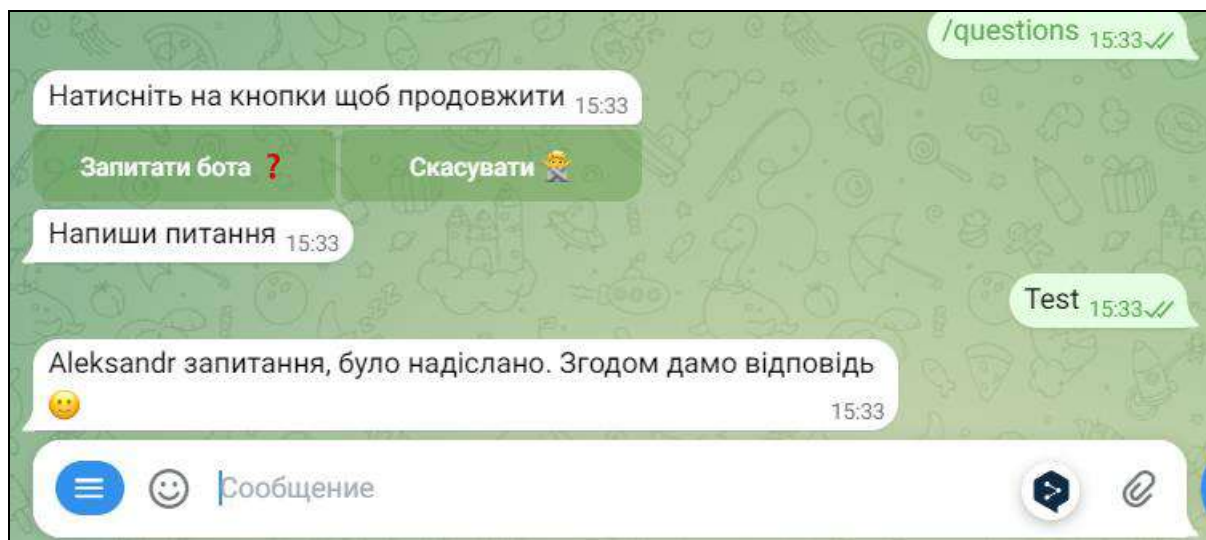


Рисунок 2.11 – Робота команди /questions

2.2 Опис додаткових команд бота

Перейдемо до команд, які не будуть видимі для користувача але вони використовуються ботом, здебільшого ці команди це кнопки, які повертають інформацію методом `callback.data`. Більш детально про метод буде описано нижче.

Почнемо, команда `/cancel`, ця команда потрібна для роботи кнопки «Скасувати» у команді `/questions`. Вони виконує функцію скасування запиту користувача.

Далі буде 3 команди які схожі, це команда `/school`, `/colleg`, `/magistr`. або «Після школи», «Після коледжу», «Магістратура». Так само, ці команди не бачить користувач, але вони використовуються у команді `/documents`. Саме в цих командах міститься інформація о списку документів до вступу, кожна команда свій список. Приклад на рис 2.12

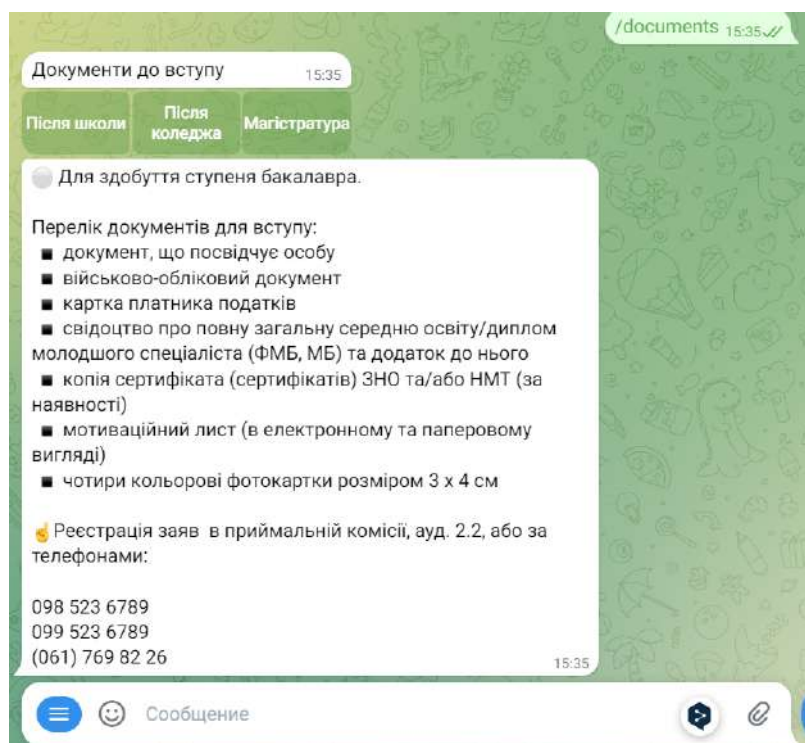


Рисунок 2.12 – Робота додатковою команди `/colleg`

Ще 3 команди, які схожі, команда `/departmen`, `/reception`, `/dekanat`. або «Кафедра», «Приймальна комісія», «Деканат» У цих командах міститься

інформація про місцезнаходження навч. установ, кафедри, приймальної комісії та деканату. Вони використовуються у команді /location. Та мають наступний вигляд див рис. 2. 13



Рисунок 2.13 – робота додатковою командою /dekanat

Та остання навіть не команда, а функція, яка перевірить введений текст користувача. Якщо це команда, то вона перевіряє чи є ця команда зі списку команд, якщо так то вона виконає команду, якщо напише, що даної команди не існує то вона запропонує ознайомитись з командами.

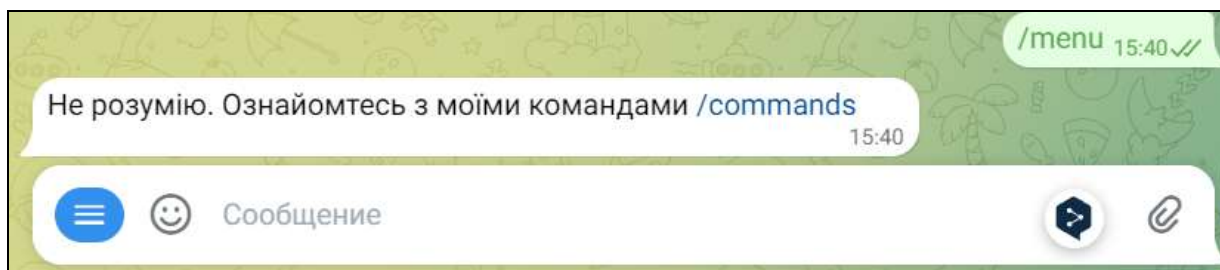


Рисунок 2.14 – робота менеджера команд

3 ПОЕТАПНА РОЗРОБКА СИСТЕМИ

Розробка буде мати в собі три етапи. На першому етапі створимо оболонку бота у BotFather, проведемо налагодження зовнішнього виду бота та отримаємо токін, який буде використаний надалі у розробці бота. На другому етапі, треба встановити усі необхідні середовища, додатки, фреймворки та бібліотеки для подальшої розробки. Третій етап, це розробка команд, які були описані у другому розділі ПЗ. Та останній етап – це тестування та вивантаження бота на відповідних сервер.

3.1 Створення оболонки бота

Перше, що необхідно зробити, це створити бота у BotFather. Для цього знаходимо його за відповідним username: «@BotFather». Далі виконуємо наступні дії:

- виконуємо команду /newbot;
- водимо ім'я бота. У моєму випадку це CSN_Vstup;
- далі потрібно ввести username бота, @CSN_NUZPbot. Звертаємо увагу, обов'язково повинен бути префікс bot.

Після цих кроків була створена оболонка. Нам надали посилання на нього, отримали токін бота та отримали посилання на документацію по експлуатації API telegram. Токін треба зберегти, він знадобиться для подальшої розробки.

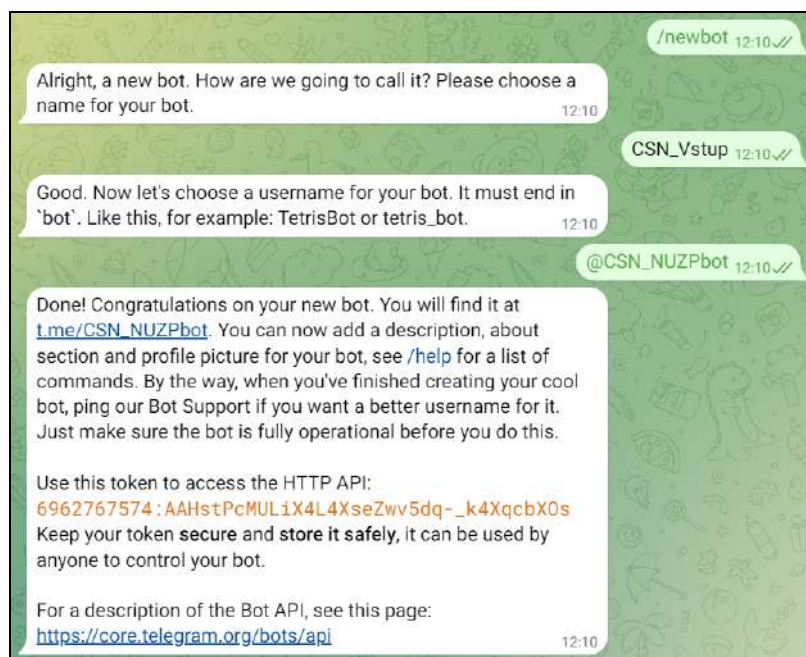


Рисунок 3.1 – Дії для створення оболонки бота

Далі потрібно попрацювати із зовнішнім виглядом бота. Тобто, зробити аватар, та додати наповнення боту. Для цього виконуємо команду `/mybots` та обираємо `@CSN_NUZPbot`. З'явиться кнопкова форма, де ми можемо редагувати бота, як нам потрібно. Потрібно додати аватар, опис боту та команди. Для встановлення аватара викликаємо команду `/setuserpic` або обираємо кнопку у формі з назвою `Edit Botpic`. Бот пришле повідомлення, що потрібно завантажити фотографію у чат. Виконуємо цю дію, після чого у нас бот має власну фотографію.



Рисунок 3.2 – Аватар бота, який буде відображатись у користувача

Для заповнення розділу «Про себе» або «about» виконуємо наступні дії, які наведені нижче:

Далі виконуємо наступні дії:

- виконуємо команду /setabouttext;
- водимо у чат з ботом «Бот кафедри Комп'ютерних систем та мереж, спеціальності 123 "Комп'ютерна інженерія»;

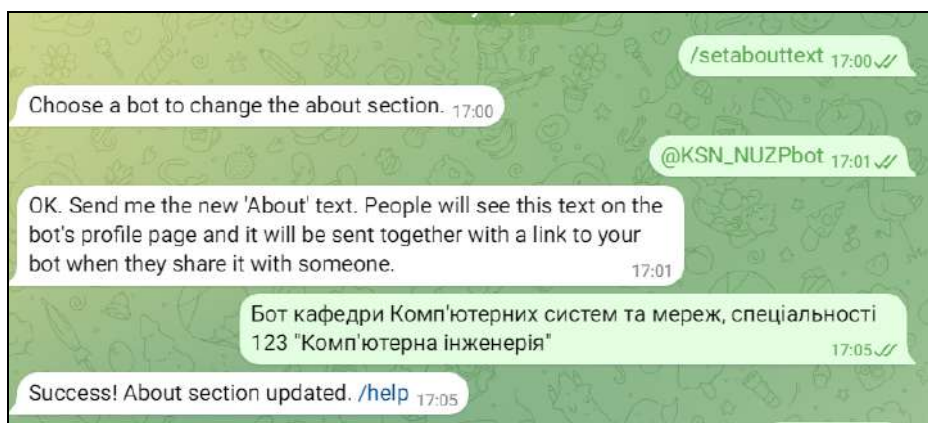


Рисунок 3.3 – Встановлення тексту «about»

Так само додаємо команди, які будуть у боті, використовуючи /setcommands. Виконуємо наступні дії:

- виконуємо команду /etcommands;
- водимо у чат з ботом перелік команд, які будуть відображатись у спливаючому вікні;

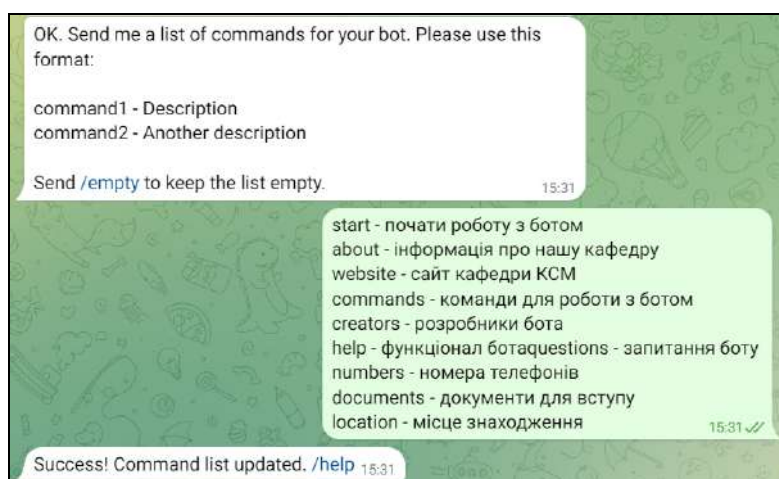


Рисунок 3.4 – Додавання списку команд

Після налагодження оболонки, ми маємо бота у телеграм. Після цього, перейдемо до другого етапу розробки, а саме встановлення середовища та додаткових бібліотек.

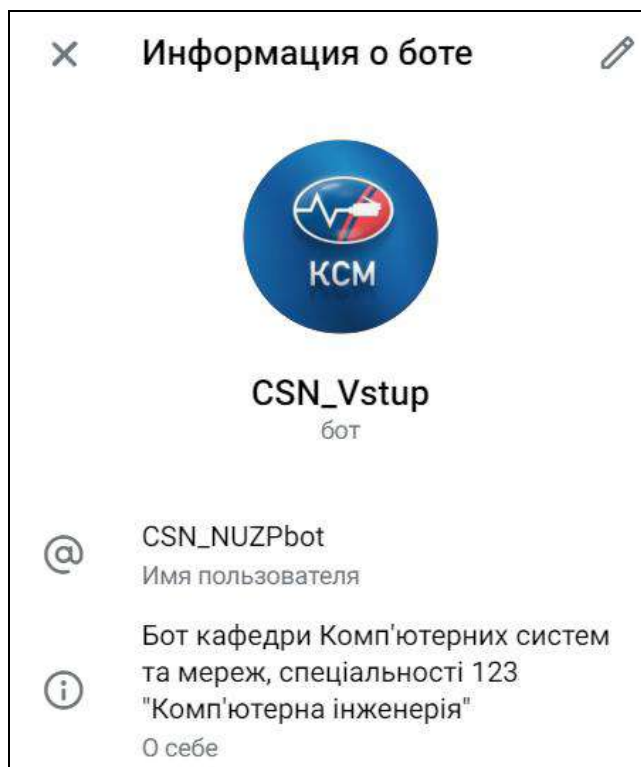


Рисунок 3.5 -Зовнішній вигляд бота у телеграм

3.2 Встановлення необхідного ПЗ

Для початку роботи потрібно середовище розробки з мовою програмування Python. Як зазначено у першому розділі використовувати будемо VS CODE. Це IDE можна скачати безкоштовно з офіційного сайту. Проводимо швидку інсталяцію та додаток встановлюється. Далі потрібно завантажити мову програмування Python та пакетний менеджер `pip` для цієї мови. Все завантажується також з офіційного сайту [Python.org](https://python.org).

Крім цих дій, потрібно налагодити наше середовище, тому запускаємо його, переходимо у вкладнику «Розширення» та у пошуку вводимо Python. Потрібно завантажити 3 розширення саму мову, додаткові до неї бібліотеки та

дебагер, завдяки якому буде виконуватись код. Ці розширення наведені на рисунку 3.6.

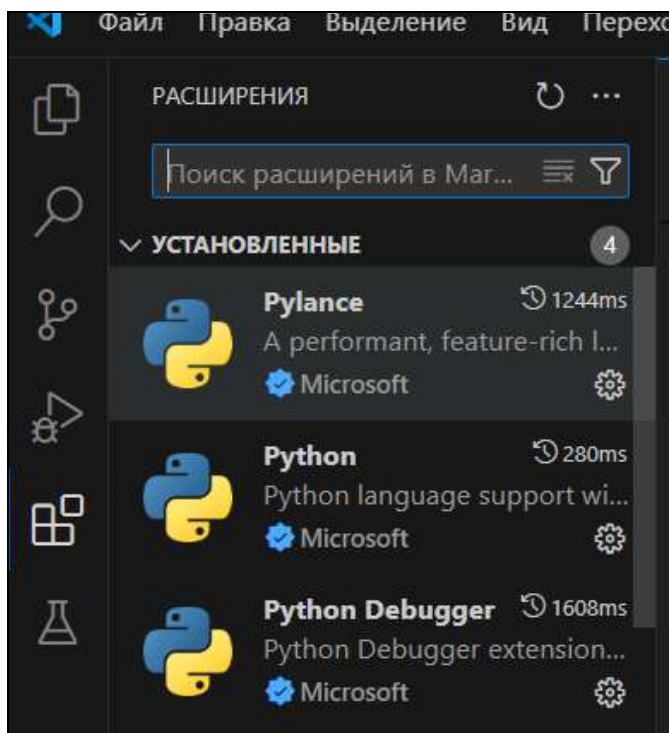


Рисунок 3.6 – Додаткові розширення у середовищі

Остання дія, щоб приступити до розробки бота потрібно встановити бібліотеку Telebot. Створюємо прямо у IDE файл main.py. Зверніть увагу, обов'язково потрібне розширення «.py». Далі, для становлення бібліотеки обираємо «Terminal» та у полі вводимо наступну команду `pip install pyTelegramBotAPI` та натискаємо Enter.

```
PS D:\Bot> pip install pyTelegramBotAPI
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: pyTelegramBotAPI in c:\users\sireo\appdata\local\packages\pythonsoftwarefoundation.python.3.12_qbz5n2kfra8p0\localcac
he\local-packages\python312\site-packages (4.16.1)
Requirement already satisfied: requests in c:\users\sireo\appdata\local\packages\pythonsoftwarefoundation.python.3.12_qbz5n2kfra8p0\localcac
he\local-packages\python312\site-packages (from pyTelegramBotAPI) (2.31.0)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\sireo\appdata\local\packages\pythonsoftwarefoundation.python.3.12_qbz5n2kfra8p0\
local-packages\python312\site-packages (from requests->pyTelegramBotAPI) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in c:\users\sireo\appdata\local\packages\pythonsoftwarefoundation.python.3.12_qbz5n2kfra8p0\localcac
he\local-packages\python312\site-packages (from requests->pyTelegramBotAPI) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\sireo\appdata\local\packages\pythonsoftwarefoundation.python.3.12_qbz5n2kfra8p0\localc
ache\local-packages\python312\site-packages (from requests->pyTelegramBotAPI) (2.2.1)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\sireo\appdata\local\packages\pythonsoftwarefoundation.python.3.12_qbz5n2kfra8p0\localc
ache\local-packages\python312\site-packages (from requests->pyTelegramBotAPI) (2024.2.2)
PS D:\Bot>
```

Рисунок 3.7 – Встановлення бібліотеки

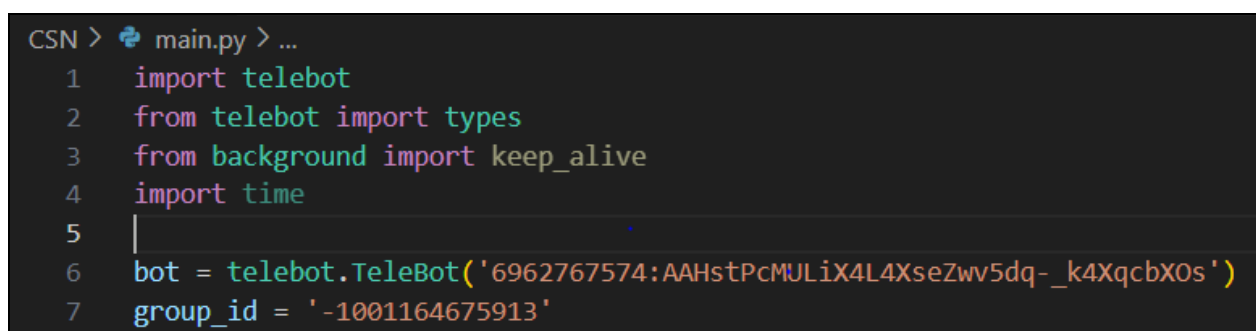
Усе, середовище розробки налагоджене та можна переходити до 3 етапу розробки.

3.3 Розробка програмної частини бота

Щоб розпочати прописувати команди боту, спочатку потрібно підключити бібліотеку до проекту. Для цього у полі прописуємо наступні імпорти:

- `import telebot` - бібліотека для роботи з ботом;
- `from telebot import types` – імпортування типів для кнопочкових форм;
- `from background import keep_alive` - підключення фласк сервера для тестування;
- `import time` – бібліотека для роботи з датами та часом.

Як було описано у підпункті 3.1, при створенні оболонки бота, нам надали токін, цей токін будемо використовувати для розробки власних команд, через API ми будемо управляти ботом. Додаємо дві змінних які будем, токен бота та ID групи. ID групи знадобиться для створення команди «Питання» для зворотнього зв'язку з користувачем, про це буде описано нижче. Лістинг кода у проекті наведений нижче на рисунку 3.8.



```
CSN > main.py > ...
1  import telebot
2  from telebot import types
3  from background import keep_alive
4  import time
5  |
6  bot = telebot.TeleBot('6962767574:AAHstPcMULiX4L4XseZwv5dq-_k4XqcbXOs')
7  group_id = '-1001164675913'
```

Рисунок 3.8 – Підключення бібліотек та токенів

Далі у проекті ми будемо розробляти тільки команди. Тому перша команда яка буде зустрічати користувача це команда «/start». Вона буде містити у собі текст з привітанням та посиланням на іншу команду.

Лістинг 3.1 – Команда «старт»

```
@bot.message_handler(commands=['start'])

def start(message):
    mess = f'Вітаю 🙌, {message.from_user.first_name}, я бот  
кафедри "Комп'ютерних систем та мереж". Натисніть /about щоб  
почати роботу.'
```

`bot.send_message(message.chat.id, mess, parse_mode='html')`

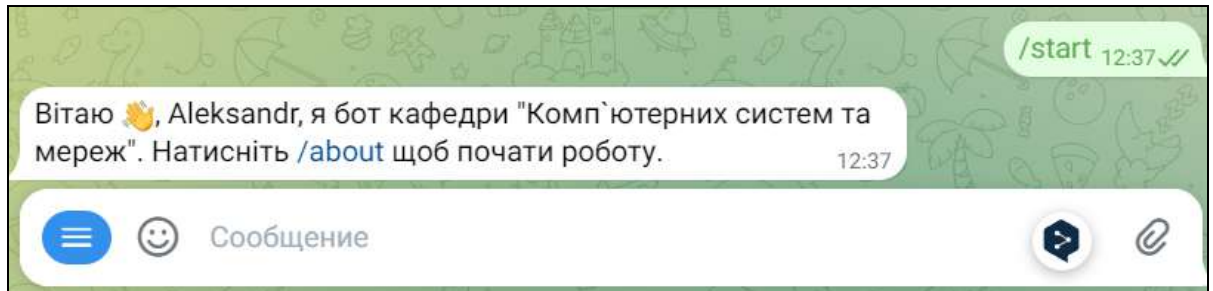


Рисунок 3.9 – Результат виконання команди /start

Наступна команда «/about», яка знайомить користувача з кафедрою, та надає посилання на наступну команду з функціоналом бота.

Лістинг 3.2 – Команда «Про себе»

```
@bot.message_handler(commands=['about'])

def about(message):
    mess = f"Ми кафедра 'Комп'ютерних систем та мереж'.  
Займаємося навчанням студентів спеціальності 123 'Комп'ютерна  
інженерія' та знайомимо їх зі світом ІТ-технологій.\nДля  
ознайомлення з функціонлом бота натисніть /help."
```

`bot.send_message(message.chat.id, mess, parse_mode='html')`

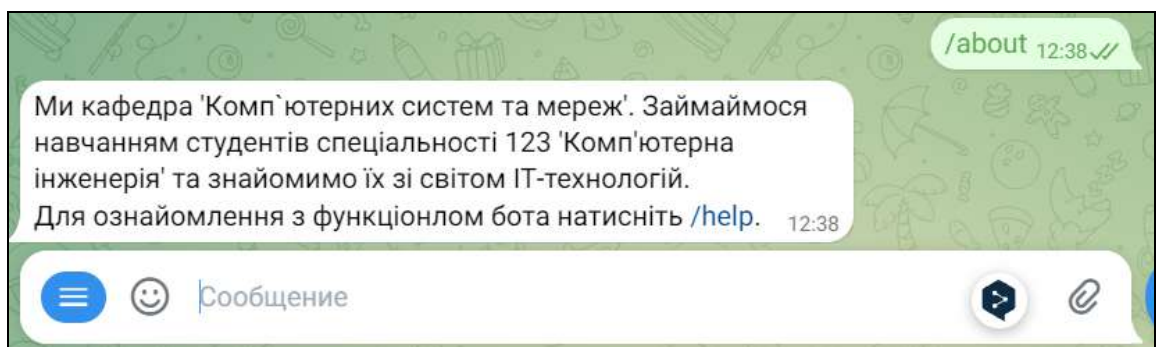


Рисунок 3.10 – Результат виконання команди /about

Далі, команда `/creators`. При виконанні команди користувач отримує текстовий список у якому наведено творці бота для вирішення технічних питань.

Лістинг 3.3 – Команда «Творці»

```
@bot.message_handler(commands=['creators'])

def creators(message):
    mess = f'Розробники бота:@Aleksandr_AGW'
    bot.send_message(message.chat.id, mess, parse_mode='html')
```

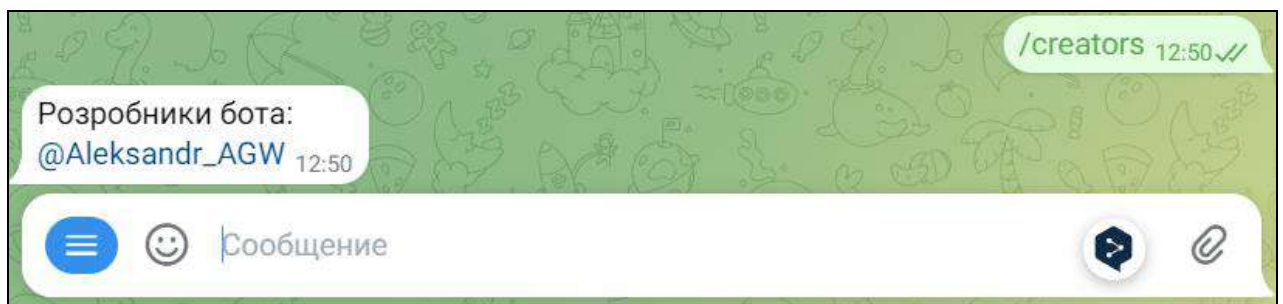


Рисунок 3.11 - Результат виконання команди `/creators`

Наступна команда - це `«/commands»`. При виконанні команди, користувач отримує текстовий список з переліком усіх команд цього бота.

Лістинг 3.4 – Команда «команди»

```
@bot.message_handler(commands=['commands'])

def commands(message):
    bot.send_message(message.chat.id, "Ось команди для роботи з ботом:\n"
    "/start - почати роботу з ботом;\n"
    "/about - інформація про нашу кафедру;\n"
    "/website - сайт кафедри КСМ;\n"
    "/commands - команди для роботи з ботом;\n"
    "/creators - розробники бота;\n"
    "/help - функціонал бота;\n"
    "/questions - запитання боту;\n"
    "/numbers - номери телефонів;\n"
    "/documents - документи для вступу;\n"
    "/location - місце знаходження")
```

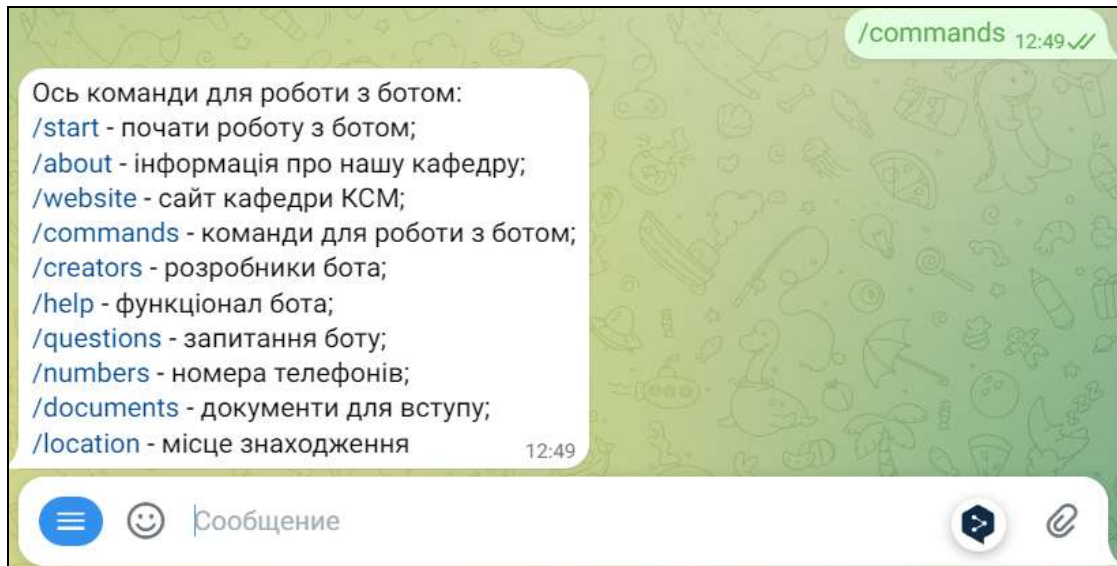


Рисунок 3.12 - Результат виконання команди /command

Далі за списком команда «/website». При виконанні команди, користувач отримує повідомлення від бота з інлайн кнопкою, при натисканні якої він перейде на головний сайт кафедри. Для створення інлайн кнопок використовується методи `types`, які ми імпортували.

Лістинг 3.5 – Команда «вебсайт»

```
@bot.message_handler(commands=['website'])
def website(message):
    markup = types.InlineKeyboardMarkup()
    markup.add(types.InlineKeyboardButton("Натисни на мене",
url="http://csn.zntu.edu.ua/"))
    bot.send_message(message.chat.id, "Вебсайт кафедри ",
reply_markup=markup)
```



Рисунок 3.13 - Результат виконання команди /website

Ще одна команда, це команда «/numbers». При виконанні команди, користувач отримує текстову відповідь з переліком номерів та електронних скриньок кафедри, декана та приймальної комісії.

Лістинг 3.6 – Команда «Номери»

```
@bot.message_handler(commands=['numbers'])
def numbers(message):
    mess = f'Номера для зв'язку з нами:\n
    'Деканат - (061) 769-83-32/(063) 0400215, пошта:
    dekanat.knt@gmail.com\n'
    'Приймальна комісія - (061) 769-82-26, пошта:
    nuzp.pk@gmail.com \n'
    'Кафедра - (061) 769-82-49/(073)-550-94-80, пошта:
    kafedra_ksm@zntu.edu.ua''
    bot.send_message(message.chat.id, mess, parse_mode='html')
```

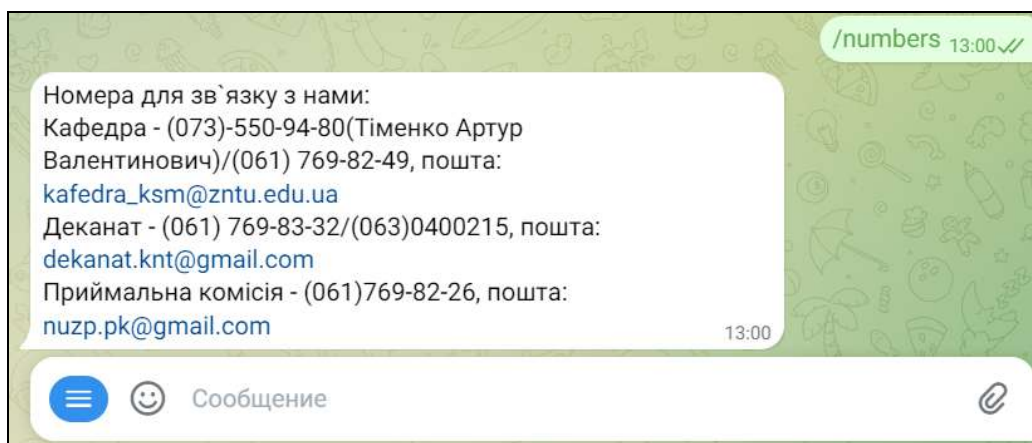


Рисунок 3.14 - Результат виконання команди /numbers

Далі команда «/documents». При виконанні команди, користувач отримує кнопкову форму з трьома варіантами вибору, далі користувач обирає на якій базі навчання він буде поступати до університету. Ця команда потребує метода call.data, тобто виклику даних за допомоги натискання кнопки. Щоб кнопки працювали, отрібно прописати так звані колбеки, про них поговоримо, коли будемо створювати командую функціонала бота.

Лістинг 3.7 – Команда «Документи»

```
@bot.message_handler(commands=['documents'])
def documents(message):
    markup = types.InlineKeyboardMarkup(row_width=3)
    markup_school = types.InlineKeyboardButton(
        text='Після школи', callback_data='/school')
    markup_colleg = types.InlineKeyboardButton(
```

```

        text='Після коледжа',callback_data='/colleg')
markup_magistr = types.InlineKeyboardButton(
    text='Магістратура',callback_data='/magistr')
markup.add(markup_school, markup_colleg, markup_magistr)
bot.send_message(message.chat.id, 'Документи до вступу',
reply_markup=markup)

```

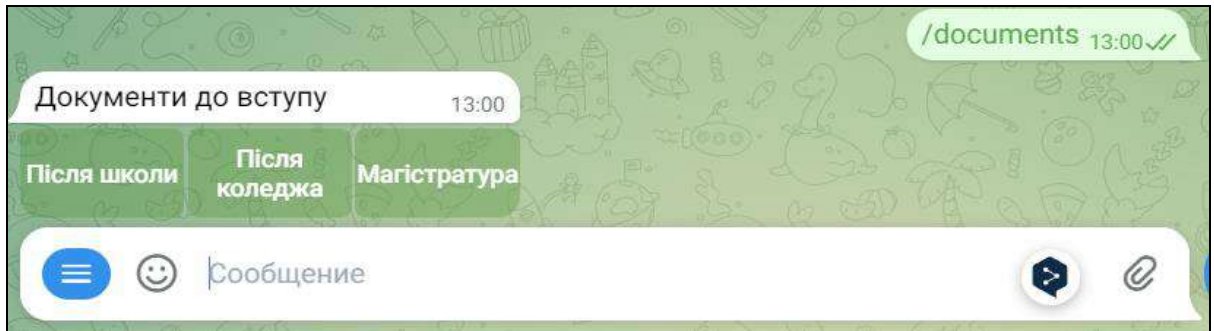


Рисунок 3.15 - Результат виконання команди /document

Команда «/location» схожа з минулою командою, але на відміну від неї, вона показує геолокацію основних точок, які знадобляться майбутньому студенту. Ця команда також працює за допомоги виклику даних колбек.

Лістинг 3.8 – Команда «Локація»

```

@bot.message_handler(commands=['location'])
def location(message):
    markup = types.InlineKeyboardMarkup(row_width=3)
    markup_reception = types.InlineKeyboardButton(
        text='Приймальна комісія', callback_data='/reception')
    markup_dekanat = types.InlineKeyboardButton(
        text='Деканат', callback_data='/dekanat')
    markup_department = types.InlineKeyboardButton(
        text='Кафедра', callback_data='/departmen')
    markup.add(markup_reception,markup_dekanat,
markup_department)
    bot.send_message(message.chat.id, 'Де нас шукати:',
reply_markup=markup)

```

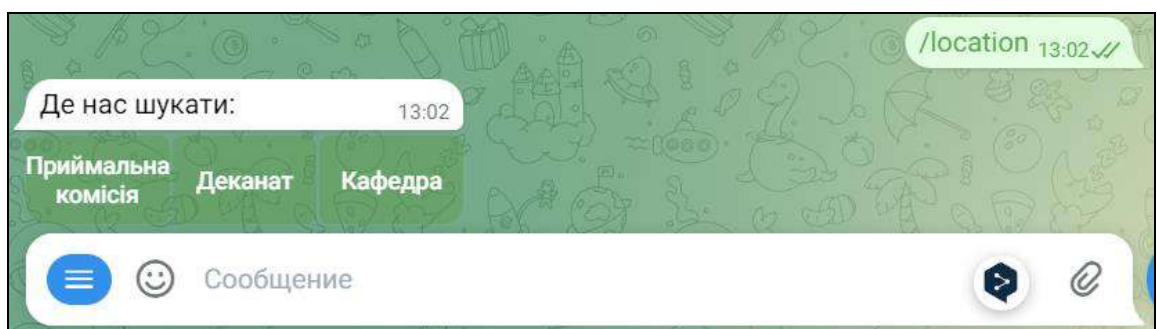


Рисунок 3.16 - Результат виконання команди /location

Одна з головних команд «/questions». При виконанні команди, користувач отримує кнопкову форму для задавання питань. При виборі «Задати питання», бот у користувача запропонує написати питання у чат, якщо «Скасувати» - запит буде скасовано.

Лістинг 3.9 – Команда «Питання»

```
@bot.message_handler(commands=['questions'])
def questions(message):
    markup = types.InlineKeyboardMarkup(row_width=2)
    markup.add(types.InlineKeyboardButton(
        text='Запитати бота?',
        callback_data='questions'), types.InlineKeyboardButton(text='Скасувати 🙅',
        callback_data='cancel'))
    msg = bot.send_message(message.chat.id, 'Натисніть на кнопки
    щоб продовжити', reply_markup=markup)
    bot.register_next_step_handler(msg, user_answer)

    def user_answer(message):
        if message.text == "Запитати бота?":
            mess_questions = bot.send_message(message.chat.id, 'Напиши питання')
            bot.register_next_step_handler(mess_questions,
            user_questions)
        elif message.text == "Скасувати 🙅":
            bot.send_message(message.chat.id, "Запит був скасован")

    def user_questions(message):
        bot.send_message(message.chat.id,
        f"{message.from_user.first_name} запитання, було надіслано. Згодом
        дамо відповідь 😊")
        bot.send_message(chat_id=group_id, text=f"Вопрос пользователя
        - {message.text}\n"
        f"Имя пользователя - {message.from_user.username}\n"
        f"ID пользователя - {message.from_user.id}")
```

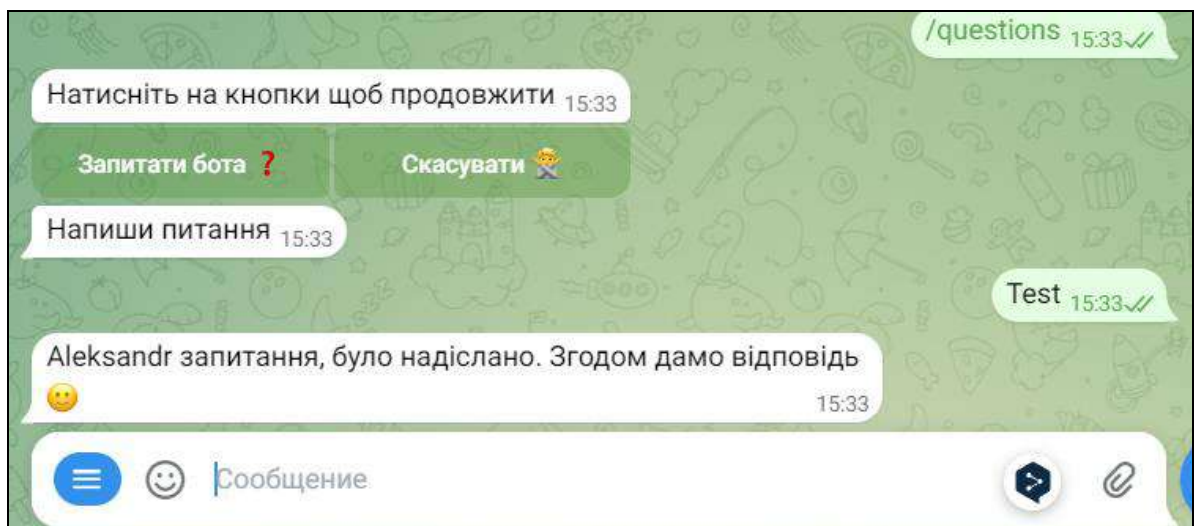


Рисунок 3.17 – Результат виконання команди /questions

Остання головна команда, це «/help». Ця команда є функціоналом бота, коли користувач викликає її, він отримує кнопкову форму з усіма вище перерахованими командами. Коли користувач обирає одну з кнопок, при натисканні виконується команда, як він обрав. При створенні цієї команди був використаний метод `type`, який був задіяний у інших командах. За допомогою типів є змога створити таке інтерактивне меню.

Для реалізації виклику команд використовуємо метод `callback.data`. Також його використовуємо для команд «Документи» та «Локація». Він працює за таким принципом: користувач натискає кнопку, програма бачить що визивається та чи інша команда, та визиває потрібну команду. Для того, щоб програма бачила, яку команду потрібно визивати, у методі пишеться наступні строки `markup=types.InlineKeyboardButton(text="callback_data=")` де:

- `markup` – це створення кнопки, яка може приймати будь-яку назву;
- `types.InlineKeyboardButton` – частина коду яка показує що це буде саме inline кнопка;
- `(text="callback_data=")` – у лапках «text» пишемо саму назву кнопки, а у «callback_data» задаємо id на який буде робиться callback.

Команда функціонала можна побачити на лістингу 3.10:

Лістинг 3.10 – Створення кнопкової форми, команда «help»

```
@bot.message_handler(commands=['help'])
def help(message):
    markup = types.InlineKeyboardMarkup(row_width=3, )
    markup_website = types.InlineKeyboardButton(
        text='Сайт кафедри КСМ', callback_data='/website')
    markup_commands = types.InlineKeyboardButton(
        text='Команди бота', callback_data='/commands')
    markup_creators = types.InlineKeyboardButton(
        text='Розробники', callback_data='/creators')
    markup_numbers = types.InlineKeyboardButton(
        text='Зв'язок з нами', callback_data='/numbers')
    markup_documents = types.InlineKeyboardButton(
        text='Документи', callback_data='/documents')
    markup_location = types.InlineKeyboardButton(
        text='Геолокація', callback_data='/location')
    markup_questions = types.InlineKeyboardButton(
        text='Питання боту', callback_data='/questions')
    markup.add(markup_website, markup_commands, markup_creators,
markup_documents, markup_numbers, markup_location,
markup_questions)
```

```
bot.send_message(message.chat.id, 'Функціонал бота',  
reply_markup=markup)
```

Кнопкова форма готова, але лише візуально, кнопки не будуть працювати, поки ми не налаштуємо колбеки для кожної кнопки, тому далі йде функція `def` з усіма колбеками. Треба звернути увагу, що у проєкті має бути лише одна функція, яка містить у собі колбека, інакше функції будуть заважати працювати одна одній.

Також хочу звернути увагу, що у визові функції можна написати, яку саме інформацію потрібно повертати. Тобто можна повернути функцію команди, наприклад `website(call.message)` – виклик вебсайту, або щоб поверталось текстове повідомлення. Завдяки цьому реалізовано повернення даних у командах «Документи» та «Локація». Лістинг коду команди можна побачити у додатку б.1

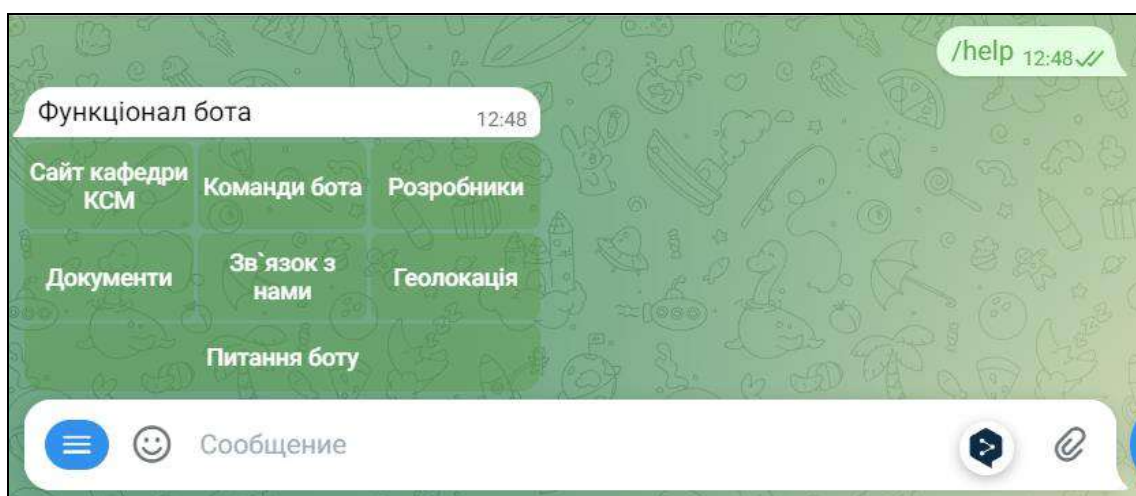


Рисунок 3.18 - Результат виконання команди `/help`

Остання команда, яка не буде явно використовуватись у боті, але має теж важливе значення. Ця команда текстова, перевіряє введений текст користувача. Якщо він не використовує запропоновані команди бот пише повідомлення «Не розумію. Ознайомтесь з моїми командами `/commands`» та пропонує ознайомитись з командами.

Лістинг 3.11 – Команда перевірки введеного тексту

```
@bot.message_handler(content_types=['text'])
def get_user_text(message):
    if message.text.lower() == "старт" == "start":
        mess = f"Вітаю 🤖, {message.from_user.first_name}, я бот
кафедри Комп'ютерних систем та мереж. "
        bot.send_message(message.chat.id, mess, parse_mode='html')
    else:
        bot.send_message(message.chat.id, "Не розумію.
Ознайомтесь з моїми командами /commands")
```

Та останній рядок у розробці, це команда яка вказує боту працювати далі після виклику команди.

```
bot.polling(none_stop=True, interval=0)
```

На програмному рівні бот готовий до старту. У наступному пункті проведемо тестування та виконаємо вивантаження на віртуальний сервер.

3.4 Тестування на хостинг бота

Останнім кроком реалізації системи є тестування та вивантаження на віртуальний сервер. Як зазначено вище, для реалізації цього етапу буду використовувати сайт Replit.

Для початку потрібно пройти реєстрацію на сайті. Це можна зробити за допомоги соц. мереж чи пройти швидку реєстрацію у ручну. У мене вже є акаунт який я створював за допомоги GitHub, тому продовжу використовуючи його. Після реєстрації сайт вам проведе коротку інструкцію що де знаходиться.

Далі потрібно створити Repl – це проект, у якому створимо середовище для бота та завантажимо код. Для цього у лівому верхньому куті обираємо Create Repl, далі обираємо мову програмування, назву проекту та створюємо його.

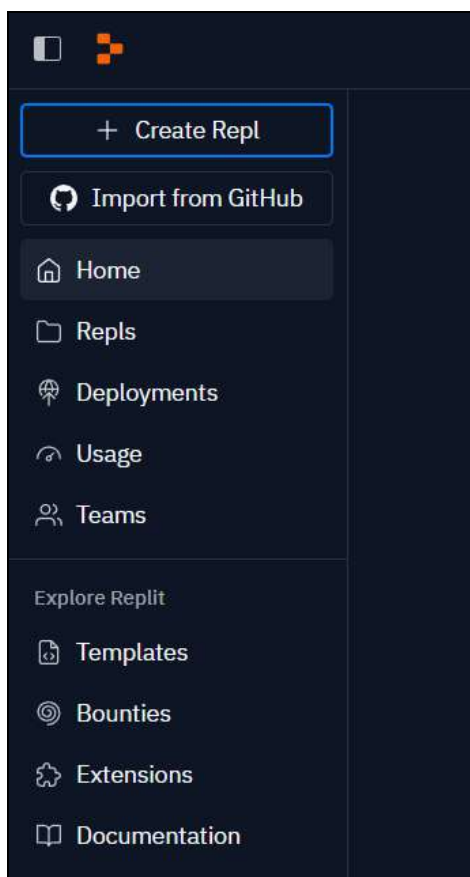


Рисунок 3.19 – Меню створення проекту

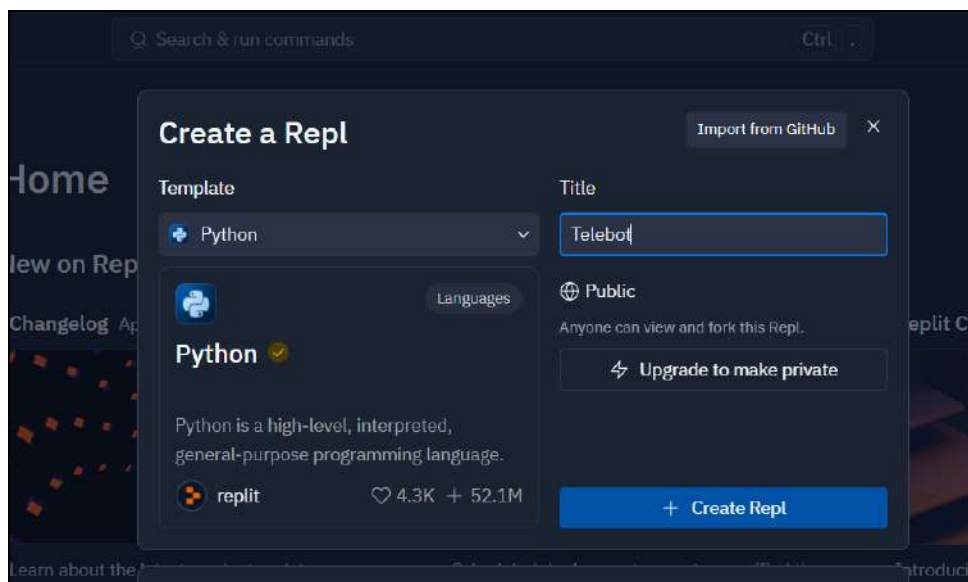


Рисунок 3.20 – Обираємо мову та назву для проекту

Усе проект створений, його можна знайти у власному профілі. Для цього переходимо у правий верхній кут. Натискаємо на іконку аватара. Обираємо Profile. Далі відкриється профіль де будуть наші проекти.

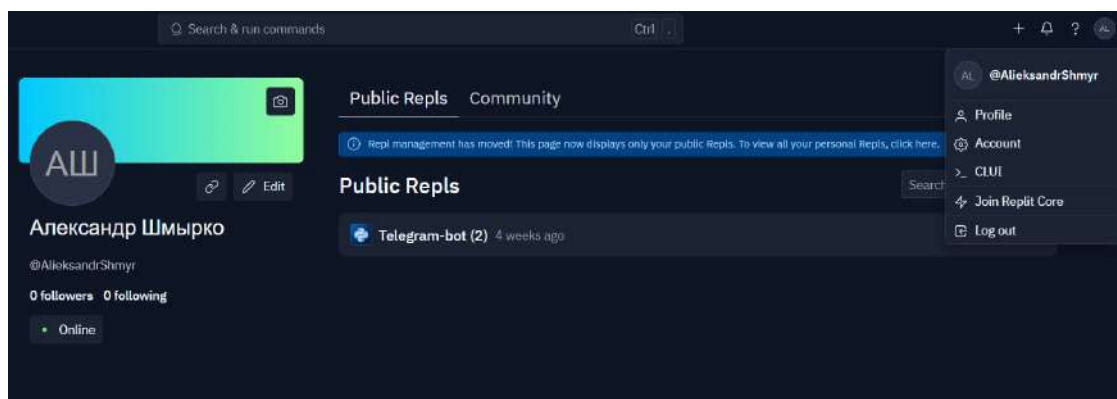


Рисунок 3.21 – Профіль з проектами

Далі заходимо у проект. При відкритті проекту ми бачимо зліва меню з усіма файлами, та одразу бібліотеками Python тому додатково щось встановлювати не потрібно. По центру головне робоче поле, куди ми завантажимо код. Та праворуч консоль, термінал та інші інструменти для роботи. Нам ще це знадобиться у тестуванні.

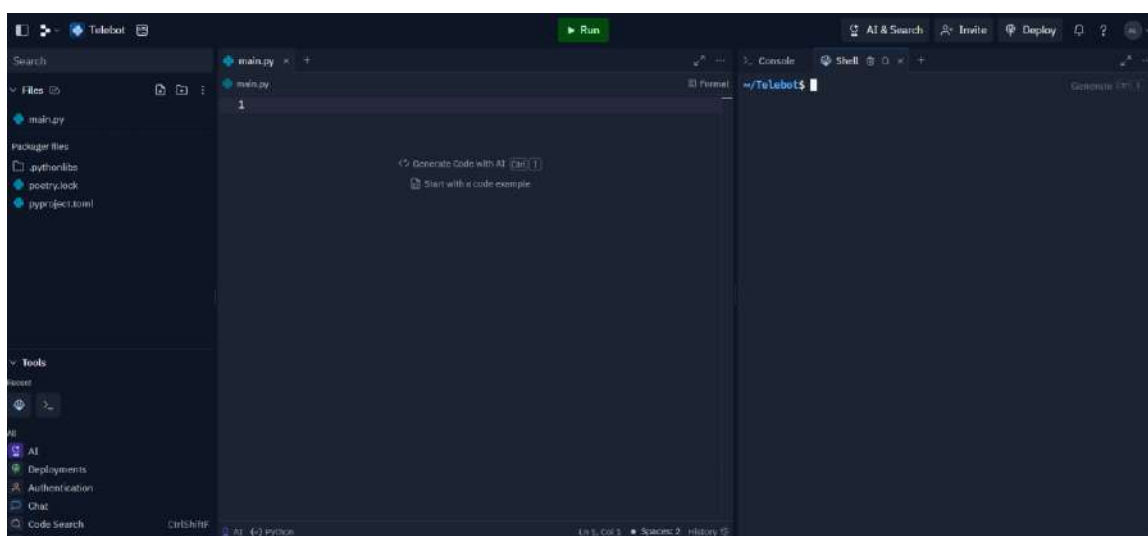


Рисунок 3.21 – Робоче поле проекту

Перейдемо до завантаження коду у робоче поле. Самий простий спосіб – це скопіювати код та вставити його сюди, або завантажити його за допомоги гіт хаба. Для цього потрібно використовувати додаткові інструменти. Далі потрібно встановити бібліотеку telebot. Для цього імпортуємо Pip у проект `import pip` та у терміналі вводимо наступне `pip install pytelegrambotapi`. Далі проходить встановлення.

```

~/Telebot$ pip install pytelegrambotapi
Collecting pytelegrambotapi
  Downloading pytelegrambotapi-4.18.1-py3-none-any.whl.metadata (48 kB)
    48.1/48.1 kB 1.9 MB/s eta 0:00:00
Collecting requests (from pytelegrambotapi)
  Downloading requests-2.32.1-py3-none-any.whl.metadata (4.6 kB)
Collecting charset-normalizer<4,>=2 (from requests->pytelegrambotapi)
  Downloading charset_normalizer-3.3.2-cp310-cp310-manylinux_2_17_x86_64.manylin
  nux2014_x86_64.whl.metadata (33 kB)
Collecting idna<4,>=2.5 (from requests->pytelegrambotapi)
  Downloading idna-3.7-py3-none-any.whl.metadata (9.9 kB)
Collecting urllib3<3,>=1.21.1 (from requests->pytelegrambotapi)
  Downloading urllib3-2.2.1-py3-none-any.whl.metadata (6.4 kB)
Collecting certifi>=2017.4.17 (from requests->pytelegrambotapi)
  Downloading certifi-2024.2.2-py3-none-any.whl.metadata (2.2 kB)
Downloading pytelegrambotapi-4.18.1-py3-none-any.whl (242 kB)
    242.9/242.9 kB 1.9 MB/s eta 0:00:00
Downloading requests-2.32.1-py3-none-any.whl (63 kB)
    63.7/63.7 kB 2.6 MB/s eta 0:00:00
Downloading certifi-2024.2.2-py3-none-any.whl (163 kB)
    163.8/163.8 kB 7.0 MB/s eta 0:00:00
Downloading charset_normalizer-3.3.2-cp310-cp310-manylinux_2_17_x86_64.manylinu
  x2014_x86_64.whl (142 kB)
    142.1/142.1 kB 5.9 MB/s eta 0:00:00
Downloading idna-3.7-py3-none-any.whl (66 kB)
    66.8/66.8 kB 2.9 MB/s eta 0:00:00
Downloading urllib3-2.2.1-py3-none-any.whl (121 kB)
    121.1/121.1 kB 5.4 MB/s eta 0:00:00
Installing collected packages: urllib3, idna, charset-normalizer, certifi, requ
  ests, pytelegrambotapi
Successfully installed certifi charset-normalizer idna pytelegrambotapi request
  s urllib3
~/Telebot$

```

Рисунок 3.22 – Встановлення бібліотеки telebot

Після цього перенесемо код у робоче поле. Import `pip` можна прибрати. Та натискаємо зелену кнопку «Run». Replit має потужний дебагер помилок, тому спочатку код перевіряємо на синтаксичні помилки. Помилки нема, тому можна купляти хостинг та ставити його на безперебійну роботу.

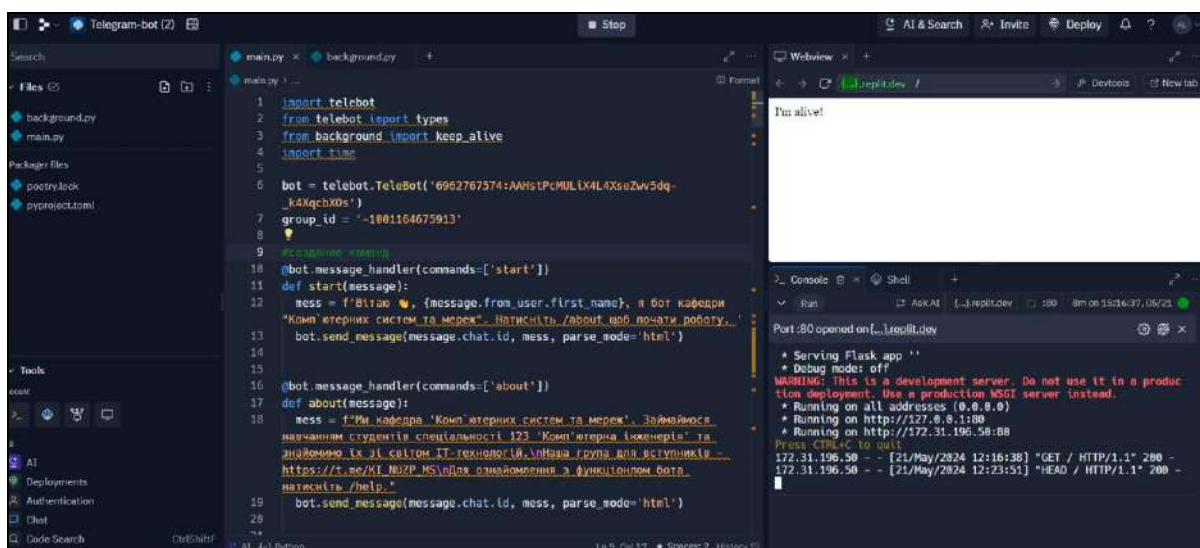


Рисунок 3.23 – Готовий проект до запуску

Останнє – це покупка хостингу. Для цього перейдемо на головну сторінку сайту та обираємо тариф 5\$ на місяць. Надалі, адміністрування ботом буде займатись відповідальна за це людина. Усі вдосконалення бота можна проводити безпосередньо вже на хостингу.

ВИСНОВКИ

Після аналізу технічного завдання було сформовано інструмент та програмне забезпечення для розробки комп'ютерної системи інформаційної допомоги абітурієнтам. Обґрунтовано обрано сервер для вивантаження бота для безперебійної роботи системи. В процесі використовувались сучасні методи та інструменти, такі як сучасні мови програмування та фреймворки, оптимальні середовища реалізації кода, що гарантує високу ефективність та актуальність продукту.

Сформовані завдання з тестування та адаптації системи для різних пристроїв, забезпечуючи її оптимальну функціональність. Вивантаження на сервер та подальша підтримка роботи системи відбудеться з дотриманням вимог безпеки та стабільності.

Система буде надалі вдосконалюватись для більш оптимальної роботи. Та додаватись нові функції.

Отже системи створена, протестована та готова до безпербійної роботи, та допоможе зменшити навантаження на деканат та інші учбові установи.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Matthes E. Python Crash Course, 2nd Edition: A Hands-On, Project-Based Introduction to Programming. No Starch Press, 2019. 544 с.
2. Beck K. Test Driven Development: By Example (The Addison-Wesley Signature Series). Addison-Wesley Professional, 2002. 240 с.
3. Haverbeke M. Eloquent JavaScript: A Modern Introduction to Programming. No Starch Press, 2014. 472 с.
4. Shevat A. Designing Bots: Creating Conversational Experiences. O'Reilly Media, Incorporated, 2017. 348 с.
5. Raj S. Building Chatbots with Python: Using Natural Language Processing and Machine Learning. Apress, 2018. 211 с.
6. Ciesla R. The Book of Chatbots. Cham : Springer Nature Switzerland, 2024. URL: <https://doi.org/10.1007/978-3-031-51004-5> (дата звертання 17.04.2024).
7. Що таке чат-бот: визначення, приклади, відео | SendPulse UA. SendPulse. URL: <https://sendpulse.ua/support/glossary/chatbot> (дата звертання 17.04.2024).
8. Welcome to Python. Python.org. URL: <https://www.python.org/> (дата звертання 17.04.2024).
9. Що таке Python? - aCode. URL: <https://acode.com.ua/intro-python/> (дата звертання 17.04.2024).
10. Менеджер ботів BotFather. Telegram Web. URL: <https://web.telegram.org/k/#@BotFather> (дата звертання 20.04.2024).
11. Flask vs Django. Обираємо фреймворк для веб-розробки - Альтернативна Наука Віталія Сема. Альтернативна Наука Віталія Сема. URL: <https://alternativescience.net/programming/140-flask-vs-django-obyrayemo-frejmwork-dlya-veb-rozrobky/> (дата звертання 21.04.2024).
12. Bots: An introduction for developers. Telegram APIs. URL: <https://core.telegram.org/bots> (date of access: (дата звертання 24.04.2024).

13. MTProto Mobile Protocol. Telegram APIs.
URL: <https://core.telegram.org/mtproto> (дата звертання 24.04.2024).
14. Telegram APIs. URL: <https://core.telegram.org/api> (дата звертання 24.04.2024).
15. Telegram Bot API. Telegram APIs.
URL: <https://core.telegram.org/bots/api> (дата звертання 24.04.2024).
16. JetBrains. PyCharm: the Python IDE for data science and web development. JetBrains. URL: <https://www.jetbrains.com/pycharm/> (дата звертання 01.05.2024)
17. Microsoft. Visual Studio Code - Code Editing. Redefined. Visual Studio Code - Code Editing. Redefined. URL: <https://code.visualstudio.com/> (дата звертання 01.05.2024)
18. Replit. URL: <https://replit.com/> (дата звертання 01.05.2024)

ДОДАТОК А Основні команди бота BotFather

Таблиця А.1

Команди	Значення
/newbot	Створює нового бота
/mybot	Дає змогу змінювати вже існуючих ботів
/token	Генерує унікальний токен аунтифікації бота
/setname	Дозволяє встановити ім'я боту
/setabouttext	Дозволяє встановити поле «Про себе»
/setdescription	Дозволяє встановити опис бота
/setuserpic	Дозволяє встановити аватар боту
/setcommands	Дозволяє встановити команди боту
/deletebot	Видалення бота

ДОДАТОК Б Лістинг команд бота

Лістинг Б.1 – Створення визову даних для кнопкової форми

```

bot.callback_query_handler(func=lambda call: True)
def help_answer(call):
    if call.data == "/website":
        website(call.message)
    elif call.data == "/about":
        about(call.message)
    elif call.data == "/commands":
        commands(call.message)
    elif call.data == "/creators":
        creators(call.message)
    elif call.data == "/documents":
        documents(call.message)
    elif call.data == "/location":
        location(call.message)
    elif call.data == "/numbers":
        numbers(call.message)
    elif call.data == "/reception":
        bot.send_message(call.message.chat.id, f'Приймальна
комісія: м.Запоріжжя, вул. Жуковського, 60, 7-ий корпус, каб
2.2\n' 'Геолокація: https://bit.ly/placereception')
    elif call.data == "/dekanat":
        bot.send_message(call.message.chat.id, f'Деканат
факультету комп'ютерних наук і технологій м.Запоріжжя,
вул.Жуковського, 64, головний корпус, к.362.\n' 'Геолокація:
https://bit.ly/placedekanat')
    elif call.data == "/departmen":
        bot.send_message(call.message.chat.id, f'Кафедра
комп'ютерних систем та мереж НУ "Запорізька політехніка":
м.Запоріжжя, вул.Жуковського, 64, 3-й корпус, каб.53б.\n'
'Геолокація: https://bit.ly/placedepartmen')
    elif call.data == "/school":
        bot.send_message(call.message.chat.id,
f'○ Для здобуття ступеня бакалавра. \n\n'
"Перелік документів для вступу:\n " "▪ документ, що посвідчує
особу\n "
"▪ військово-обліковий документ\n "
"▪ картка платника податків\n "
"▪ свідоцтво про повну загальну середню освіту/диплом
молодшого спеціаліста (ФМБ, МБ) та додаток до нього\n "
"▪ копія сертифіката (сертифікатів) ЗНО та/або НМТ (за
наявності)\n "
"▪ мотиваційний лист (в електронному та паперовому
вигляді)\n"
"▪ чотири кольорові фотокартки розміром 3 x 4
см\n\n'"👉Реєстрація заяв в приймальній комісії, ауд. 2.2, або за
телефонами: \n\n' '098 523 6789 \n' '099 523 6789 \n' '(061) 769
82 26\n')
    elif call.data == "/colleg":

```

```

        bot.send_message(call.message.chat.id,
        f'○ Для здобуття ступеня бакалавра. \n\n'
        "Перелік документів для вступу:\n "
        "▪ документ, що посвідчує особу\n "
        "▪ військово-обліковий документ\n "
        "▪ картка платника податків\n "
        "▪ свідоцтво про повну загальну середню освіту/диплом
        молодшого спеціаліста (ФМБ, МБ) та додаток до нього\n "
        "▪ копія сертифіката (сертифікатів) ЗНО та/або НМТ (за
        наявності)\n "
        "▪ мотиваційний лист (в електронному та паперовому
        вигляді)\n"
        "▪ чотири кольорові фотокартки розміром 3 x 4
        см\n\n"')
        # Реєстрація заяв в приймальній комісії, ауд. 2.2, або за
        телефонами: \n\n''098 523 6789 \n''099 523 6789 \n''(061) 769 82
        26\n')
        elif call.data == "/magistr":
            bot.send_message(call.message.chat.id,
            f'● Для здобутого ступеня вищої освіти.\n\n'
            'Прийом заяв та документів відбувається з 4 по 23 вересня.
            \n\n'
            'Перелік документів для вступу:\n '
            '▪ документ, що посвідчує особу \n'
            '▪ військово-обліковий документ \n'
            '▪ картка платника податків \n'
            '▪ документ про освіту з додатком \n'
            '▪ копія сертифіката (сертифікатів) ЄВІ та/або ЄФВВ (за
            наявності)\n '
            '▪ мотиваційний лист (в електронному та паперовому вигляді)\n'
            '▪ чотири кольорові фотокартки розміром 3 x 4 см \n\n'
            '📌 Реєстрація заяв на участь у вступних іспитах здійснюється
            до 18:00 21.09 особисто в приймальній комісії, ауд. 2.2, або за
            телефонами:\n''098 523 6789\n''099 523 6789\n''(061) 769 82 26\n')
            elif call.data == "/questions":
                markup = types.InlineKeyboardMarkup(row_width=2)
                markup.add(types.InlineKeyboardButton(text='Запитати бота ?',
                callback_data='questions'),
                types.InlineKeyboardButton(text='Скасувати 🙋',
                callback_data='cancel'))
                msg = bot.send_message(call.message.chat.id, "Натисніть
                на кнопки щоб продовжити", reply_markup=markup)
                bot.register_next_step_handler(msg, user_answer)
            elif call.data == 'questions':
                mess_questions = bot.send_message(call.message.chat.id,
                'Напиши питання')
                bot.register_next_step_handler(mess_questions,
                user_questions)
            elif call.data == 'cancel':
                bot.send_message(call.message.chat.id, "Запит був
                скасован")
            else:
                bot.send_message(call.message.chat.id, "Не розумію.
                Ознайомтесь з моїми командами /commands")

```

Актуальність використання чат-ботів

Як будуть допомагати чат-боти компаніям та іншим установам на думку фахівців



Вже долучені до використання:



Завдання, які виконують чат-боти



					13.02070849.00034 ПЛ1						
					Розробка комп'ютерної системи інформаційної допомоги абітурієнтам	Лист.		Маса		Масшт.	
Зам.	Лист.	№ докум.	Підп.	Дата	Актуальність використання чат-ботів	Лист 1		Листів 1			
Розроб.		Шмирко О.С.									
Перев.		Хохлов М.М.									
Т.контр.											
Н.контр.		Щербак Н.В.				НУ «Запорізька політехніка» КНТ-511сп					
Затв.		Кудерметов Р.К.									

Виконання основних команд чат-бота

/start 12:37✓✓

Вітаю 🙌, Aleksandr, я бот кафедри "Комп'ютерних систем та мереж". Натисніть /about щоб почати роботу.

12:37

Сообщение

/help 12:48✓✓

Функціонал бота

12:48

Сайт кафедри КСМ

Команди бота

Розробники

Документи

Зв'язок з нами

Геолокація

Питання боту

Сообщение

/questions 15:33✓✓

Натисніть на кнопки щоб продовжити

15:33

Запитати бота ?

Скасувати

Напиши питання

15:33

Test

15:33✓✓

Aleksandr запитання, було надіслано. Згодом дамо відповідь

15:33

Сообщение

Информация о боте

КСМ

CSN_Vstup

бот

@ CSN_NUZPbot
Имя пользователя

Бот кафедри Комп'ютерних систем та мереж, спеціальності 123 "Комп'ютерна інженерія"

О себе

Ось команди для роботи з ботом:

/start - почати роботу з ботом;

/about - інформація про нашу кафедру;

/website - сайт кафедри КСМ;

/commands - команди для роботи з ботом;

/creators - розробники бота;

/help - функціонал бота;

/questions - запитання боту;

/numbers - номери телефонів;

/documents - документи для вступу;

/location - місце знаходження

12:49

Сообщение

/location 13:02✓✓

Де нас шукати:

13:02

Приймальна комісія

Деканат

Кафедра

Сообщение

/documents 13:00✓✓

Документи до вступу

13:00

Після школи

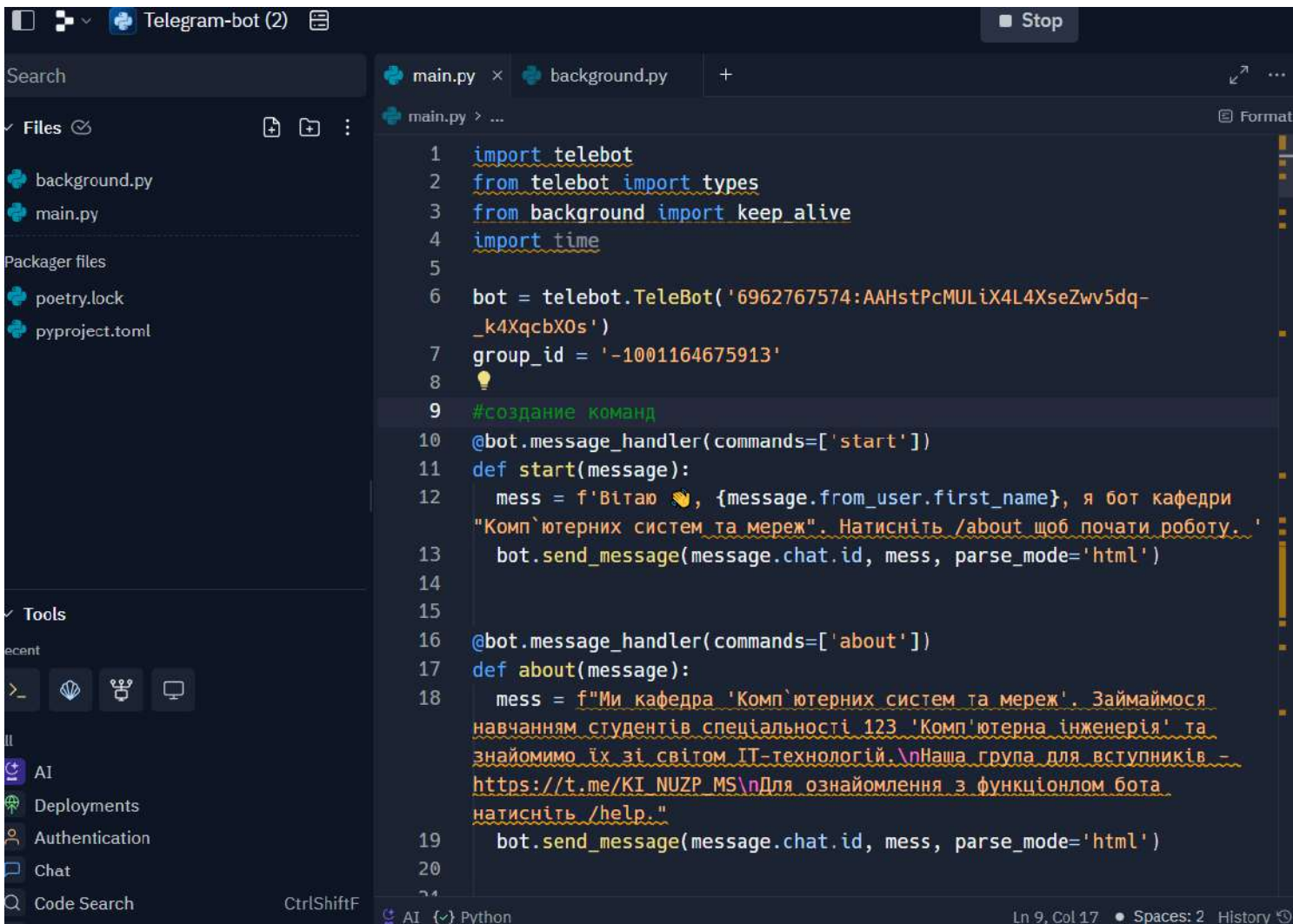
Після коледжа

Магістратура

Сообщение

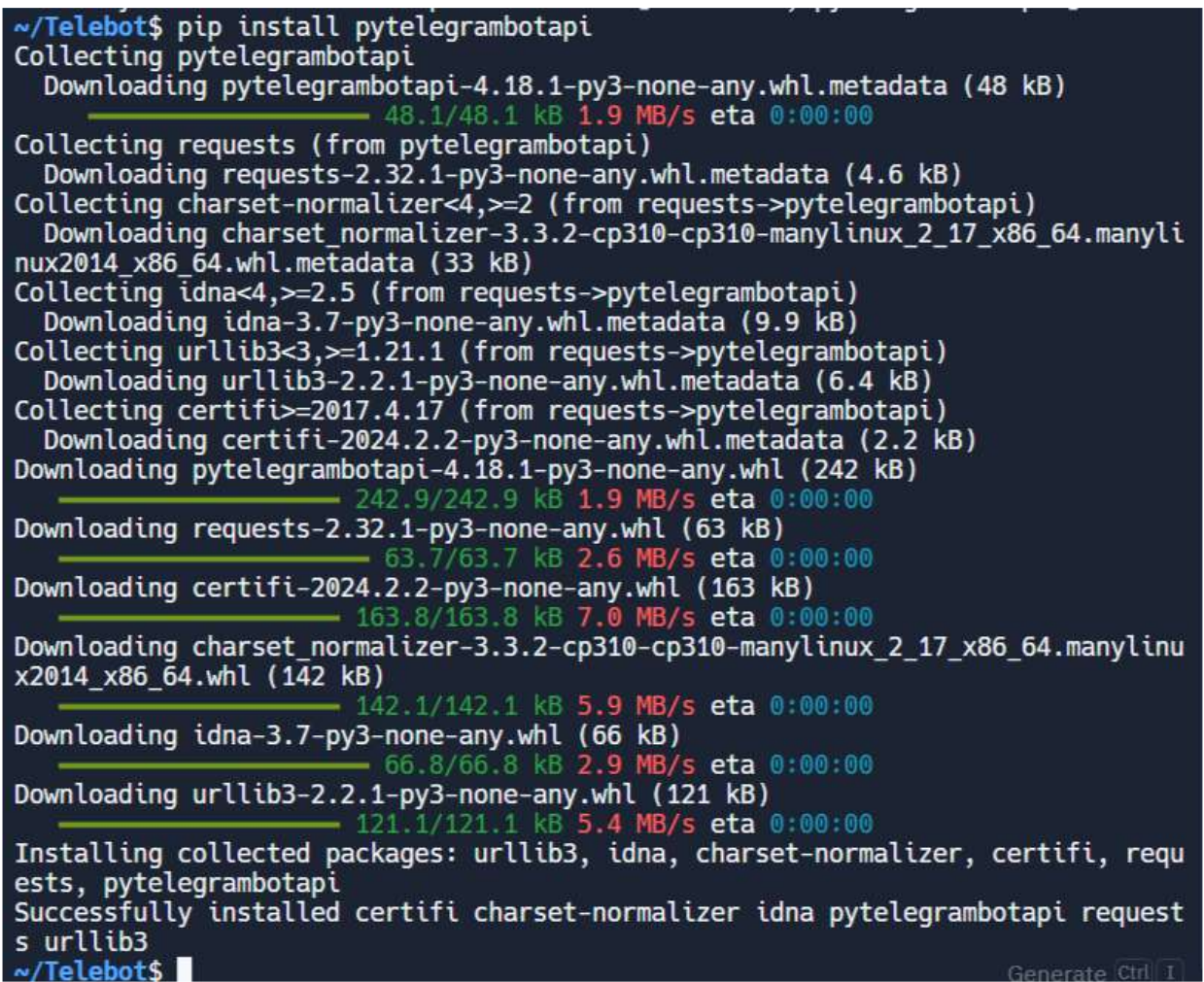
					13.02070849.00034 ПЛ2			
					Розробка комп'ютерної системи інформаційної допомоги абітурієнтам	Лист.	Маса	Масшт.
Зам.	Лист.	№ докум.	Підп.	Дата				
Розроб.		Шмирко О.С.			Виконання основних команд чат-бота	Лист 1		Листів 1
Перев.		Хохлов М.М.				НУ «Запорізька політехніка»		
Т.контр.						КНТ-511сп		
Н.контр.		Щербак Н.В.						
Затв.		Кудерметов Р.К.						

Вивантаження чат-бота на віртуальний сервер



```
1 import telebot
2 from telebot import types
3 from background import keep_alive
4 import time
5
6 bot = telebot.TeleBot('6962767574:AAHstPcMULiX4L4XseZwv5dq-
7 _k4XqcbX0s')
8
9 #создание команд
10 @bot.message_handler(commands=['start'])
11 def start(message):
12     mess = f'Вітаю 🤖, {message.from_user.first_name}, я бот кафедри
13     "Комп'ютерних систем та мереж". Натисніть /about щоб почати роботу.'
14     bot.send_message(message.chat.id, mess, parse_mode='html')
15
16 @bot.message_handler(commands=['about'])
17 def about(message):
18     mess = f"Ми кафедра 'Комп'ютерних систем та мереж'. Займаємося
19     навчанням студентів спеціальності 123 'Комп'ютерна інженерія' та
20     знайомимо їх зі світом IT-технологій.\nНаша група для вступників -
21     https://t.me/KI_NUZP_MS\nДля ознайомлення з функціями бота
22     натисніть /help."
23     bot.send_message(message.chat.id, mess, parse_mode='html')
```

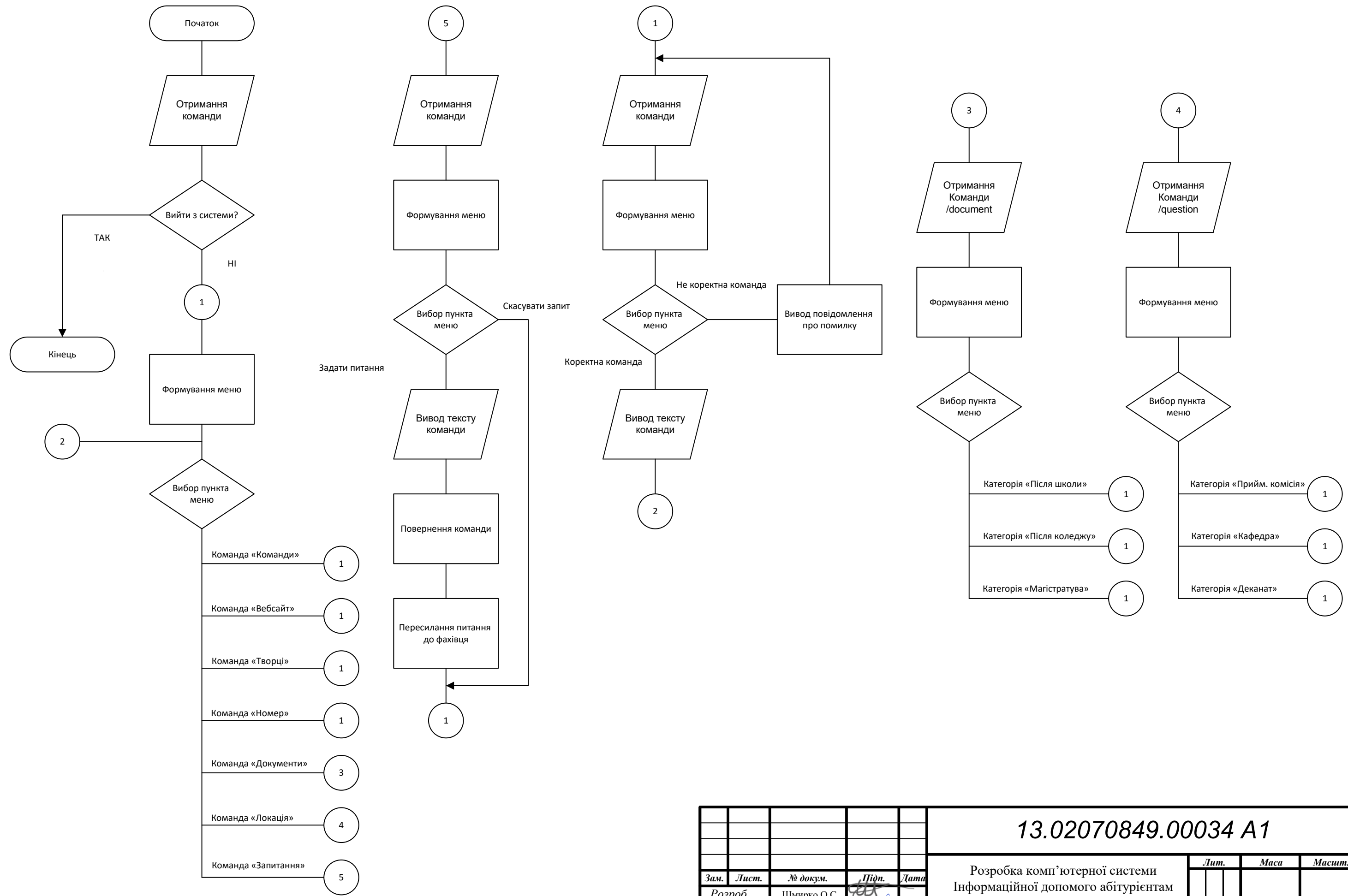
Вигрузка бота на віртуальний сервер



```
~/Telebot$ pip install pytelegrambotapi
Collecting pytelegrambotapi
  Downloading pytelegrambotapi-4.18.1-py3-none-any.whl.metadata (48 kB)
    48.1/48.1 kB 1.9 MB/s eta 0:00:00
Collecting requests (from pytelegrambotapi)
  Downloading requests-2.32.1-py3-none-any.whl.metadata (4.6 kB)
Collecting charset-normalizer<4,>=2 (from requests->pytelegrambotapi)
  Downloading charset_normalizer-3.3.2-cp310-cp310-manylinux_2_17_x86_64.manyli
nux2014_x86_64.whl.metadata (33 kB)
Collecting idna<4,>=2.5 (from requests->pytelegrambotapi)
  Downloading idna-3.7-py3-none-any.whl.metadata (9.9 kB)
Collecting urllib3<3,>=1.21.1 (from requests->pytelegrambotapi)
  Downloading urllib3-2.2.1-py3-none-any.whl.metadata (6.4 kB)
Collecting certifi<=2017.4.17 (from requests->pytelegrambotapi)
  Downloading certifi-2024.2.2-py3-none-any.whl.metadata (2.2 kB)
Downloading pytelegrambotapi-4.18.1-py3-none-any.whl (242 kB)
    242.9/242.9 kB 1.9 MB/s eta 0:00:00
Downloading requests-2.32.1-py3-none-any.whl (63 kB)
    63.7/63.7 kB 2.6 MB/s eta 0:00:00
Downloading certifi-2024.2.2-py3-none-any.whl (163 kB)
    163.8/163.8 kB 7.0 MB/s eta 0:00:00
Downloading charset_normalizer-3.3.2-cp310-cp310-manylinux_2_17_x86_64.manylinu
x2014_x86_64.whl (142 kB)
    142.1/142.1 kB 5.9 MB/s eta 0:00:00
Downloading idna-3.7-py3-none-any.whl (66 kB)
    66.8/66.8 kB 2.9 MB/s eta 0:00:00
Downloading urllib3-2.2.1-py3-none-any.whl (121 kB)
    121.1/121.1 kB 5.4 MB/s eta 0:00:00
Installing collected packages: urllib3, idna, charset-normalizer, certifi, requ
ests, pytelegrambotapi
Successfully installed certifi charset-normalizer idna pytelegrambotapi request
s urllib3
~/Telebot$
```

Встановлення додаткової бібліотеки TeleBot

					13.02070849.00034 ПЛЗ									
					Розробка комп'ютерної системи інформаційної допомоги абітурієнтам	Лист.			Маса			Масшт.		
Зам.	Лист.	№ докум.	Підп.	Дата										
Розроб.		Шмирко О.С.												
Перев.		Хохлов М.М.												
Т.контр.														
					Вивантаження чат-бота на віртуальний сервер	Лист 1			Листів 1					
Н.контр.		Щербак Н.В.				НУ «Запорізька політехніка»								
Затв.		Кудерметов Р.К.				КНТ-511сп								



					13.02070849.00034 А1							
					Розробка комп'ютерної системи Інформаційної допомоги абітурієнтам				Лист.		Маса	Масшт.
Зам.	Лист.	№ докум.	Підп.	Дата	Алгоритм обробки команд системи				Лист 1		Листів 1	
Розроб.		Шмирко О.С.										
Перев.		Хохлов М.М.										
Т.контр.												
Н.контр.		Щербак Н.В.										
Затв.		Кудерметов Р.К.										