Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____Комп'ютерної інженерії та управління_____
(повна назва)
Кафедра ____Автоматизації проектування обчислювальної техніки_____
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти ___другий (магістерський)_____
(рівень вищої освіти)

Хмарний сервіс на основі чат-боту для зберігання, сортування та передачі файлів електронною поштою
(тема)

Виконав: студент 2 курсу, групи СКСм-21-1
Деркачов О.В.
(прізвище, ініціали)
Спеціальність _____
123 Комп'ютерна інженерія
(код і повна назва спеціальності)
Тип програми _____
освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма_____
Спеціалізовані комп'ютерні системи
(повна назва освітньої програми)
Керівник роботи _____доц. Шкіль О.С.____
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Чумаченко С.В._____
(підпис) (прізвище, ініціали)

2022 р.

Харківський національний університет радіоелектроніки

Факультет_____Комп'ютерної інженерії та управління _____

Кафедра_____Автоматизації проектування обчислювальної техніки_____

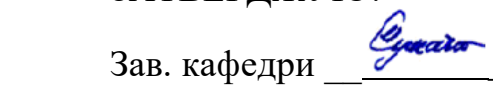
Рівень вищої освіти____другий (магістерський)_____

Спеціальність_____123_ Комп'ютерна інженерія _____
(шифр і назва)

Тип програми _____Освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)

Освітня програма _____Спеціалізовані комп'ютерні системи_____
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри __  _
(підпис)

«_02_» _____ 2022 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту__Деркачову Олегу Віталійовичу_____
(прізвище, ім'я, по батькові)

Тема роботи (проекту)_ Хмарний сервіс на основі чат-боту для зберігання, сортування та передачі файлів електронною поштою
Chatbot-Based Cloud Service for Storage, Sorting and Transferring Files by E-mail

затверджена наказом по університету від «__14_»__11__2022 р. № _1478 Ст_____

2. Термін подання студентом роботи _____

3. Вихідні дані до роботи (проекту)_____
Електронна пошта_____
Хмарний сервіс_____
Середовище PyCharm_____
Мова програмування Python_____
_____

4. Перелік питань, що потрібно опрацювати у роботі _____
Налаштування середовища чат-боту_____
Проектування архітектури хмарного сервісу_____
Процес введення даних_____
_____
_____

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) 17 слайдів

6. Консультанти розділів роботи (проекту)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

7. Дата видачі завдання 01.09.2022

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи (проекту)	Термін виконання етапів проекту (роботи)	Примітка
1	Отримання завдання на кваліфікаційну роботу	01.09.2022 - 05.09.2022	виконано
2	Пошук та аналіз літературних джерел за темою дипломної роботи	05.09.2022 - 30.09.2022	виконано
3	Визначення основних етапів алгоритмів для оптимізації	01.10.2022 - 15.10.2022	виконано
4	Пошук методів вирішення проблеми	15.10.2022 - 05.11.2022	виконано
5	Програмна реалізація	05.11.2022 - 20.11.2022	виконано
6	Опис програмної реалізації	20.11.2022 - 01.12.2022	виконано
7	Оформлення роботи	01.12.2022 - 15.12.2022	виконано
8	Представлення роботи до захисту	15.12.2022 - 25.12.2022	виконано

Студент _____
(підпис)

Керівник роботи (проекту) _____ доц. кафедри АПОТ Шкіль О.С.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи: 57 с., 39 рис., 5 джерел.

ЧАТ-БОТ, PYTHON, PYCHARM, ХМАРНИЙ СЕРВІС, ЕЛЕКТРОННА ПОШТА, ОБРОБКА ФАЙЛІВ,

В Структурі кіберуніверситету є хмарні кібер-сервіси для безпаперового електронного документообігу. Чат-бот на основі хмарного сервісу є одним із них.

Об’єкт дослідження у роботі – Чат-бот на основі хмарного сервісу у складі кіберуніверситету.

Предмет дослідження – моделі, методи та процедури оптимізації безпаперового електронного документообігу з використанням хмарних технологій.

Мета роботи - це проектування хмарного сервісу для файлової системи на основі чат-боту в рамках реалізації хмарних сервісів кіберуніверситету для полегшення якості навчального процесу студента.

Хмарний сервіс передбачає покращення процесів менеджменту файлів та полегшення організації навчання студентів в період сесії та екзаменів.

ABSTRACT

The explanatory note contains: 57 pages, 39 figure, 5 sources according to the list of links.

CHAT-BOT, PYTHON, PYCHARM, CLOUD SERVICE, ELECTRONIC MAIL, FILE PROCESSING

In the Structure of the Cyber University, there are cyber-services for paperless electronic document processing. A chatbot based on cloud service is one of them.

The object of research in the work is a chatbot based on a cloud service as part of a cyber university.

The subject of the research is models, methods and procedures for optimizing paperless electronic document flow using cloud technologies.

The purpose of the work is to design a cloud service for a file system based on a chatbot as part of the implementation of cloud services of a cyber university to facilitate the quality of the student's educational process.

The cloud service provides for improving file management processes and facilitating the organization of student learning during the session and exams.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ……8
ВСТУП……9
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ ……10
1.1 Хмарний сервіс ……10
1.2 Telegram API ……12
1.3 Telebot……14
1.4 Хмарний сервіс Heroku……16
1.5 Інструменти розробки……17
1.5.1 Мова програмування……17
1.5.2 Середовище розробки……18
1.5.3 Хмарний сервіс……19
1.6 Постановка завдання ……19
2 РОЗРОБКА ПРОГРАМНОЇ ЧАСТИНИ ПРОЕКТУ……20
2.1 Завантаження Telegram ……20
2.2 Активація BotFather……21
2.3 Створення власного чат-боту ……23
2.4 Зв'язування чат-боту з програмним забезпеченням ……26
2.5 Ініціалізація чат-бота ……26
2.6 Створення команд чат-бота ……28
2.7 Імплементація команди збереження файлів ……30
2.8 Імплементація команди показати всі файли……33
2.9 Імплементація команди конвертувати текст……34
2.10 Імплементація команди надіслати Email……37
2.11 Публікація чат-боту на хмарний сервіс ……42
3 ТЕСТУВАННЯ ПРОЕКТУ……44
ВИСНОВКИ……56

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ……..…….57
ДОДАТОК А. Графічна частина кваліфікаційної роботи……..…….58

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API – описання методів за допомогою яких одна програма може комунікувати з іншою (анг., Application programming interface);
IaaS – Інфраструктура як послуга (анг., Infrastructure-as-a-Service);
HTTPS – розширення протоколу HTTP для підтримки шифрування з метою підвищення безпеки (анг. HyperText Transfer Protocol Secure);
MacOS – Пропрієтарна операційна система компанії Apple;
PaaS – Платформи як послуга (анг., Platforms-as-a-Service);
SaaS – Програмне забезпечення як послуга (анг., Software-as-a-Service).

ВСТУП

Актуальність роботи визначається розвитком кіберфізичного цифрового простору та завданням щодо розвитку хмарного кіберсервісу та підвищення якості освітнього процесу. Практичний аспект також полягає в прискоренні навчального процесу студентів. Через дистанційне навчання та велику кількість онлайн-сервісів складність управління файлами між ними зростає.

Щоб вирішити ці проблеми, ви можете використовувати чат-бота в мобільному телефоні. Це перше і логічне рішення, яке може бути. Телефон майже завжди з людиною, користувач може конвертувати текст у будь-який файл і відправляти його на будь-яку електронну адресу за допомогою чат-бота, але це рішення має деякі недоліки. По-перше, вам потрібне найменше підключення до Інтернету. По-друге, чат-бот може лише конвертувати текст у певний файл. Тобто, якщо вам потрібно зробити презентацію, наприклад, чат-бот не підтримує такий функціонал. Таким чином чат-бот може прискорити роботу з файлами, але не вирішити всі описані проблеми.

У даній кваліфікаційній роботі пропонується створити хмарний сервіс на основі чат-бота для зберігання, сортування та передачі файлів електронною поштою. Користувач зможе самостійно конвертувати текст у файл, сортувати файли та відправляти їх на будь-яку електронну пошту.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Хмарний сервіс

Хмарні сервіси - це мережа потужних комп'ютерів - серверів, які дозволяють клієнтам користуватися своїми ресурсами через інтернет: зберігати файли і обмінюватися ними, працювати в онлайн-офісах, проводити обчислення. Хмарна мережа складається із вузлів зберігання інформації, які називаються дата-центрами. Це цілі будівлі, заповнені величезними шафами із серверним обладнанням. Вони розташовані по всьому світу та пов'язані через інтернет. Будують та обслуговують обладнання хмарні провайдери. Вони ж розподіляють ресурси залізних серверів на окремі віртуальні машини і здають їх в оренду.

У вузькому розумінні хмарні сервіси – це онлайн-програми, які допомагають організувати віддалену роботу та вирішувати бізнес-завдання. Співробітники отримують доступ до загальної бази даних з будь-якої точки світу та можуть керувати проектами. Кожен працівник бачить результат у реальному часі, може вносити коментарі, правки та виконувати персональні чи спільні завдання. Приклад хмарного сервісу (рис. 1.1.).

Рисунок 1.1 – Схема як працює хмарний сервіс

Є три типи хмарних сервісів: приватні, публічні, гібридні.

Приватна хмара належить одній компанії та працює на її обладнанні. Ним користуються лише співробітники. Вся інформація залишається всередині, її простіше контролювати та захищати. Але приватні хмари можуть дозволити собі лише великі компанії, тому що вони дорогі у використанні: потрібно купувати чи орендувати обладнання, самим обслуговувати та адмініструвати його.

Публічна хмара містить та обслуговує провайдер, який здає різним клієнтам обчислювальні потужності в оренду. Бізнес може закупити рівно стільки ресурсів, скільки потрібно для роботи та зберігання файлів. Це зручно та вигідно. Коли ми говоримо про використання хмарних сервісів, найчастіше маємо на увазі саме публічні хмари.

Гібридна хмара - коли частина роботи знаходиться в публічній хмарі, а інша - в приватній хмарі або взагалі на фізичних носіях. Таке буває, наприклад, у процесі поступового переходу компанії від традиційної інфраструктури до хмарної.

Існує три найпопулярніші види послуг, які надають хмарні сервіси: IaaS - Infrastructure as a Service, PaaS - Platform as a Service, SaaS - Software as a Service.

IaaS - інфраструктура як послуга, коли бізнес орендує віртуальні обчислювальні ресурси та сховища. Наприклад, клієнт може використовувати віртуальний комп'ютер на MacOS зі свого Windows-пристрою, щоб використовувати програми, які є тільки для MacOS. При цьому підтримкою та обслуговуванням пристроя займається провайдер.

PaaS – платформа як послуга. Насамперед, це продукти для розробників, наприклад програми для створення та тестування додатків, віртуальні бази даних, системи машинного навчання та обробки великих даних.

SaaS – програмне забезпечення як послуга – це готові до використання програми та сервіси, що не потребують встановлення, обслуговування та оновлення з боку користувача.

Для більшості бізнесів стануть у нагоді саме SaaS — рішення, тому що основна аудиторія IaaS і PaaS — це системні адміністратори та розробники, а не кінцеві користувачі на зразок менеджерів. Порівняння виду послуг хмарних сервісів (рис. 1.2.).

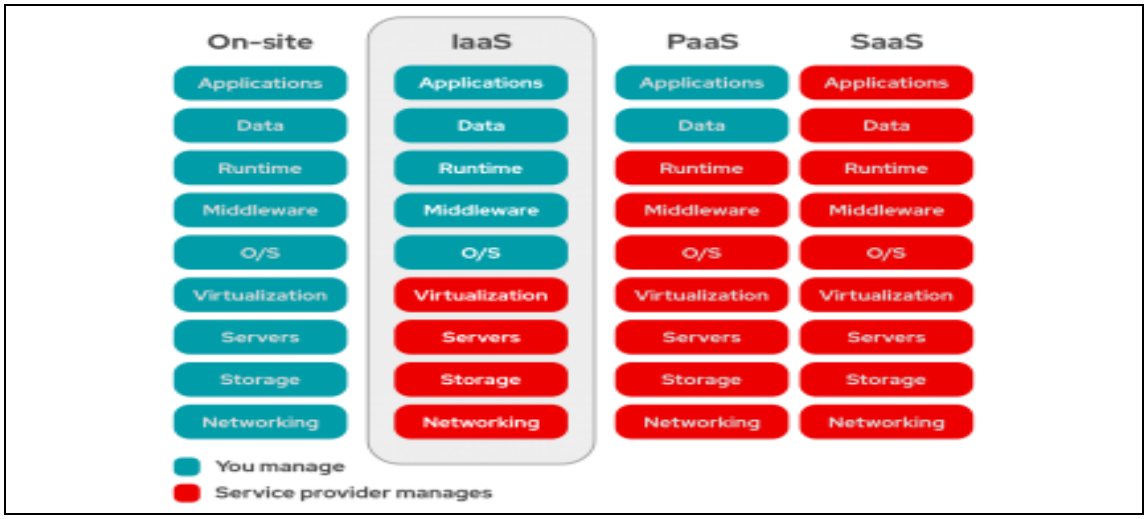


Рисунок 1.2 – Види послуг хмарних сервісів

1.2 Telegram API

Боти Telegram — це спеціальні облікові записи, для налаштування яких не потрібен додатковий номер телефону. Ці облікові записи служать інтерфейсом для коду, що виконується десь на хмарному сервесі.

Щоб використовувати Telegram API, не потрібно нічого знати про те, як працює протокол шифрування MTProto — проміжний сервер оброблятиме все шифрування та зв’язок із Telegram API. Бот спілкуєтеся з цим сервером через простий HTTPS-інтерфейс, який пропонує спрощену версію Telegram API (рис. 1.3).

Користувачі можуть взаємодіяти з ботами двома способами: Надсилати повідомлення та команди роботам, відкриваючи з ними чат або додаючи їх до груп. Або надсилати запити можна безпосередньо з поля введення, ввівши @username бота та запит. Це дозволяє надсилати вміст із вбудованих ботів безпосередньо в будь-який чат, групу чи канал. Повідомлення, команди та запити, надіслані користувачами, передаються програмному забезпеченню, що працює на хмарних серверах.

Використовуйте команду /newbot, щоб створити нового бота. BotFather запитає у вас ім’я та ім’я користувача, а потім згенерує маркер автентифікації для вашого нового бота. Ім’я вашого бота відображається в контактних даних та в інших місцях. Ім’я користувача – це коротке ім’я, яке використовується у згадках і посиланнях t.me. Імена користувачів мають довжину від 5 до 32 символів і не враховують регістр, але можуть містити лише латинські символи, цифри та символи підкреслення. Ім’я користувача вашого бота має закінчуватися на «бот», напр. "tetris_bot" або "TetrisBot". Маркер — це рядок у вигляді 110201543:AAHdqTcvCH1vGWJxfSeofSAs0K5PALDsaw, необхідний для авторизації бота та надсилання запитів до API бота. Маркер треба тримати у безпеці та зберігати його безпечно.

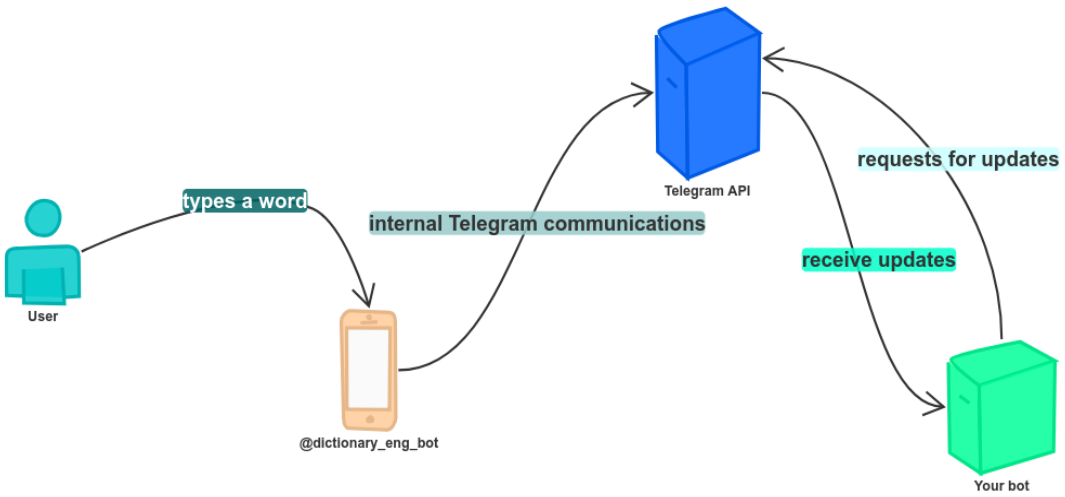


Рисунок 1.3 – Схема роботи Telegram API

1.3 Telebot

Telebot – це фреймворк для роботи з Telegram API. Цей пакет надає найкращий у своєму роді API для маршрутизації команд, вбудованих запитів і клавіатур, а також зворотних викликів. Telebot перевершує конкурентів через такі функції:

- справжній стислий API;
- маршрутизація команд;
- проміжне програмне забезпечення;
- API прозорого файлу;
- прості зворотні виклики бота.

Усі методи Telebot API надзвичайно прості в використанні. Крім того, Telebot готовий до високого навантаження. Telebot підтримує асинхронну роботу з чат-ботом. Завдяки цьому чат-бот може спілкуватися з багатьма користувачами одночасно. Навіть якщо операції занадто великі.

Система маршрутизації Telebot піклується про доставку оновлень до їхніх кінцевих точок, тому, щоб отримати обробку будь-якої важливої події, все, що потрібно зробити, це підключити свою функцію до однієї з наданих Telebot кінцевих точок. Ці атрибути допомагають боту розуміти, що саме необхідно зробити. Існують десятки підтримуваних кінцевих точок. Ця система повністю розширювана, тому користувачі можуть перетворювати їх на власні методи, не порушуючи зворотну сумісність.

Telebot має простий і впізнаваний спосіб налаштування проміжного програмного забезпечення - зв’язаних функцій із доступом до контексту, які викликаються перед виконанням обробника. Контекст – це особливий тип, який охоплює величезну структуру оновлення та представляє контекст поточної події. Він надає кілька помічників, які дозволяють отримати, наприклад, чат, у який було надіслано це оновлення, незалежно від того, яке це оновлення та від якого користувача.

Telebot розуміє, як ви надаєте йому вхідні оновлення, якщо ви налаштуєте його за допомогою Poller або викликаєте ProcessUpdate для кожного оновлення.

Під час обробки команд Telebot підтримує як прямий, так і груповий синтаксис і ніколи не доставлятиме повідомлення, адресовані іншому боту, навіть якщо режим конфіденційності вимкнено.

Для спрощеного глибокого посилання Telebot також витягує корисне навантаження.

Telebot дозволяє як завантажувати файли з диска, так і завантажувати з Telegram файли в межах бота. Крім того, надсилання будь-якого медіа-файлу, створеним із диска, автоматично завантажить файл у Telegram. Тому користувачу непотрібно робити додаткових дій для завантаження файлів.

Також Telebot можна з’єднати з хмарним сховищем. Це дозволить прискорити роботу чат-боту, тому що обчислювальні процеси будуть виконуватися на потужних серверах, що дозволить виконувати великий обсяг роботи паралельно. Вся інформація зберігається в безпечному місці, до якої немає доступу у третіх осіб. Будь-яке хмарне сховище підтримує резервне копіювання, тому в разі виникнення проблеми можна відновити доступ до файлів без великих проблем. Можливість одночасного доступу до хмарного сховища дає змогу об’єднувати багато сервісів в одну мережу (рис. 1.4).

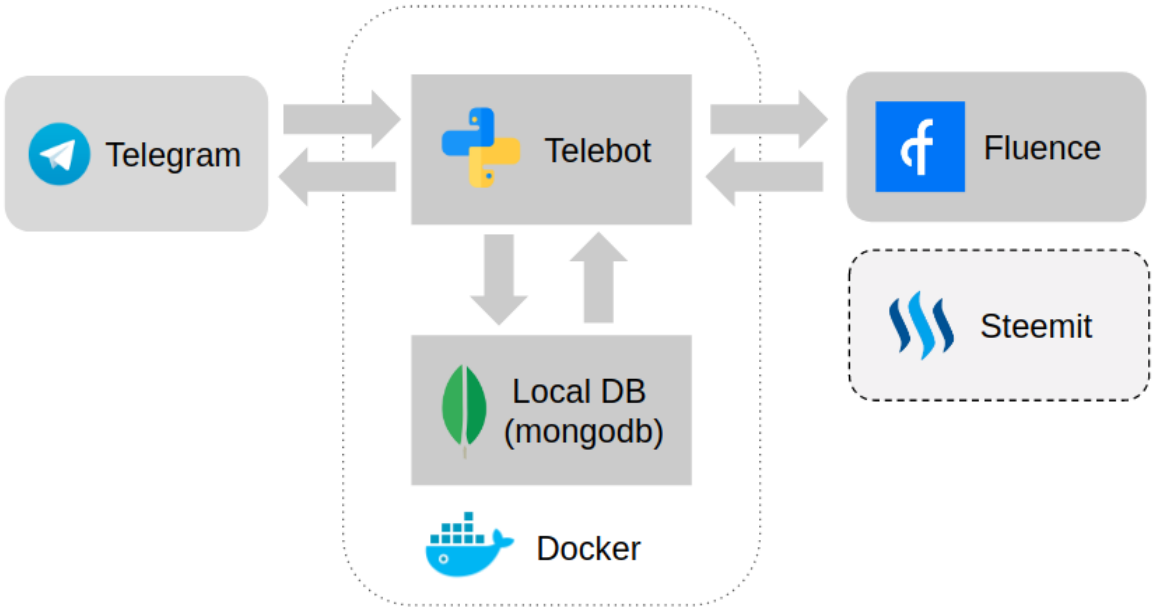


Рисунок 1.4 – Схема з’єднання з хмарним сховищем

1.4 Хмарний сервіс Heroku

Heroku – Platform as a Service. Це означає, що платформа працює як сервіс: надає користувачеві певні функції та можливості, доступ до систем та програмного забезпечення. При цьому її інфраструктура повністю прихована.

За користувача все роблять співробітники сервісу - ця робота залишається "під капотом", а багато процесів автоматизовано. За безпеку, архітектуру та налаштування сервера відповідають фахівці платформи.

Додатки, що працюють в Heroku, виконуються ізольовано від інших – вони укладені в спеціальні контейнери, які називаються диносами або діно. Диноси дозволяють створити легковажне незалежне середовище і розгорнути в ньому додаток так, щоб налаштування його середовища не конфліктували з іншими. Одна програма може використовуватися кількома диносами, і проект легко масштабується під завдання розробника.

Диноси мають шаблони – прототипи, на основі яких створюється контейнер, як деталь по кресленню. Саме завдяки їм програми в Heroku легко

масштабувати. Кожен тип процесу відповідає за свою частину роботи та не зачіпає інших модулів. Це допомагає паралелізму: процеси поділяються і завдання не поєднуються. Так можна уникнути конфліктів. Діноси легко масштабувати. Якщо програма потребує більше ресурсів, можна збільшити робочі потужності в кілька кліків. Для цього потрібно додати необхідну кількість нових диносів з такими ж типами процесів, як у тих, що використовуються до того.

Для додавання, розгортання та запуску програми достатньо ввести декілька команд у консолі. Тривала підготовка та попереднє налаштування не потрібні. Працювати з сервісом може фахівець-початківець. Також використання Heroku заощаджує час розробника при запуску та масштабуванні нового проекту.

Серед проектів Heroku – власна СУБД SQL database as a service, програмне забезпечення для зв'язку команди розробників між собою, сервіси автоматизації для програм різними мовами та багато іншого. Платформа працює і з noSQL-рішеннями.

1.5 Інструменти розробки

Перед тим як створювати хмарний сервіс на основі чат-боту чи навіть розглядати його майбутню структуру, необхідно вирішити низку запитань та визначитися із платформою, на якій буде використовуватися забезпечення. Після цього обрати відповідну мову програмування серед множини існуючих мов та середовище розробки, як робочий простір розробника.

1.5.1 Мова програмування

Для розробки програмного забезпечення було обрано Python. Він підтримує пакети модулів, що сприяє повторному використанню коду.

Python відносно нова мова програмування, тому Python має легкий та зрозумілий синтаксис. Також один із плюсів – це розширюваність. Існують бібліотеки та фреймворки під будь-який тип завдань та потреб. Також величезним плюсом є те, що ми можемо використовувати C-код з Python

Python безкоштовно доступний для використання розробниками, викладачами і студентами за ліцензією на поширення ПЗ з відкритим вихідним кодом.

1.5.2 Середовище розробки

Оскільки мовою програмування обрано Python, то було обрано PyCharm як середовище розробки.

PyCharm складається з набору інструментів, які розробники використовують для створення різноманітних додатків, наприклад, сайтів, GUI додатків, Штучного інтелекту та Backend. Використовуючи PyCharm можна керувати всім робочим процесом розробки: від створення програми до тестування та оптимізації.

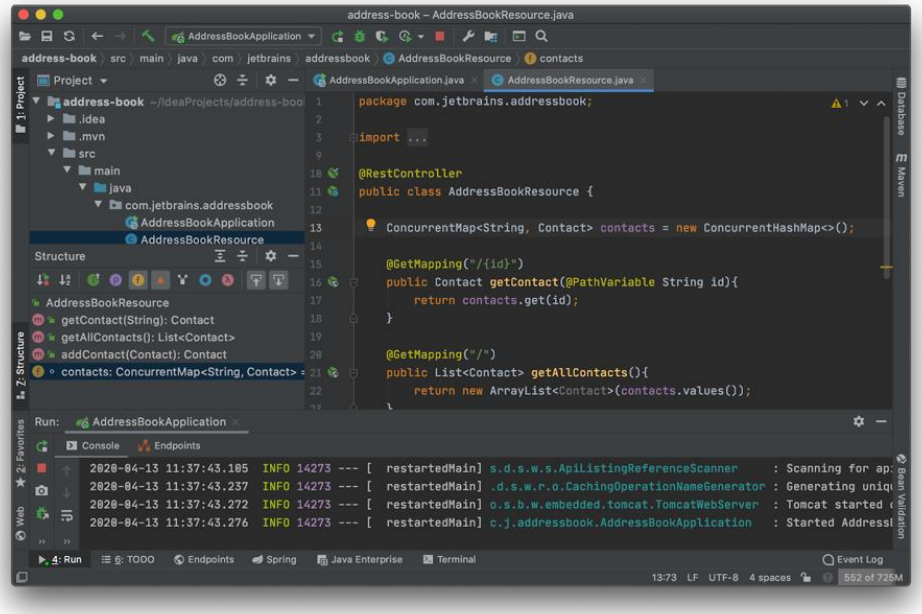


Рисунок 1.5 – Інтерфейс PyCharm

PyCharm допомагає писати зрозумілий код, який легко підтримувати. IDE контролює якість коду за допомогою перевірок відповідності вимогам, розумних рефакторингів та безлічі інспекцій, а також надає допомогу при тестуванні. Також PyCharm надає широкі можливості реорганізації коду за допомогою рефакторингів Rename і Delete, Extract Method, Introduce Variable, Inline Variable, Inline Method та багатьох інших. Рефакторинги враховують особливості конкретної мови або фреймворку, допомагаючи вносити зміни по всьому проекту.

1.5.3 Хмарний сервіс

В якості хмарного сховища було обрано сервіс Heroku. Цей сервіс підходить для невеликого проекту. Тому що має різні цінові плани для розробників. А також цей хмарний сервіс легко масштабувати.

1.6 Постановка завдання

В Структурі кіберуніверситету є хмарні кібер-сервіси для безпаперового електронного документообігу. Чат-бот на основі хмарного сервісу є одним із них.

Об’єкт дослідження у роботі – Чат-бот на основі хмарного сервісу у складі кіберуніверситету.

Предмет дослідження – моделі, методи та процедури оптимізації безпаперового електронного документообігу з використанням хмарних технологій.

Мета роботи - це проектування хмарного сервісу для файлової системи на основі чат-боту в рамках реалізації хмарних сервісів кіберуніверситету для полегшення якості навчального процесу студента.

2 РОЗРОБКА ПРОГРАМНОЇ ЧАСТИНИ ПРОЕКТУ

2.1 Завантаження Telegram

Для створення чат-боту в Telegram необхідно завантажити додаток на мобільний пристрій. Telegram є кросплатформним додатком та доступний на будь-якій платформі. На Iphone необхідно відкрити AppStore та перейти на вкладку «Пошук». В пошуковому вікні введемо «Telegram» та списку мобільних додатків виберемо Telegram (рис.2.1). На екрані профілю мобільного додатку (рис.2.2) необхідно натиснути на кнопку «Завантажити». Після того як Telegram був завантажен на мобільний пристрій необхідно авторизувати свій особистий профіль в ньому.

Рисунок 2.1 – Результат пошуку в AppStore

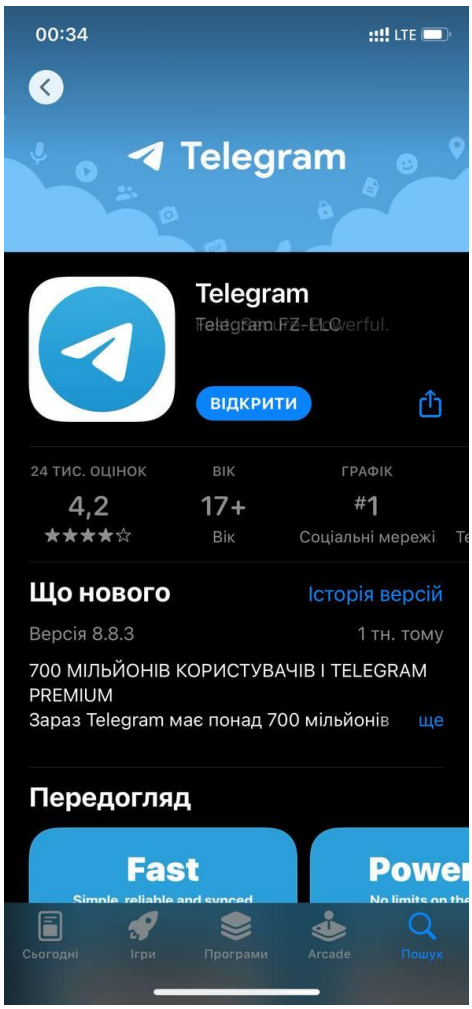


Рисунок 2.2 – Екран профілю мобільного додатку

2.2 Активація BotFather

В пошукову вікні введемо текст «BotFather» - це головний бот в телеграм через якого користувачі можуть створювати власних чат-ботів. Головний чат-бот помічений офіційною галочкую, тому його легко знайти серед результатів пошуку (рис.2.3). Натиснемо на BotFather та перейдемо на екран чат-боту (рис.2.4). На екрані натиснемо команду «start». Після цього запускається процесс створення чат-боту.

Головний бот буде покроково показувати підказки на екрані. Також во вкладці «Menu» будуть відображатися усі доступні команди для створення та

конфігурації власного чат-боту (рис.2.5).

Рисунок 2.3 – Результат пошуку в Telegram

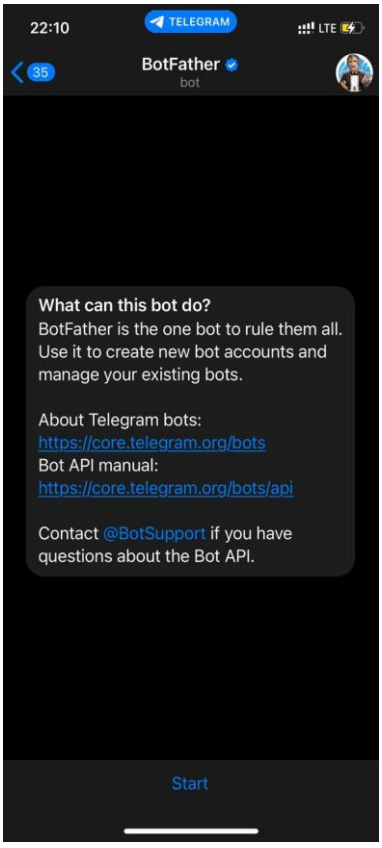


Рисунок 2.4 – Інтерфейс BotFather

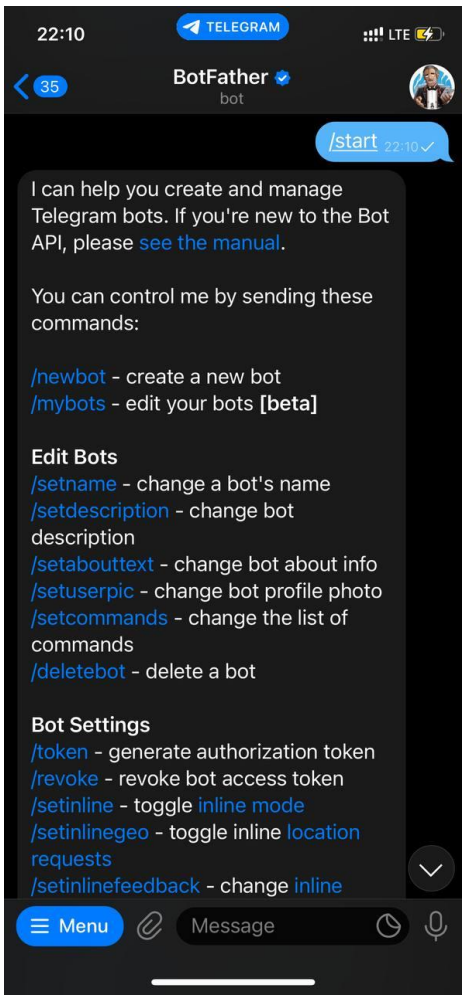


Рисунок 2.5 – Доступні команди

2.3 Створення власного чат-боту

Виберемо команду «/newbot» для створення нового чат-боту. BotFather за допомогою Telegram API власноруч сгенерує нового чат-бота. Далі необхідно вибрати нікнейм для чат-боту. Натиснемо команду «/setname» та введемо нікнейм (рис.2.6). Встановим нікнейм «FileManagerNureBot». Нікнейм повинен бути унікальним, якщо вже існує бот с таким нікнеймом, то буде необхідно обрати інший, який ще не використовується. Також необхідно задати унікальний ідентифікатор для бота. Встановимо унікальний ідентифікатор «File_Manager_NURE_Bot» за допомогою команди «/setdescriprion» (рис.2.7). Також необхідно сгенерувати токен доступ до чат-

боту (рис. 2.8).

Для кожного чат-боту існує окремий токен доступу. Токен доступу виступає унікальним ідентифікатором чат-бота. Тільки через нього Telegram API розуміє з яким чат-ботом треба вхзаємодіяти та обробляти команди.

Токен доступу не повинен бути в публічному доступі. Його треба зберігати безпечному місці, а також не поширювати третім особам.

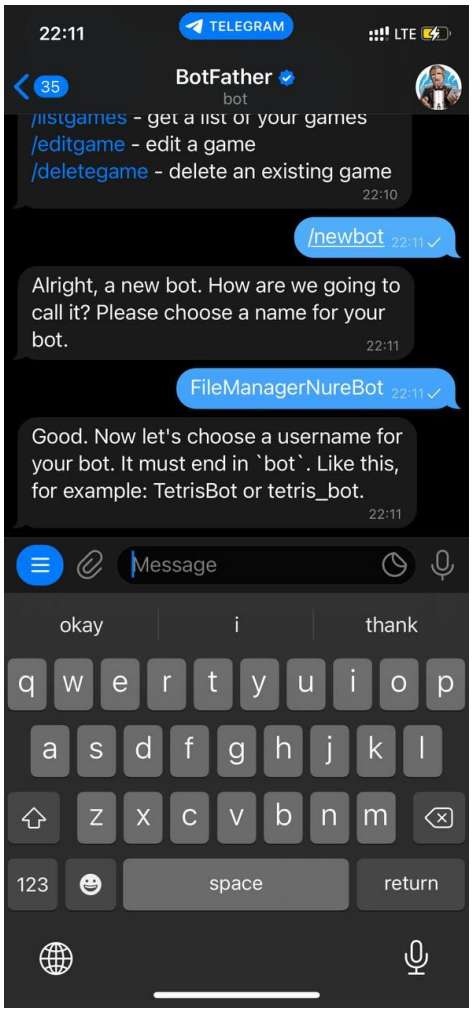


Рисунок 2.6 – Команда встановлення нікнейму

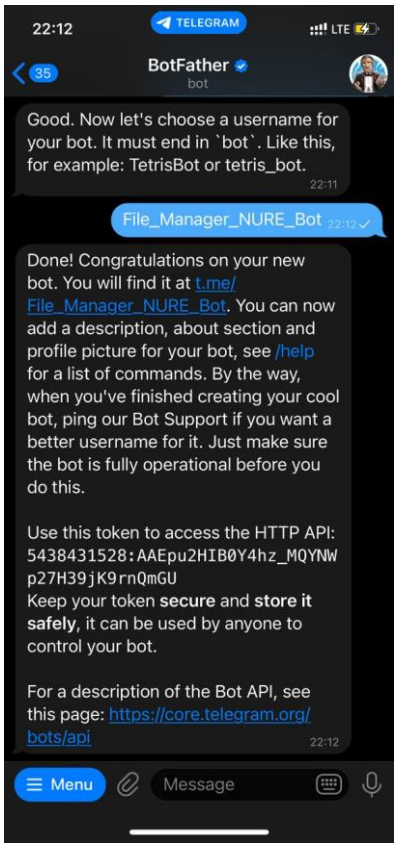


Рисунок 2.7 – Команда встановлення унікального ідентифікатора.

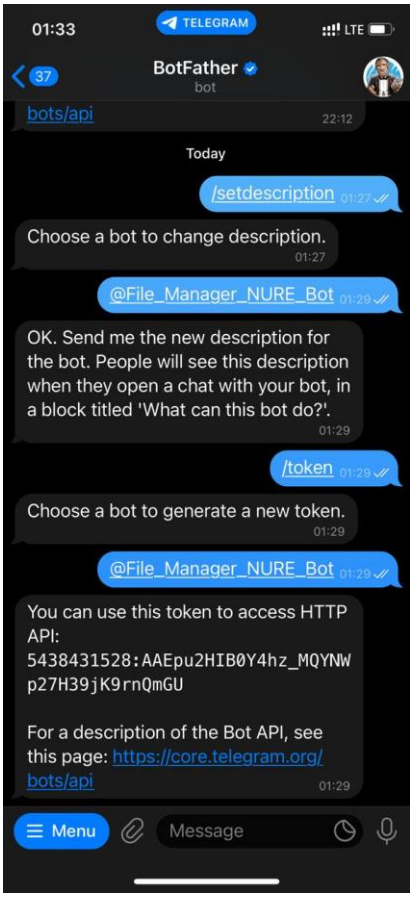


Рисунок 2.8 – Команда створення токену доступа

2.4 Зв'язування чат-боту з програмним забезпеченням

Сгенерований токен доступу необхідно з'єднати с проектом. Токен доступу необхідний для того, щоб сервіс знав з яким саме ботом взаємодіяти. Для цього необхідно створити Python проект та з'єднати його з чат-ботом. Відкриємо термінал та за допомогою команди «$ pip install pyTelegramBotAPI» інсталюємо необхідну бібліотеку для роботи з чат-ботом (рис.2.9). Після того як бібліотека була інстальована, додамо токен доступу в проект за допомогою команди «bot = telebot.TeleBot("TOKEN", parse_mode=None)» (рис. 2.9).

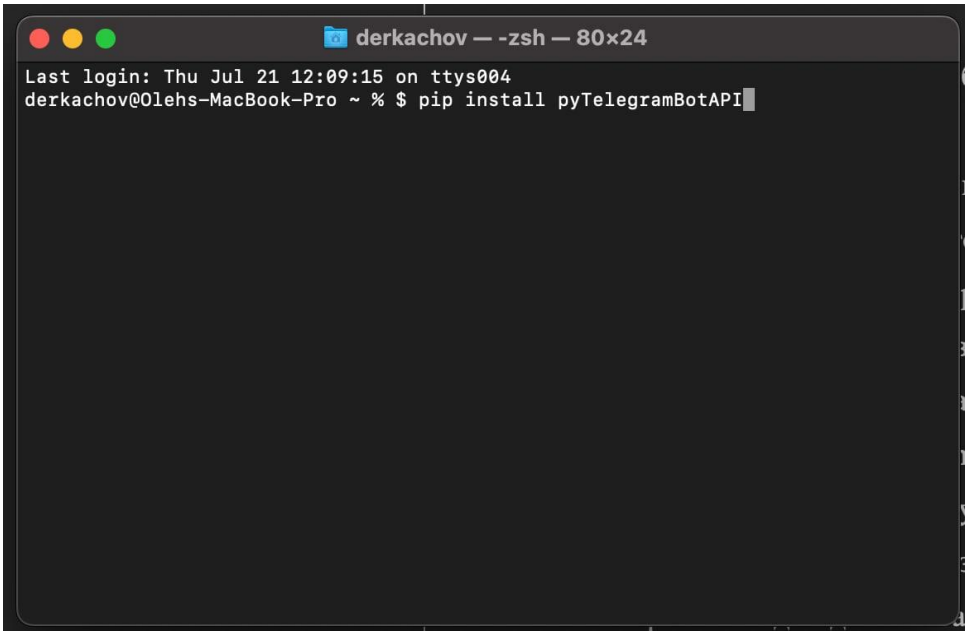


Рисунок 2.9 – Команда в терміналі

2.5 Ініціалізація чат-бота

Після того, як токен доступу було додано до проекту. Треба ініціалізувати проект. Для цього необхідно імпортувати бібліотеку з командами для роботи з чат-ботом. Це дозволить взаємодіяти с Telegram API за допомогою спеціальних команд.

Лістинг 2.1 – імпорт необхідної бібліотеки

```
import telebot
from telebot import types
```

Бібліотека Telebot містить усі необхідні методи та реалізації для повного циклу розробки чат-боту. Це дозволяє розробляти конкурентоспроможні чат-боти для усіх галузей. Від найпримітивніших чат-ботів з текстовими командами до чат-ботів поєднаних с різноманітними API та штучним інтелектом.

Далі необхідно ініціалізувати токен доступу чат-бота з програмним забезпеченням. Це дозволить взаємодіяти програмному забезпеченню с конкретним чат-ботом. Також необхідно зробити начальний метод для початку роботи чат-боту.

Лістинг 2.2 – Імплементація начального методу

```
bot = telebot.TeleBot(TOKEN)

@bot.message_handler(commands=['start'])
def startAction(message):
    welcome = 'Вітаю'
    description = 'Цей бот допомогає полегшити роботу з файлами та електронною поштою. ' \
                  'Введіть команду /commands для того, щоб побачити список усіх команд'

    bot.send_message(message.chat.id, f"{welcome}, {message.from_user.first_name}! {description}", )
```

Метод startAction() позначений атрибутом @bot.message_handler означає, що цьому методі відбувається обробка подій, а аргумент commands означає саме яку подію обробляє цей метод. В цьому випадку аргумент дорівнює start – це означає, що обробляється начальна подія, коли користувач вперше взаємодіє с чат-ботом. За допомогою команди message.from_user.first_name ми дізнаємось ім’я користувача.

Bot.send_message надішле привітальне повідомлення користувачу з підказками (рис. 2.10).

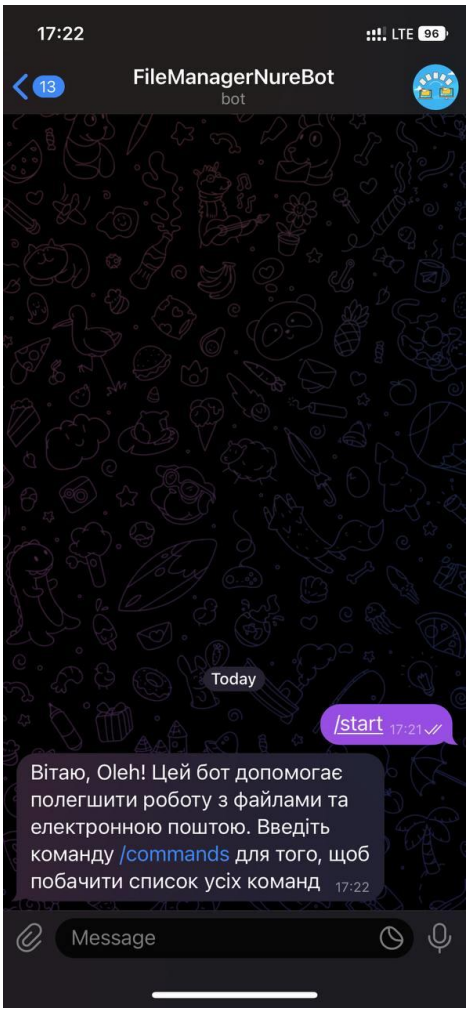


Рисунок 2.10 – Начальна команда «/start»

2.6 Створення команд чат-бота

Для взаємодії користувача з чат-ботом необхідно створити команди, за допомогою яких бот буде розуміти що саме необхідно користувачу.

Лістинг 2.3 – Створення команд

```
@bot.message_handler(commands=['commands'])
def commandsList(message):
```

```
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True, row_width=1)
    saveFile = types.KeyboardButton("Зберегти файл")
    showFiles = types.KeyboardButton("Показати всі файли")
    convertText = types.KeyboardButton("Конвертувати текст")
    sendEmail = types.KeyboardButton("Надіслати Email")
    markup.add(saveFile, convertText, showFiles, sendEmail)
    bot.send_message(
                    message.chat.id,
                    'Список команд',
                    reply_markup=markup
    )
```

В функції commandsList(), позначеною атрибутом @bot.message_handler, створюються команди для взаємодії з чат-ботом. В аргументах функції передається один аргумент commands, який означає, що функція поверне список команд.

Клас KeyboardButton створює нову команду та кнопку у меню чат-бота. Змінна saveFile створює нову команду для зберігання файлу в сховище. Змінна showFiles виводить усі файли, які були збережені в сховище чат-боту. Змінна convertText конвертує введений текст у певний файл. Змінна sendEmail надсилає Email с текстом або файлом на певну електронну адресу (рис. 2.11).

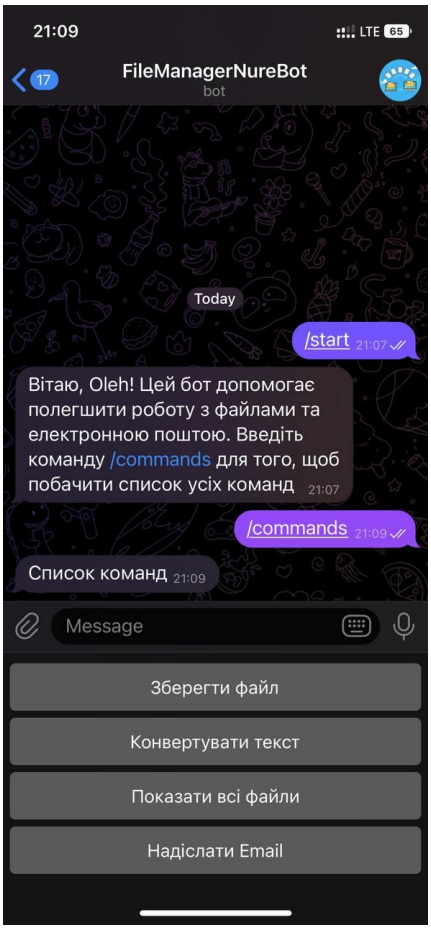


Рисунок 2.11 – Список команд чат-боту

2.7 Імплементація команди збереження файлів

Користувач має змогу зберегти файл, за допомогою команди «Зберегти файл», до глобального сховища файлів та мати доступ до цих файлів з будь-якого пристрою. Ці файли зберігаються в хмарному сховище Telegram. Якщо на пристрої користувача немає достатньо місця у сховище пристрою, то це ідеальне рішення.

Лістинг 2.4 - Функція збереження файлів

```
bot.message_handler(content_types=['document'])
def saveFile(message):

    file_name = message.document.file_name

    file_info = bot.get_file(message.document.file_id)

    downloaded_file = bot.download_file(file_info.file_path)
```

```
with open(file_name, 'wb') as new_file:
    new_file.write(downloaded_file)

bot.send_message(message.chat.id, f"{file_name} успішно збережено!")
```

Функція saveFile(), позначена атрибутом @bot.message_handler, приймає аргумент типу document. Спочатку функція обробляє файл, далі ініціалізує необхідні данні. Змінна file_name приймає ім’я файлу з метаданих. Змінна file_info приймає унікальний ідентифікатор файлу. Далі функція зберігає файл до тимчасової змінної downloaded_file. А за допомогою методу with open зберігає файл до сховища чат-боту. Після успішного завантаження файлу користувач отримає повідомлення за допомогою метода send_message.

Виберемо команду «Зберегти файл» в меню команд чат-боту (рис.2.12).

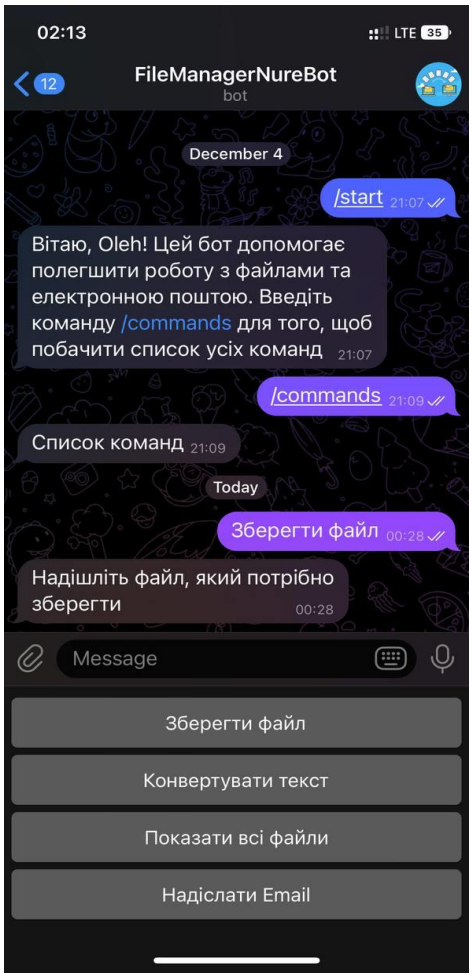


Рисунок 2.12 – Відповідь чат-боту на команду «Зберегти файл»

Після того як користувач відправив команду, йому слід виконувати підказки чат-боту. Далі треба надіслати файл, який потрібно загрузити до хмарного сховища.

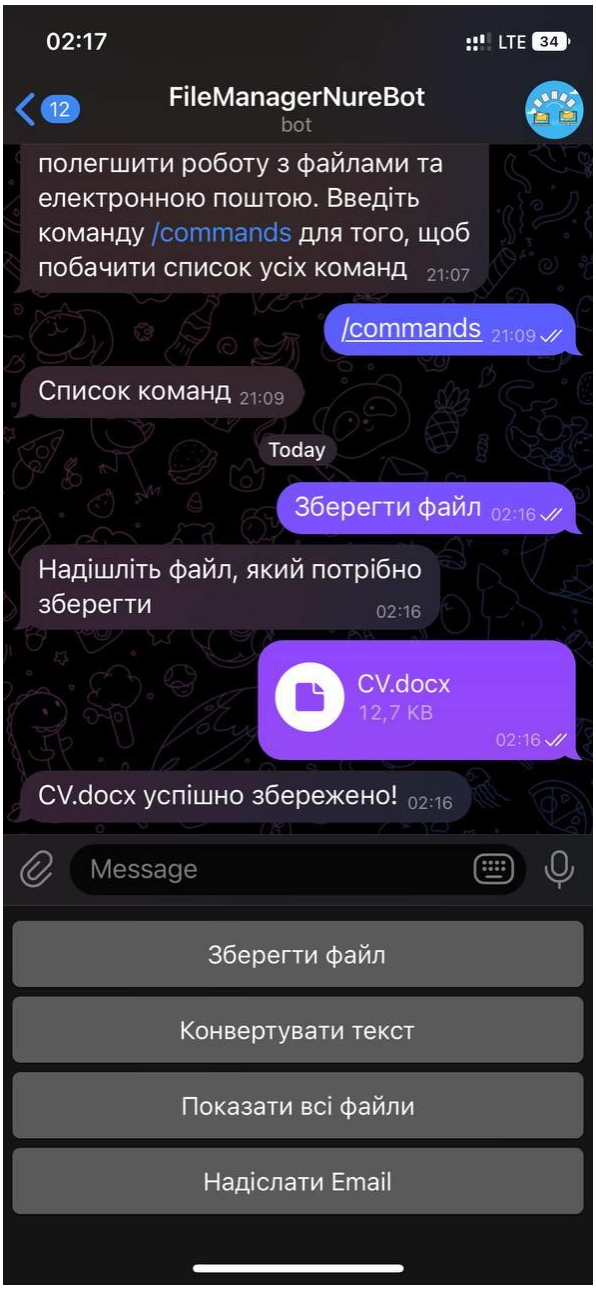


Рисунок 2.13 – Результат збереження файлу

Після того, як користувач відправив файл, чат-бот обробляє його та повідомляє про успішне збереження файлу до хмарного сховища (рис.2.13).

2.8 Імплементація команди показати всі файли

За домопогою команди «Показати всі файли» користувач має змогу отримати список всіх документів завантажених до хмарного сховища. Далі користувач має змогу завантажити файли на всій пристрій або надіслати файли іншим користувачам.

Лістинг 2.5 - Функція показу всіх документів

```
def showAllFiles(message):
    filesPath = "/Users/derkachov/Desktop/Projects/pythonProject/Files"
    os.chdir(filesPath)

    for file in glob.glob("*"):
        encodedFile = open(file, encoding="ISO-8859-1")

        bot.send_document(message.chat.id, encodedFile)
```

Функція showAllFiles() приймає атрибут message. В цій функції реалізован функціонал пошуку файлів в хмарному сховище, декодування файлу до спеціального формату чат-боту. А також відправка файлів користувачу.

Для змінної filesPath задається параметр, в якому прописан шлях до директорії, де зберігаються файли. Далі за домопогою методу chdir зчитуються усі файли в директорії. Далі змінна encodedFile зчитує файл та форматує його у спеціальний формат, який підтримує чат-бот. Метод send_document надсилає вже форматований файл користувачу (рис. 2.14).

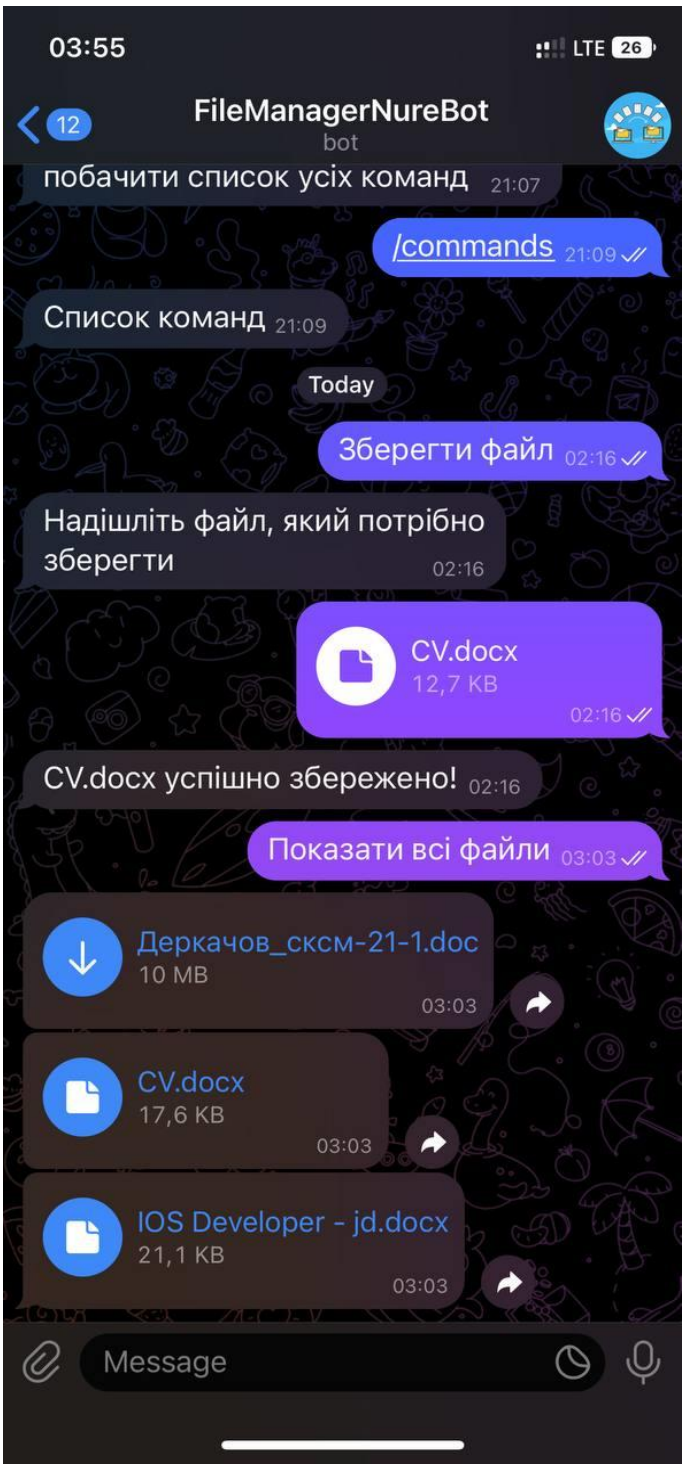


Рисунок 2.14 – Результат команди «Показати всі файли»

2.9 Імплементація команди конвертувати текст

Ця команда здатна перетворити текст у будь-який файл. Користувач має змогу написати будь яке повідомлення чат-боту, та він автоматично

перетворить цей текст у файл. Чат-бот підтримує усі необхідні типи файлів.

Коли користувач відправляє команду «Конвертувати текст», чат-бот надсилає підказки користувачу що робити далі (рис. 2.15).

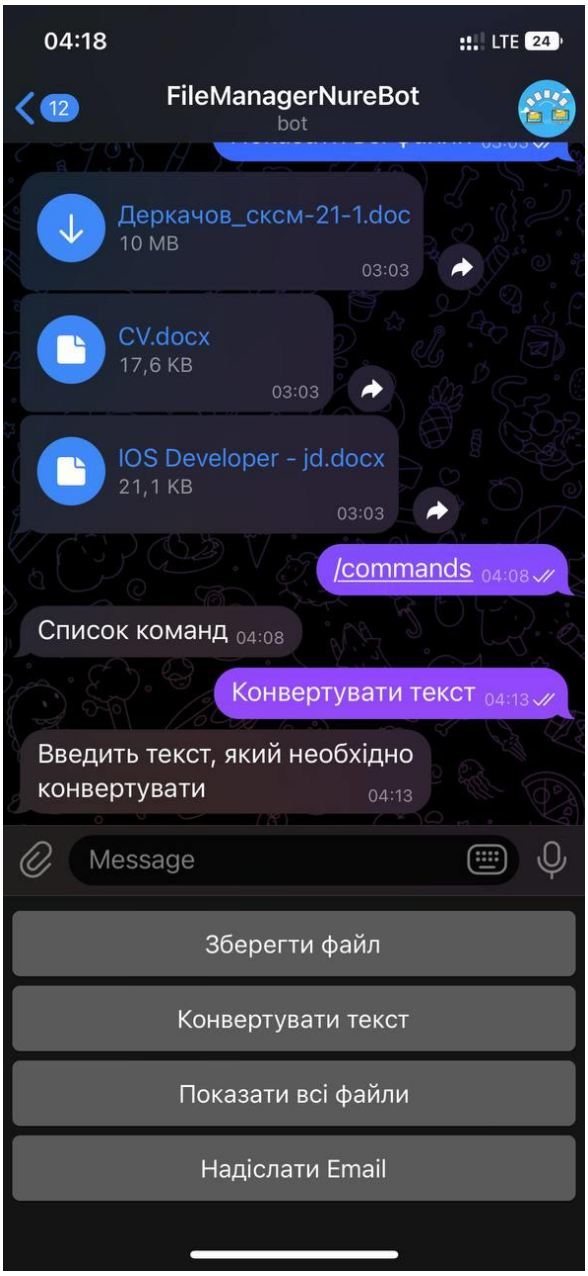


Рисунок 2.15 – Команда «Конвертувати текст»

Далі користувачу необхідно ввести текст, який потрібно перетворити

на файл. Після цього користувач надсилає текст, а чат-бот перетворює його на файл та автоматично завантажує до хмарного сховища. Користувачу приходить повідомлення, що файл успішно створено (рис. 2.16).

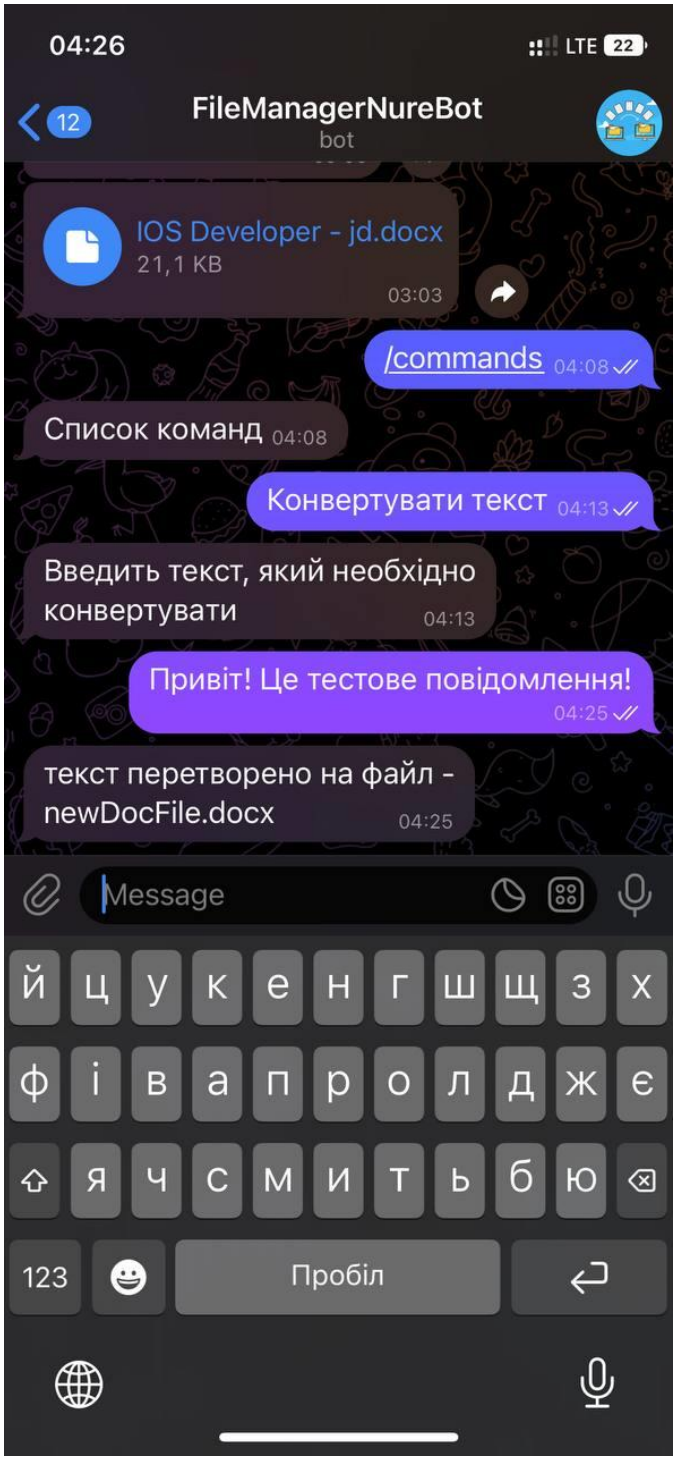


Рисунок 2.16 – Створений текстовий файл

Лістинг 2.6 - Функція перетворення тексту на документ

```
def convertText(message):
    text_file = open('/Users/derkachov/Desktop/Projects/pythonProject/Files/' + "newDocFile.docx", "w")

    text_file.write(message.text)

    text_file.close()

    bot.send_message(message.chat.id, "текст перетворено на файл - newDocFile.docx")
```

Функція convertText() перетворює введений текст у файл. За допомогою методу open() створюється новий пустий файл. Метод write() записує в новий файл текст, який написав користувач. Після цього файл треба зберегти за допомогою методу close(). Після цього файл зберігається у хмарному сховище чат-боту та буде доступний з будь-якого пристрою. Метод send_message() повідомить користувачу, що файл успішно створено та надішле детальну інформацію про файл.

2.10 Імплементація команди надіслати Email

Користувач має змогу надіслати Email за допомогою чат-боту. Будь це звичайний текст або файл, чат-бот конвертує це у електронний лист та надішле його зі свого домену. Також користувач може вибрати файл з хмарного сховища та надіслати його електронною поштою.

Коли користувач відправляє команду «Надіслати Email», чат-бот опрацьовує цю команду та надсилаю підказки користувачу (рис. 2.17).

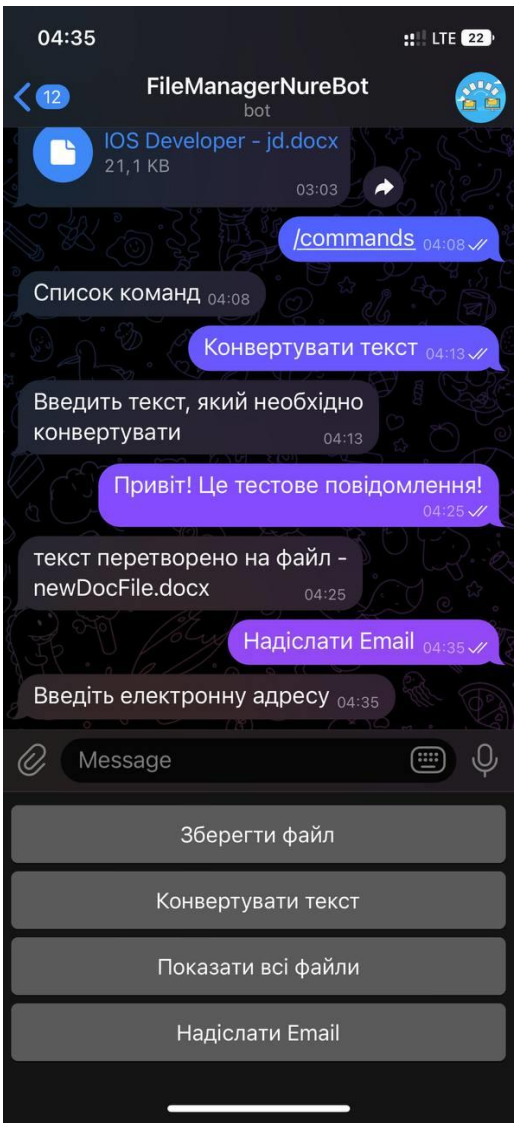


Рисунок 2.17 – Команда «Надіслати Email»

Далі необхідно ввести електронну адресу, на яку потрібно надіслати електронний лист, чат-бот опрацює електронну адресу та надішле підказки, що необхідно робити далі (рис.2.18).

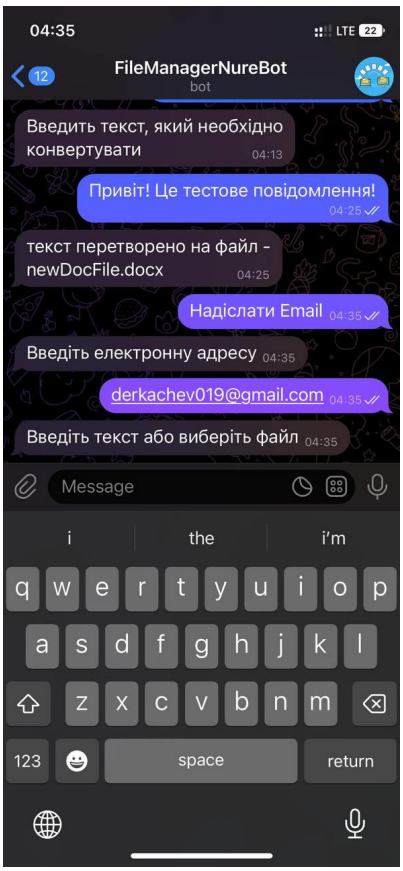


Рисунок 2.18 – Підказки чат-бота

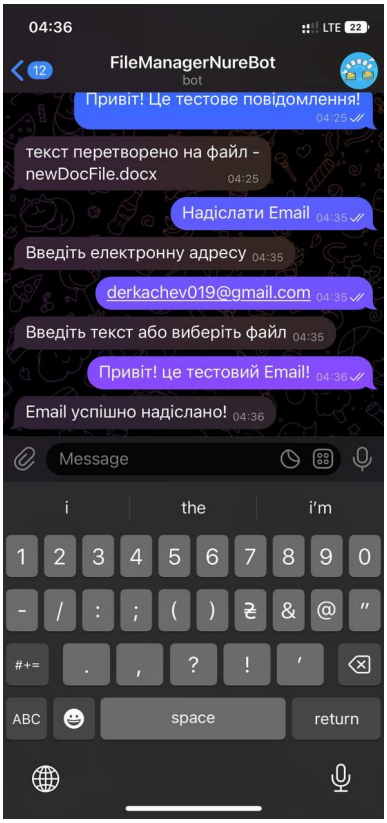


Рисунок 2.19 – Результат команди

Далі користувачу необхідно ввести текст або вибрати файл, який він хоче надіслати електронною поштою. Бот перетворить це у електронний лист та надішле email (рис. 2.19).

Після цього на електронну пошту прийде лист від імені чат-боту (рис. 2.20).

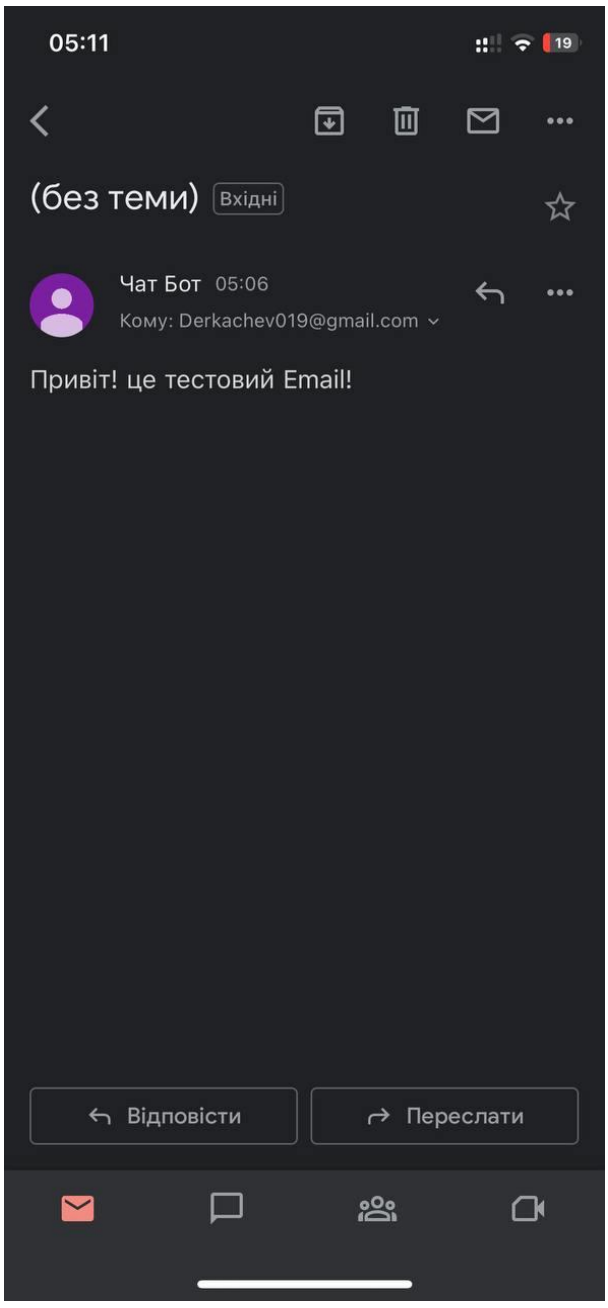


Рисунок 2.20 – Електронний лист від чат-бота

Лістинг 2.7 - Функція надсилання електронного листа

```
port = 2525
smtp_server = "smtp.mailtrap.io"
login = "LOGIN"
password = "PASSWORD"

subject = "An example of boarding pass"
sender_email = "mailtrap@example.com"
receiver_email = "RECEIVER_EMAIL"

message = MIMEMultipart()
message["From"] = sender_email
message["To"] = receiver_email
            message["Subject"] = subject

# Add body to email
body = "BODY"
message.attach(MIMEText(body, "plain"))

filename = "yourBP.pdf"

with open(filename, "rb") as attachment:
 part = MIMEBase("application", "octet-stream")
part.set_payload(attachment.read())

encoders.encode_base64(part)

part.add_header(
  "Content-Disposition",
  f"attachment; filename= {filename}",
)

message.attach(part)
text = message.as_string()

with smtplib.SMTP("smtp.mailtrap.io", 2525) as server:
  server.login(login, password)
server.sendmail(
  sender_email, receiver_email, text
)
```

В цій функції ініціалізується почтовий акаунт чат-боту. Також для надсилання електронного листа необхідно налаштувати порти зв’язку. Далі обробляється введений текст. Також обробляється електронна адреса на яку потрібно відправити електронний лист. Коли лист успішно надіслано, чат-бот інформує користувача про це.

2.11 Публікація чат-боту на хмарний сервіс

В якості хмарного сервісу було обрано сервіс Heroku. Спочатку необхідно зареєструватись на сервісі, щоб мати змогу налаштувати та підключити проект (рис. 2.21).

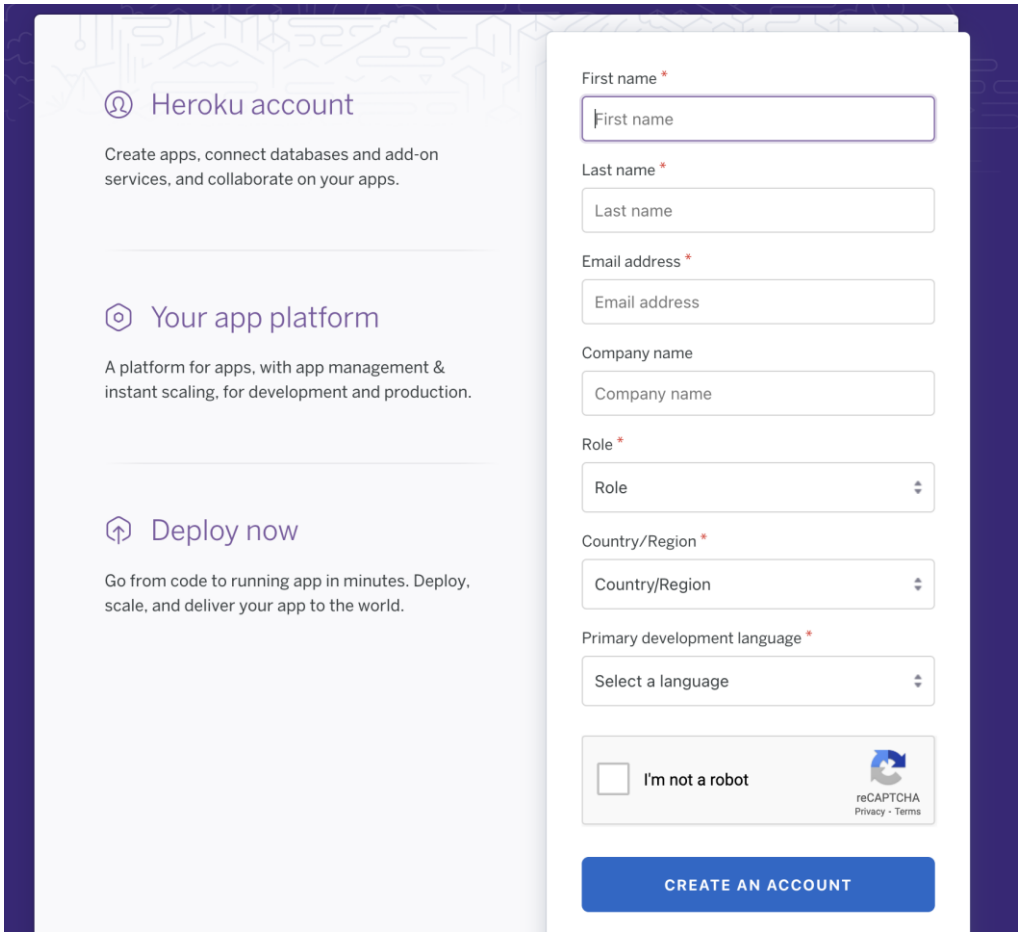


Рисунок 2.21 – Форма реєстрації на сервісі

Далі необхідно створити новий проект. Треба обрати ім’я проекту та регіон, на якому буде находитися облачний сервіс. Ім’я проекту повинно бути унікальним та складатись лише з літер, цифр та дефісу (рис. 2.22).

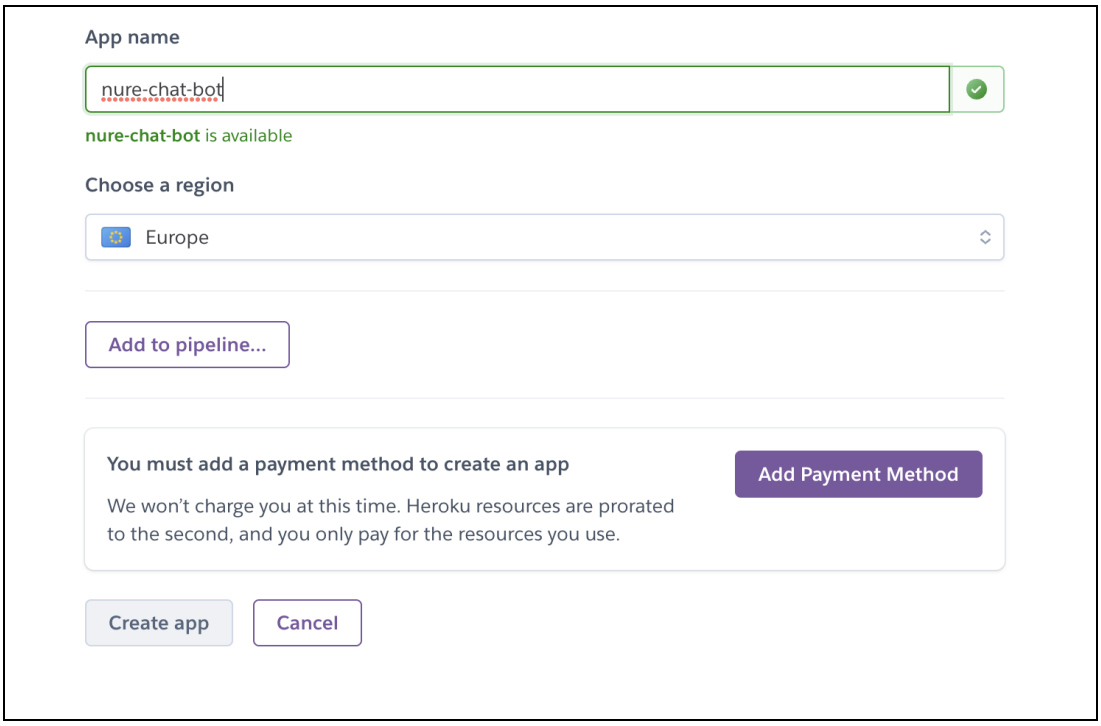


Рисунок 2.22 – Створення нового проекту

Далі необхідно обрати план передоплати та добавити платіжний метод. В якості платіжного методу можна обрати банківську карту. А тип передоплати залежить від багатьох чинників. Від масштабності проекту до кількості запитів та користувачів.

Після необхідно з’єднати чат-бот з хмарним сервісом за допомогою секретного ключа, який необхідно згенерувати на комп’ютері та додати в налаштування проекту. Після цього треба завантажити файли проекту до хмарного сервісу. Чат-бот треба запустити на хмарному сервісі та він буде працювати незалежно від типу пристрою.

3 ТЕСТУВАННЯ ПРОЕКТУ

Початкові умови не містять інформацію про користувача. Спочатку протестуємо пошук чат-бота в меню. В меню напишемо File та в списку результатах появиться чат-бот для роботи з файлами (рис. 3.1).

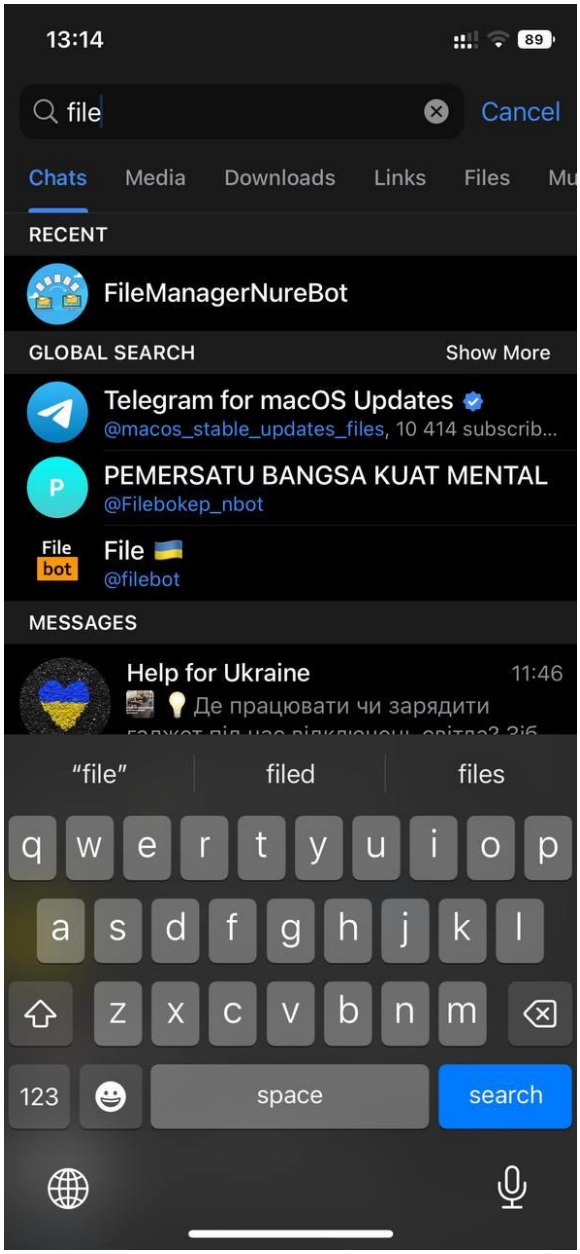


Рисунок 3.1 – Чат-бот в результатах пошуку

Виберемо чат-бот зі списку результатів. Далі виберемо команду start. Після цієї команди чат-бот почав взаємодіяти с користувачем. Чат-бот проінформує та надішле наступні підказки (рис. 3.2).

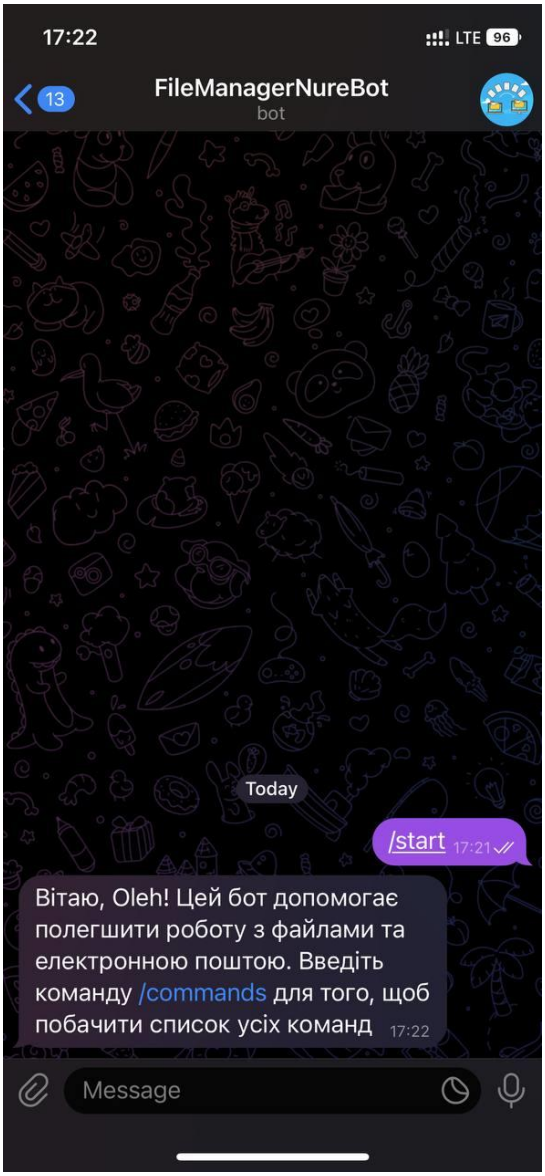


Рисунок 3.2 – Початок роботи з чат-ботом

Для того, щоб побачити список усіх команд, потрібно вибрати команду commands. За допомогою цієї команди чат-бот надішле список всіх актуальних команд доступних для користувача (рис. 3.3).

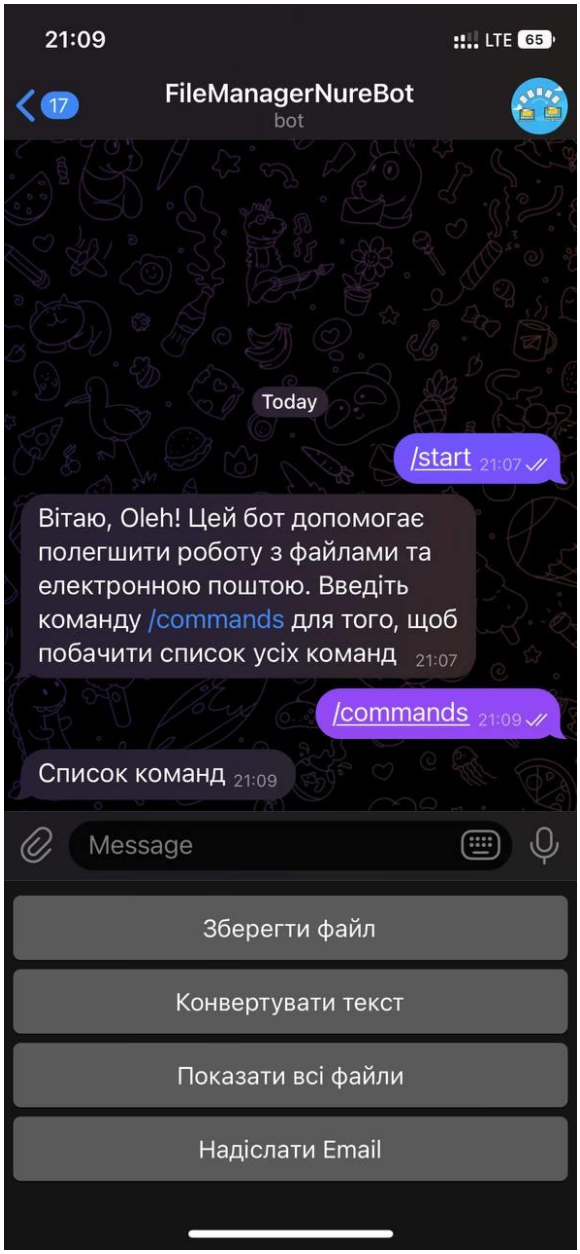


Рисунок 3.3 – Список актуальних команд

Далі протестуємо команду «Зберегти файл». Для цього необхідно вибрати цю команду зі списку команд. Натиснути на команду та зачекати на підказки чат-бота. Чат-бот опрацьовує команду та надсилає наступні підказки(рис. 3.4)

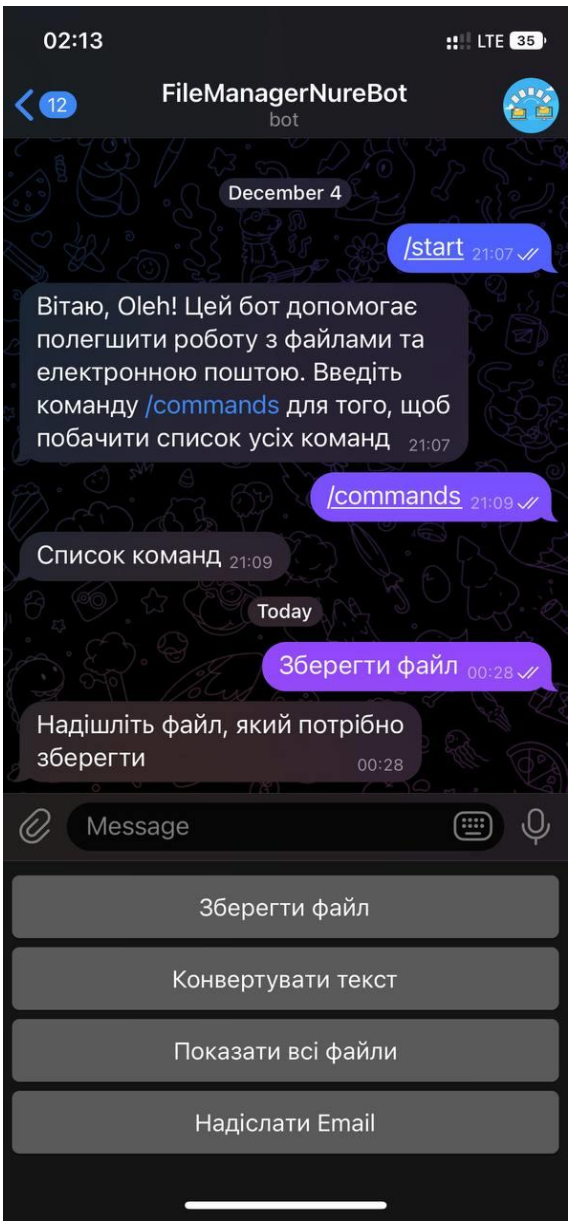


Рисунок 3.4 – Опрацювання команди «Зберегти файл»

Чат-бот надіслав підказки, що потрібно користувачу робити далі. Далі необхідно надіслати файл, який потрібно зберегти. Це може бути файл будь-якого формату. Надішлемо звичайний документ. Коли документ було успішно завантажено в сховище, чат-бот надішле повідомлення про це (рис. 3.5).

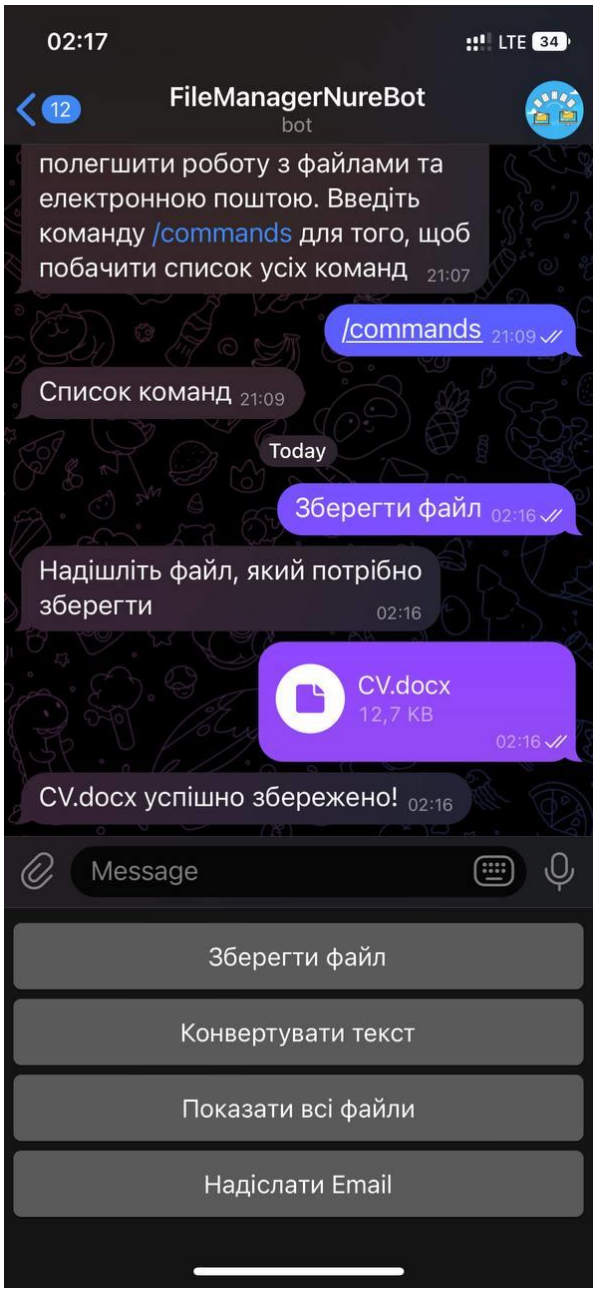


Рисунок 3.5 – Успішне збереження файлу до сховища

Далі протестуємо команду «Показати всі файли». Ця команда виводить список всіх файлів, які були збережені чат-ботом до хмарного сховища (рис. 3.6).

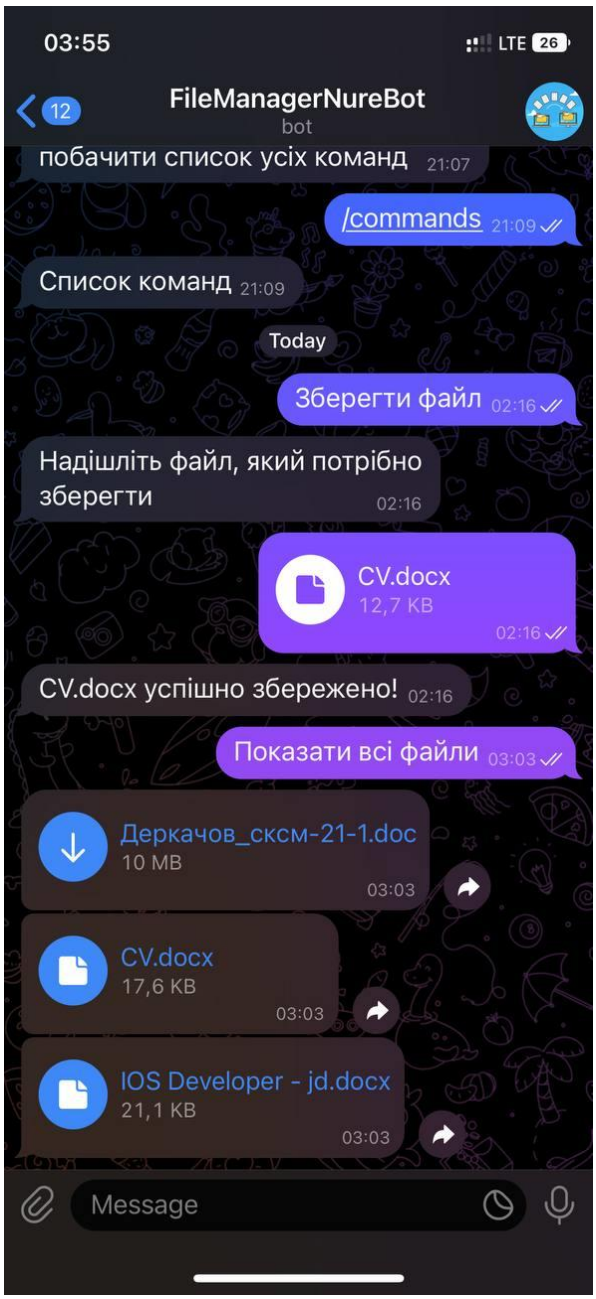


Рисунок 3.6 – Список всіх збережених файлів

Команда «Конвертувати текст» конвертує введений текст у файл та завантажує його до сховища хмарного сервісу. Після вибору команди чат-бот надсилає підказки, що потрібно робити далі (рис.3.7).

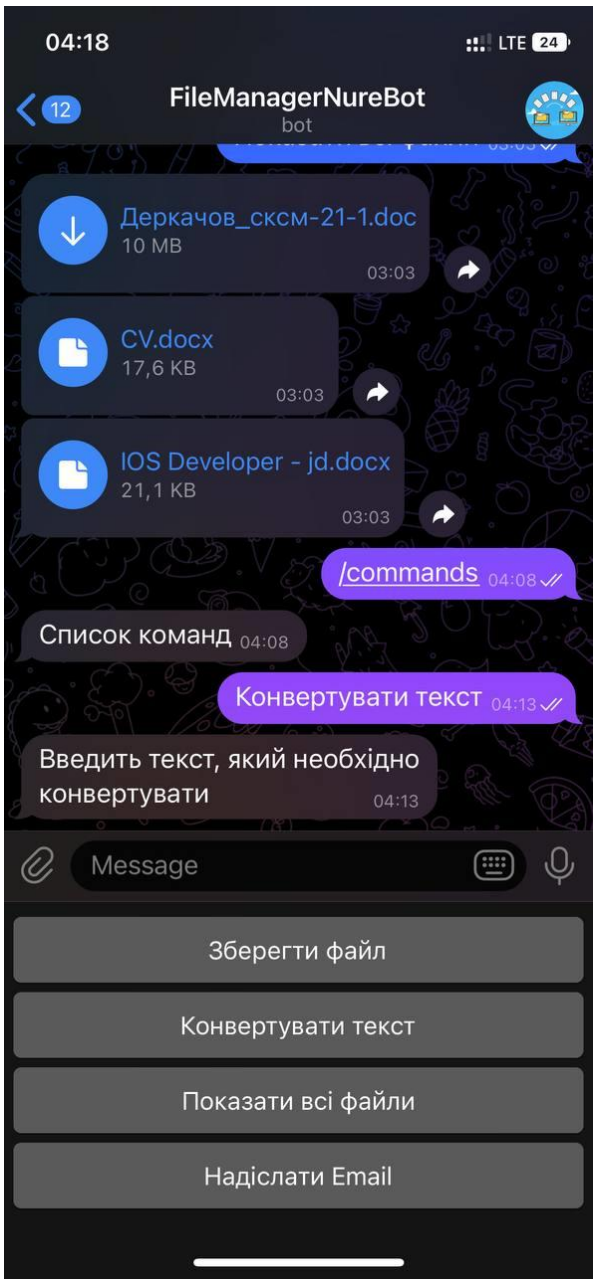


Рисунок 3.7 – Підказки для команди «Конвертувати текст»

Введемо текст та надішлемо його. Чат-бот опрацьовує текст, створює файл, зберігає його в сховище та надсилає команду про результат. В результаті написано також ім’я створеного файлу (рис. 3.8).

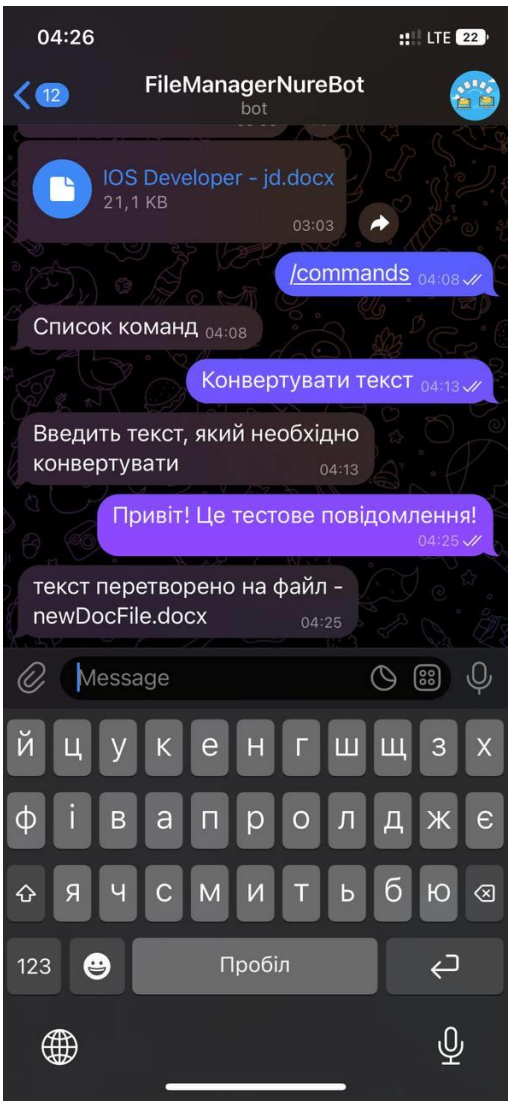


Рисунок 3.8 – Результат роботи команди

Протестуємо команду «Надіслати Email». Ця команда надсилає електронний лист від імені чат-боту на электрону адресу, яку введе користувач. Надішлемо команду чат-боту та отримаємо наступні підказки (рис. 3.9).

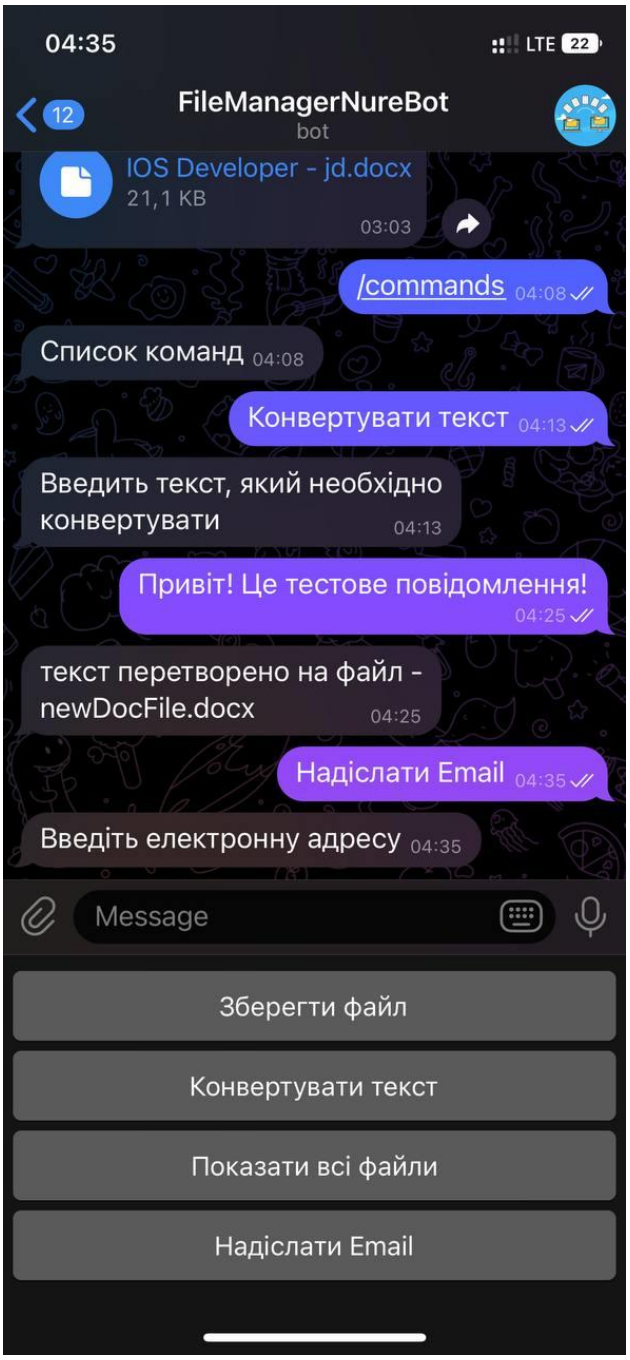


Рисунок 3.9 – Підказки для команди

Введемо електронну адресу, на яку потрібно надіслати електронний лист. Коли бот опрацює цю команду та отримаємо підказки від чат-бота (рис. 3.10).

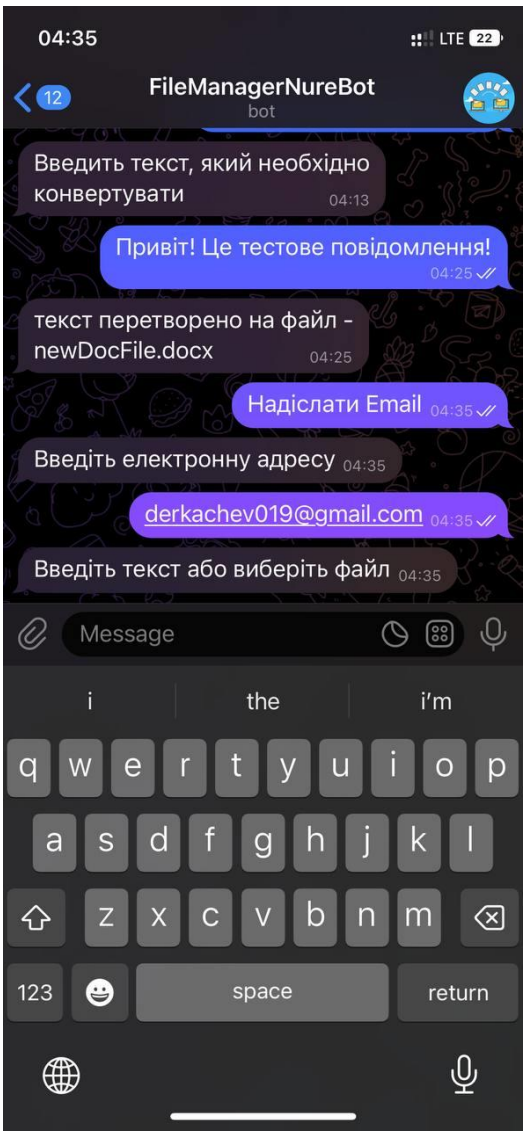


Рисунок 3.10 – Підказки чат-бота

Надішлемо тестове повідомлення з текстом «Привіт! Це тестовий Email!». Чат-бот опрацює це повідомлення та надішле його на електронну адресу, про що також повідомить в чаті (рис. 3.11).

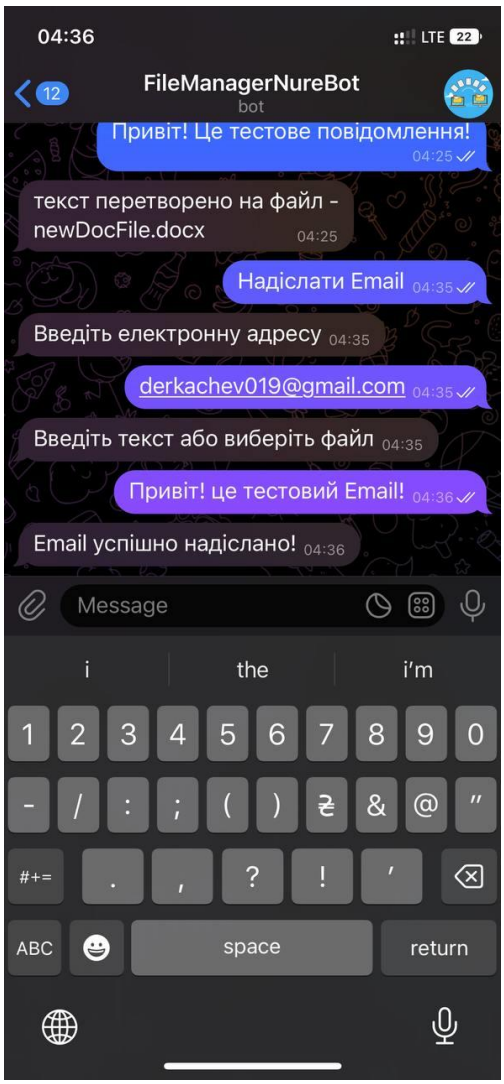


Рисунок 3.11 – Успішне виконання команди

Треба перевірити нашу електронну пошту, чи надійшов туди електронний лист від чат-боту (рис. 3.12).

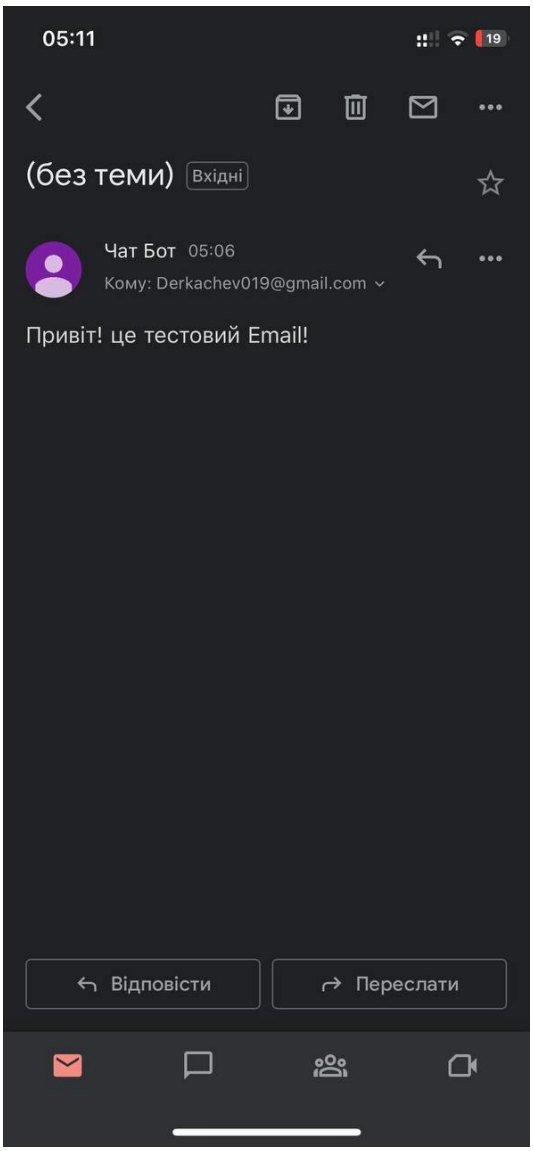


Рисунок 3.12 – Успішно надісланий електронний лист

Для подальшого тестування були протестовані ці ж самі команди але с різними вхідними даними.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи було спроектовано модель хмарного сервісу на основі чат-боту для файлової системи в рамках реалізації хмарних сервісів кіберуніверситету.

Як результат спроектовано чат-бота для створення, редагування там надсилання файлів електронною поштою. Розглянуто обрані інструменти програмування. Детально розглянуто структуру чат боту. Реалізовані команди хмарного сервісу з відповідним функціоналом. Хмарний сервіс надає можливість створювати файли, конвертувати текст у файли, а також надсилати файли електронною поштою.

Розроблене програмне забезпечення дозволяє пришвидшити процес навчання у студентів. Легке створення, редагування та надсилання файлів позитивно відобразиться на самопочутті користувача. Легкий менеджмент файлів та доступ з будь-якого девайсу зменшує тиск на студентів під час навчального року та підчас сесії та екзаменів.

Серед технологічних сервісів розумного кіберуніверситету, визначених у роботі, видокремлюється хмарний сервіс на основі чат-боту для поліпшення та прискрення якості освітніх процесів та компонентів. У межах даного підходу запропоноване у роботі рішення може бути імплементовано як компонент хмарного сервісу, що підтверджує його практичну значущість.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Хаханов В.И. Киберсоциальная система – умный кибер-университет / В.И. Хаханов, Е.И. Литвинова, С.В. Чумаченко, А.С. Мищенко // Радіоелектронні і комп'ютерні системи. – 2016. – № 5 (79). – С.187-194.

2. PyCharm Capabilities [Електронний ресурс] / Tech Target − Режим доступу: www / URL: https://www.jetbrains.com/ru-ru/pycharm/features/ – 05.05.2021. − Загол. з екрану.

3. Python 3.12.0a2 documentation [Електронний ресурс] / Tech Target − Режим доступу: www / URL: https://docs.python.org/dev/ – 01.05.2021. − Загол. з екрану.

4. Bots: An introduction for developers [Електронний ресурс] / Tech Target − Режим доступу: www / URL: https://core.telegram.org/bots#6-botfather – 01.10.2021. − Загол. з екрану.

5. What is cloud service based technology [Електронний ресурс] / Tech Target − Режим доступу: www / URL: https://www.techtarget.com/searchitchannel/definition/cloud-services – 05.09.2021. − Загол. з екрану.