Міністерство освіти і науки
України Національний
технічний університет
«Дніпровська політехніка»

_____Інститут електроенергетики_____
(навчально-науковий інститут)
_____факультет інформаційних технологій____
(факультет)
Кафедра_____інформаційних технологій та комп'ютерної інженерії_____
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційної роботи ступеня __бакалавра__

(бакалавра, магістра)

Здобувача вищої освіти_____ Похиленко Владислав Володимирович _____
(ПІБ)

академічної групи_____126-21-1_____
(шифр)

спеціальності _____126 «Інформаційні системи та технології»_____
(код і назва спеціальності)

~~спеціалізації~~ за освітньо-професійною (освітньо-науковою) програмою_____
(за наявності)

_____
(офіційна назва)

на тему__Розробка чат-бота на основі нейронних мереж для автоматизації_____ ____
клієнтської підтримки взаємодії студентів зі співробітниками деканату університету
(назва за наказом ректора)

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	проф. Коротенко Г.М.			
розділів:				

Рецензент	доц. Ширін А.Л.			

Нормоконтролер	проф. Коротенко Г.М.			

Дніпро
2025-

ЗАТВЕРДЖЕНО:
завідувач кафедри
_інформаційних технологій та комп'ютерної інженерії _
(повна назва)

_____ _В.В. Гнатушенко_
(підпис) (ініціали та прізвище)

«_____» _____2025 року

ЗАВДАННЯ

на кваліфікаційну роботу
ступеня__бакалавра_____
(бакалавра, магістра)

здобувача вищої освіти____Похиленко В.В._____**академічної групи**___126-21-1___
(прізвище та ініціали) (шифр)

спеціальності_____126 «Інформаційні системи та технології»_____

спеціалізації за освітньою-професійною програмою _____
(за наявності)

_____

на тему __Розробка чат-бота на основі нейронних мереж для автоматизації_____
_клієнтської підтримки взаємодії студентів зі співробітниками деканату університету____

затверджену наказом ректора НТУ «Дніпровська політехніка» від 05.05.2025 р № _336-с_

Розділ	Зміст	Термін виконання
Розділ 1	Огляд сучасних технологій реалізації чат-ботів	24.02.25 – 29.03.25
Розділ 2	Аналіз предметної області та постановка задачі	01.04.25 – 18.04.25
Розділ 3	Реалізація Telegram-бота	19.04.25 – 22.05.25
Розділ 4	Тестування та оцінка якості системи	23.05.25 – 12.06.25

Завдання видано _____ ____Г.М.Коротенко_____
_____(підпис керівника) (ініціали та прізвище)

Дата видачі __01.02.2025 р._____

Дата подання до екзаменаційної комісії___17.06.2025 р_____

Прийнято до виконання _____ _____В.В. Похиленко_____
(підпис здобувача вищої освіти) (ініціали та прізвище)

РЕФЕРАТ

Пояснювальна записка: 42 с., 10 рис., 7 табл., 12 джерел.

Ключові слова: ЧАТ-БОТ, TELEGRAM BOT API, PYTHON, RASA, NLP, ГЛИБОКЕ НАВЧАННЯ, НЕЙРОННІ МЕРЕЖІ, АВТОМАТИЗАЦІЯ, ПІДТРИМКА СТУДЕНТІВ, ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ.

Об'єкт кваліфікаційної роботи — створення інтелектуального чат-бота, який автоматизує відповіді на типові студентські запити до деканату факультету інформаційних технологій.

Предмет кваліфікаційної роботи— процес інформаційної взаємодії між студентами та адміністрацією ВНЗ.

Мета роботи —створення діалогових систем на основі нейронних мереж і Telegram API.

У роботі було проведено аналіз потреб студентів, сформульовано функціональні вимоги, обрано архітектуру чат-бота та реалізовано його функціонал із використанням бібліотек Python та Telegram Bot API. Реалізовано обробку повідомлень користувача, базу знань, діалогові сценарії, а також інтерфейс для отримання розкладу, контактної інформації та PDF-документів.

Розроблений бот є прикладом ефективного застосування сучасних технологій для автоматизації внутрішніх процесів в університетах, що дозволяє зменшити навантаження на працівників деканату та покращити якість обслуговування студентів.

ABSTRACT

Explanatory note: 42 pages, 10 figures, 7 table, 12 references.

Keywords: CHATBOT, TELEGRAM BOT API, PYTHON, RASA, NLP, DEEP LEARNING, NEURAL NETWORKS, AUTOMATION, STUDENT SUPPORT, INFORMATION TECHNOLOGY.

Object of the qualification work: is to develop an intelligent chatbot that automates responses to typical student inquiries addressed to the Faculty of Information Technology.

Subject of the qualification work: the process of information interaction between students and university administration.

Purpose of the work: creating dialogue systems based on neural networks and Telegram API.

The work includes an analysis of students' needs, formulation of functional requirements, selection of chatbot architecture, and implementation of its functionality using Python libraries and the Telegram Bot API. It implements message processing, a knowledge base, dialogue scenarios, and a user interface for accessing schedules, contact information, and PDF documents.

The developed chatbot serves as an example of the effective application of modern technologies for automating internal university processes, reducing the workload on faculty staff, and improving the quality of student service.

ЗМІСТ

УМОВНІ ПОЗНАЧЕННЯ І СКОРОЧЕННЯ

1. **AI** – Artificial Intelligence (штучний інтелект)

2. **NLP** – Natural Language Processing (обробка природної мови)

3. **ML** – Machine Learning (машинне навчання)

4. **UI** – User Interface (інтерфейс користувача)

5. **API** – Application Programming Interface (інтерфейс програмування додатків)

6. **JSON** – JavaScript Object Notation (формат обміну даними)

ВСТУП

У сучасних умовах цифрової трансформації освітнього процесу все більшого значення набувають інструменти, що дозволяють автоматизувати комунікацію між студентами та адміністративними підрозділами закладів вищої освіти. Одним із таких інструментів є чат-боти, які забезпечують швидкий доступ до довідкової інформації, знижують навантаження на персонал та покращують якість обслуговування.

Актуальність теми полягає у необхідності впровадження інтелектуальних рішень для автоматизації типових інформаційних запитів студентів, зокрема до деканату факультету. Такий підхід дозволяє оперативно надавати відповіді на найпоширеніші запитання, пов’язані з розкладом занять, графіками консультацій, розташуванням корпусів, контактними даними тощо.

Метою роботи є розробка та реалізація чат-бота для Telegram, що автоматизує інформаційну взаємодію між студентами факультету інформаційних технологій та адміністрацією.

Об’єкт дослідження — процес інформаційної взаємодії між студентами та адміністрацією закладу вищої освіти.

Предмет дослідження — методи та інструменти побудови діалогових систем із використанням Telegram Bot API, мови Python, а також бібліотек для обробки природної мови.

Завдання дослідження:

- провести аналіз потреб студентів у комунікації з деканатом;
- сформулювати функціональні та нефункціональні вимоги до чат-бота;
- обрати архітектурне рішення та необхідні технології для реалізації;

розробити структуру бази знань і діалогів;

- реалізувати Telegram-бота з основними функціями;
- провести тестування функціональності.

Структура роботи. Кваліфікаційна робота складається з чотирьох розділів, списку використаних джерел та додатків. У першому розділі подано огляд сучасних технологій створення чат-ботів. У другому — проведено аналіз предметної області та сформульовано вимоги. У третьому розглянуто реалізацію чат-бота, описано його функціональні модулі. Четвертий розділ присвячено тестуванню і висновкам.

РОЗДІЛ 1. СУЧАСНІ ТЕХНОЛОГІЇ РЕАЛІЗАЦІЇ ЧАТ-БОТІВ НА БАЗІ НЕЙРОННИХ МЕРЕЖ

1.1. Концепція чат-ботів у сучасних інформаційних системах

У процесі розвитку цифрових технологій однією з найбільш динамічних форм взаємодії між користувачем і комп'ютерною системою стали чат-боти. Під цим поняттям розуміють програмне забезпечення, що здатне імітувати спілкування людини, використовуючи текстовий або голосовий формат.

Основна функція чат-бота полягає в обробці звернень користувачів із метою надання відповіді, рекомендацій або виконання дій у межах визначеного функціоналу. Такі системи реалізуються як самостійні додатки або вбудовуються у платформи комунікації, наприклад, месенджери, вебпортали або мобільні застосунки.

Застосування чат-ботів особливо поширене у сферах, де присутній високий обсяг повторюваних інформаційних запитів. У контексті вищої освіти ці технології здатні виконувати роль електронного консультанта, знижуючи навантаження на адміністративний персонал та забезпечуючи оперативне інформування студентів.

У сучасних реалізаціях чат-боти часто використовують алгоритми обробки природної мови (NLP), що дозволяє їм не лише реагувати на прості ключові слова, а й аналізувати структуру речення, виявляти смислові зв'язки та адаптувати відповіді до контексту. Це досягається за рахунок використання моделей машинного навчання, зокрема нейронних мереж.

Таким чином, чат-бот у структурі інформаційної системи виступає як гнучкий інтерфейс, що дозволяє автоматизувати комунікацію між системою і користувачем, забезпечуючи безперервну та масштабовану підтримку.

1.2. Типологія чат-ботів за принципом реалізації та сферою застосування

Чат-боти, як інструменти автоматизованої взаємодії, поділяються на кілька основних типів залежно від підходу до побудови логіки та завдань, які

вони виконують. Така класифікація допомагає краще розуміти можливості різних реалізацій у межах конкретної сфери застосування.

Першу категорію складають боти, що працюють за заздалегідь визначеними правилами. У таких системах реакція на запити користувача відбувається за принципом "ключове слово — відповідь", без глибокого аналізу змісту звернення. Подібні реалізації застосовуються переважно у випадках, де кількість можливих сценаріїв обмежена, наприклад, у довідкових службах з фіксованими відповідями.

Наступний тип — інтелектуальні чат-боти, які використовують алгоритми штучного інтелекту. Вони здатні аналізувати контекст запиту, граматику й навіть минулі повідомлення користувача. Робота таких систем ґрунтується на машинному навчанні, зокрема на нейронних мережах, що дозволяє їм адаптуватися до нових ситуацій без явного програмування кожного сценарію.

З технічної точки зору, розрізняють також текстові та голосові чат-боти. Перші реалізуються через інтерфейси вебплатформ, месенджери чи мобільні застосунки. Голосові боти інтегруються в системи розпізнавання мовлення і здатні вести діалог через мікрофон і синтез мови. Існують також комбіновані рішення, які поєднують обидва формати.

У сфері освіти, зокрема у вищих навчальних закладах, доцільно впроваджувати чат-боти, які орієнтовані на надання інформаційних послуг. Це, зокрема, відповіді на запити щодо навчального розкладу, організації освітнього процесу, документів та інших адміністративних питань. Саме такий функціонал дозволяє зменшити навантаження на працівників деканату та підвищити ефективність взаємодії зі студентами.

1.3. Історія розвитку та еволюція чат-ботів

Ідеї щодо створення програм, здатних до імітації людського спілкування, з'явились ще в середині XX століття. Одним із перших проєктів, який реалізував подібний підхід, була програма ELIZA, розроблена в Массачусетському технологічному інституті. Вона використовувала просту

підстановку фраз для створення ілюзії діалогу з користувачем і стала поштовхом до розвитку цілої галузі комп'ютерної лінгвістики.

Наступними значущими кроками стали розробки на кшталт PARRY, який мав більш складну логіку відповіді, хоча й досі базувався на правилах. У 1990-х роках, разом із поширенням інтернету, з'явилися IRC-боти та перші інтерактивні агенти, здатні підтримувати діалог у чатах.

На початку 2000-х років був створений ALICE — один із найпопулярніших чат-ботів того періоду, який тричі отримував нагороду Loebner Prize. Він базувався на використанні AIML (Artificial Intelligence Markup Language), що дозволяло розробникам створювати складні, але все ще статичні діалоги.

Справжній прорив у галузі відбувся в 2010-х роках із появою голосових асистентів, таких як Siri, Google Assistant та Cortana. Ці системи використовували хмарні технології, машинне навчання та розпізнавання мовлення, що дозволяло їм швидше адаптуватися до нових запитів.

Після 2018 року розвиток нейронних мереж на базі трансформерних архітектур (зокрема моделей BERT, GPT, T5) докорінно змінив можливості чат-ботів. Сучасні системи вже не обмежуються лише шаблонними діалогами — вони здатні генерувати відповіді, аналізувати наміри користувача, зберігати контекст розмови та забезпечувати більш "людяне" спілкування.

Сьогодні інтелектуальні боти активно впроваджуються в освіту, медицину, електронну комерцію, державні послуги. Їх застосування дає змогу автоматизувати рутинні процеси, підвищити доступність сервісів та створити новий рівень комунікації між людьми і цифровими системами.

1.4. Основи нейронних мереж у задачах обробки природної мови

Нейронні мережі є основою сучасних інтелектуальних систем, що працюють з текстовими даними. Вони були розроблені за аналогією з біологічними нейронами й дозволяють машині самостійно виявляти закономірності у даних, зокрема в мовленні. Завдяки цим властивостям

нейромережі широко використовуються в обробці природної мови (Natural Language Processing, NLP).

Одним із перших типів нейронних архітектур, які застосовувалися у мовних завданнях, стали **багатошарові персептрони (MLP)**. Проте їх ефективність була обмеженою через складність роботи з послідовностями тексту. Цю проблему частково вирішили **рекурентні нейронні мережі (RNN)**, здатні обробляти вхідні дані послідовно, з урахуванням попереднього контексту. На основі RNN були створені **LSTM (Long Short-Term Memory)** і **GRU (Gated Recurrent Unit)**, що краще зберігають інформацію на довгих відстанях у тексті.

У мовних задачах особливо важливою є здатність моделі не лише аналізувати окремі слова, а й виявляти **контекстні зв'язки між ними**. Саме тому з 2017 року почала активно розвиватися нова архітектура — **трансформери (Transformers)**. Вона працює за принципом **self-attention**, що дозволяє моделі розглядати всі елементи вхідної послідовності одночасно та визначати вагу кожного з них у загальному значенні.

На базі трансформерної архітектури побудовані такі моделі, як **BERT (Bidirectional Encoder Representations from Transformers)** і **GPT (Generative Pre-trained Transformer)**. Вони дозволили суттєво підвищити точність у завданнях класифікації текстів, генерації відповідей, машинного перекладу тощо.

Ще однією перевагою нейронних мереж є їх здатність до **навчання на великих обсягах неструктурованих даних**. Це означає, що моделі можна попередньо навчити на відкритих корпусах текстів, а потім донавчити для вузькоспеціалізованих задач, наприклад — обробки запитів студентів у системі ВНЗ.

Таким чином, саме завдяки використанню глибоких нейронних архітектур стало можливим створення гнучких чат-ботів, які не тільки відповідають за шаблоном, а й «розуміють» текст у контексті, адаптуючись до різних формулювань та стилів спілкування.

1.5. Архітектури трансформерів для обробки тексту: BERT, GPT та інші

Трансформерні моделі стали ключовою технологією в сучасних системах обробки природної мови. Запропонована у 2017 році архітектура **Transformer**, описана в дослідженні "Attention is All You Need", замінила рекурентні та згорткові підходи, які до того часу домінували у NLP. Її основна перевага — здатність ефективно працювати з контекстом, незалежно від довжини вхідного тексту.

Центральним компонентом трансформера є **механізм самоуваги (self-attention)**, який дозволяє моделі порівнювати кожне слово у реченні з усіма іншими словами і визначати, яке з них є більш релевантним у конкретному контексті. Завдяки цьому модель не втрачає важливу інформацію, як це траплялось у попередніх підходах.

Однією з найвідоміших трансформерних моделей є **BERT** (Bidirectional Encoder Representations from Transformers), розроблена Google. Вона навчається розуміти текст у двох напрямках одночасно — зліва направо і справа наліво, що дає змогу краще вловлювати сенс. BERT особливо добре працює у завданнях класифікації, пошуку інформації, відповідей на питання.

Інший приклад — **GPT (Generative Pre-trained Transformer)**, створений OpenAI. Модель GPT відрізняється тим, що є **універсальним генератором тексту**, здатним створювати зв’язні речення, вести діалог і навіть писати тексти у стилі, наближеному до людського. GPT працює у прямому напрямку (left-to-right), що підходить для задач генерації, продовження тексту, автоматизованого письма.

Серед інших трансформерних моделей, що мають велике значення, варто згадати:

1. RoBERTa — модифікація BERT з кращими результатами за рахунок оптимізації навчання;
2. T5 (Text-to-Text Transfer Transformer) — універсальна модель, яка перетворює всі завдання у формат "вхідний текст → вихідний текст";

3. XLNet — комбінує переваги автогерeративних і автокодуючих підходів.

Для розробки сучасних чат-ботів, особливо тих, які повинні аналізувати запити студентів у системі ВНЗ, трансформерні архітектури забезпечують високу якість відповідей, адаптивність до формулювань і гнучкість у використанні.

1.6. Огляд бібліотек і технологій для розробки чат-ботів

Розробка чат-ботів у сучасних умовах тісно пов’язана з використанням спеціалізованих програмних бібліотек, фреймворків і платформ. Завдяки цим інструментам створення навіть складних діалогових систем стало доступним розробникам із базовими знаннями у сфері програмування.

Однією з найпопулярніших бібліотек є **Rasa** — open-source фреймворк на мові Python, який дає змогу створювати боти із використанням технологій машинного навчання. Перевага Rasa полягає в гнучкості — бот може працювати як на локальному сервері, так і в хмарному середовищі, що є важливим для навчальних установ з обмеженим доступом до зовнішніх платформ.

Іншим зручним інструментом є **Dialogflow**, що належить Google. Це платформа з графічним інтерфейсом, яка дає можливість швидко створити бота без глибокого програмування. Вона інтегрується з Google Assistant, Telegram, Slack, Facebook Messenger, що робить її універсальною для широкого кола задач.

Для тих, хто планує реалізувати чат-бот із використанням моделей GPT або інших трансформерів, доцільно використовувати бібліотеки на кшталт **Hugging Face Transformers**. Цей пакет дозволяє інтегрувати готові попередньо навчені моделі, оптимізовані для задач обробки природної мови, зокрема генерації відповідей, класифікації тексту, аналізу тональності.

У контексті розробки ботів для освітніх цілей важливою є можливість **підключення до внутрішніх баз даних** та **інтеграції з навчальними**

платформами. Для цього використовуються REST API, Webhooks, а також фреймворки типу **Flask** або **FastAPI** для створення бекенду.

Також варто звернути увагу на **Telegram Bot API**, який надає простий і гнучкий спосіб створення ботів, зокрема для взаємодії студентів із адміністративними структурами. За допомогою бібліотеки **python-telegram-bot** можна реалізувати логіку діалогу, меню, авторизацію та збирання статистики. Таким чином, вибір бібліотеки чи платформи залежить від рівня складності проекту, технічних вимог і ресурсів, доступних для реалізації. У межах автоматизації комунікації між студентами і деканатом доцільно комбінувати готові рішення із можливістю подальшої модифікації, що забезпечує гнучкість і масштабованість системи.

У таблиці 1.1 узагальнено основні характеристики інструментів, що застосовуються для створення чат-ботів у межах дослідження.

Таблиця 1.1 – Порівняльна характеристика мовних платформ для створення чат-ботів

Платформа / Бібліотека	**Сильні сторони**	**Слабкі сторони**
Flask / FastAPI	Гнучкість для побудови бекенду, підтримка REST API, інтеграція з ботами	Потребує самостійної розробки всієї логіки
Rasa	Висока гнучкість, підтримка NLU, можливість локального розгортання	Потребує глибших технічних знань для налаштування та тренування
Dialogflow	Простота у використанні, графічний інтерфейс, інтеграція з Google-сервісами	Обмежена кастомізація, залежність від Google Cloud
Hugging Face Transformers	Доступ до сучасних моделей NLP (GPT, BERT), відкритий код	Потребує значних обчислювальних ресурсів і знань у сфері NLP
Telegram Bot API	Простота реалізації, популярність серед студентів, готові Python-бібліотеки	Обмежена функціональність без бекенд-інтеграції

Висновки до розділу 1

У цьому розділі було розглянуто ключові поняття та технологічні засади, що лежать в основі побудови сучасних чат-ботів, зокрема тих, які використовуються для автоматизації інформаційного обслуговування. Проаналізовано типи чат-ботів за принципами реалізації, способами взаємодії з користувачем та сферами застосування.

Особливу увагу приділено нейронним мережам і трансформерним архітектурам, які забезпечують гнучкість, адаптивність та високу якість відповідей у системах обробки природної мови. Розглянуті моделі, такі як BERT і GPT, демонструють високий потенціал для впровадження в освітньому середовищі, зокрема в контексті підтримки взаємодії між студентами та адміністрацією закладу.

Також наведено огляд інструментів і бібліотек, що дозволяють реалізувати чат-бот із використанням сучасних мовних моделей. Ці відомості створюють теоретичну основу для подальшого аналізу предметної області, постановки задачі та реалізації програмного продукту в межах даного дослідження.

РОЗДІЛ 2. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

2.1 Аналіз поточного стану комунікації студентів з деканатом

У сучасних умовах ефективна комунікація між студентами та адміністративними підрозділами закладів вищої освіти є важливою складовою організації навчального процесу. Одним з ключових елементів такої взаємодії є деканат — структурний підрозділ, який координує навчальні процеси, веде документацію та здійснює консультативну підтримку студентів.

Факультет “ Інформаційні системи та технології ” у НТУ «Дніпровська політехніка» обслуговує значну кількість студентів денної форми навчання. Згідно з поточними організаційними підходами, основні канали комунікації між студентами та співробітниками деканату включають особисті звернення, телефонні дзвінки та електронну пошту. В окремих випадках також використовуються неофіційні месенджери, зокрема Telegram, але без автоматизованої обробки звернень.

Аналіз поточного стану дозволяє окреслити низку викликів, що супроводжують процес взаємодії студентів з деканатом:

1. Обмежена швидкість обробки звернень у періоди підвищеного навантаження.
2. Інформація, необхідна студентам (розклад занять, графік консультацій, розташування корпусів тощо), хоч і представлена на офіційному сайті, однак її пошук може ускладнюватися через складну навігацію або фрагментарність розміщення.
3. Часте повторення однотипних запитів, що зумовлює підвищене навантаження на працівників деканату.

У зв’язку з цим доцільним видається впровадження сучасного інструменту — чат-бота, який би забезпечив оперативну та зручну взаємодію, водночас зменшивши кількість рутинних звернень до адміністрації факультету.

2.2 Вимоги до майбутнього рішення

На основі аналізу існуючих умов взаємодії студентів з деканатом факультету сформульовано ключові вимоги до функціонування майбутньої автоматизованої системи — чат-бота.

Ці вимоги поділяються на функціональні (що саме повинен робити бот) та нефункціональні (вимоги до якості роботи, інтерфейсу, безпеки тощо) Вимоги згруповані у таблиці 2.1.

Таблиця 2.1 – Класифікація вимог до чат-бота для студентської підтримки

Тип вимог	Формулювання вимоги
Функціональні	Надання довідкової інформації (розклад занять, графік консультацій, контакти тощо)
	Автоматизоване реагування на типові запити
	Можливість залишити звернення до деканату через бота
	Підтримка меню з категоріями запитів для швидкого доступу до інформації
Нефункціональні	Цілодобова доступність у месенджері Telegram
	Інтуїтивно зрозумілий інтерфейс, простота навігації
	Можливість масштабування функціоналу в майбутньому
	Підтримка української мови
	Забезпечення базового рівня захисту персональних даних користувачів

2.3 Вибір архітектури чат-бота

Одним з ключових етапів проєктування чат-бота є вибір його архітектури. Від правильної побудови внутрішньої структури залежить ефективність обробки запитів, стабільність роботи системи та можливість її масштабування в майбутньому.

Існує кілька типових архітектур чат-ботів, які відрізняються рівнем складності, способом обробки вхідної інформації та сценаріями взаємодії з користувачем:

1. **Rule-based (на правилах)** – базується на заздалегідь заданих шаблонах запит–відповідь. Відповіді формуються на основі ключових слів або регулярних виразів.

2. **AI-based (на основі NLP/ML)** – використовує технології обробки природної мови (Natural Language Processing), що дозволяє аналізувати семантику запиту, контекст і надавати гнучкіші відповіді.

3. **Гібридна архітектура** – поєднує rule-based логіку для простих запитів та NLP-компоненти для складніших випадків.

З огляду на специфіку проєкту (обмежена кількість типових звернень, переважно інформаційного характеру), а також з урахуванням обмежених ресурсів, **було обрано гібридну архітектуру** з переважанням rule-based складової та можливістю розширення NLP-функціоналу в майбутньому.Для реалізації проєкту запропоновано архітектуру, яка включає такі ключові компоненти (табл. 2.2)

Таблиця 2.2 – Основні компоненти архітектури чат-бота

Компонент	Опис
Інтерфейс взаємодії	Telegram Bot API – приймає повідомлення користувача та передає їх далі.
Модуль обробки запитів	Dispatcher – маршрутизує запити до відповідних обробників.
База знань / сценарії	Зберігає шаблони відповідей, структуру діалогів, варіанти запитань.
NLP-модуль (опційно)	Hugging Face, Rasa NLU – обробляє запити у вільній формі.
Сховище логів	База даних – зберігає історію звернень для подальшого аналізу

2.4 Побудова бази знань

Функціонування чат-бота неможливе без наявності структурованої бази знань, яка забезпечує відповідь на запити користувачів. У контексті проєкту, спрямованого на підтримку студентів факультету, база знань має включати довідкову, навчальну та адміністративну інформацію.

Джерела формування бази знань:

1. Офіційний сайт факультету та університету (розклад занять, графік сесій, консультацій, контакти, оголошення).
2. Типові запитання, що найчастіше надходять до деканату (за попереднім аналізом).
3. Внутрішні документи, що регулюють навчальний процес (наприклад, правила пропуску пар, графік видачі довідок тощо).

Інформація організовується у вигляді:

1. Пари запит–відповідь (Q&A): короткі блоки з ключовими словами.
2. Діалогових сценаріїв: багатокрокові структури із варіантами вибору (наприклад: «Оберіть факультет» → «Оберіть курс» → «Показати розклад»).
3. Категорій: групування інформації за темами (розклад, документи, контакти, адміністрація).

Технічна реалізація:

1. Дані можуть зберігатися у вигляді **JSON-структур**, **CSV-файлів**, або **SQLite-бази**, що легко імпортується в логіку бота.
2. Для відповіді на звернення використовуються **ключові слова** та прості алгоритми пошуку.
3. Передбачена можливість **редагування бази знань вручну** через окремий конфігураційний файл.

У разі розширення функціоналу з використанням NLP-платформ (таких як Rasa), база знань може включати **інтенти, сутності та шаблони відповідей**, які тренуються за допомогою навчального корпусу.

2.5 Вибір інструментів

Успішна реалізація чат-бота значною мірою залежить від правильного вибору технологій, які забезпечують ефективність, зручність розробки та підтримку необхідного функціоналу. Враховуючи обмеженість ресурсів, вимоги до простоти підтримки, підтримку української мови та інтеграцію з популярними платформами, було обрано низку інструментів, які є оптимальними для даного проєкту.

Ключовими критеріями вибору стали:

1. відкритість і безкоштовність рішень;
2. активна спільнота та наявність документації;
3. простота розгортання та підтримки;
4. можливість роботи з текстовими повідомленнями та обробки природної мови.

Основні інструменти, які використовуються у розробці чат-бота, наведено у таблиці 2.3.

Таблиця 2.3 – Обрані інструменти розробки чат-бота

Категорія	**Інструмент / Технологія**	**Обґрунтування вибору**
Мова програмування	Python	Простий синтаксис, багата екосистема, підтримка бібліотек для NLP та Telegram API
Месенджер / API	Telegram Bot API	Відкритий API, підтримка кнопок, інтерактивного меню, не потребує складного хостингу
Бібліотека для бота	python-telegram-bot	Зручна для організації обробників повідомлень, добре документована
Бібліотека NLP	spaCy, Rasa	Для розпізнавання намірів, контексту запитів і побудови розмовної

		логіки
Формат зберігання	JSON, SQLite	Прості у використанні, підходять для збереження сценаріїв, логів, шаблонів відповідей

2.6 Реалізація інтерфейсу користувача

Інтерфейс користувача чат-бота — це критично важливий елемент, що визначає зручність використання системи. З огляду на те, що цільова аудиторія — студенти, які звикли до мобільних месенджерів, інтерфейс був реалізований у Telegram, що забезпечує простоту доступу та інтуїтивну взаємодію.

Основні особливості інтерфейсу:

- **Меню з варіантами вибору**: бот пропонує набір кнопок або команд, що дозволяють швидко обрати потрібний розділ (наприклад: · Розклад занять, · Розташування корпусів, · Контакти деканату).
- **Підказки та інструкції**: у кожному розділі бот виводить короткі інструкції для користувача, пояснюючи подальші дії.
- **Обробка некоректних запитів**: якщо бот не розуміє повідомлення, він відповідає фразою на кшталт: *«На жаль, я не можу опрацювати це повідомлення. Спробуйте скористатися меню.»*
- **Підтримка кнопок у форматі inline**: це дозволяє створити зручну навігацію без необхідності вводити текст вручну.
- **Зовнішній вигляд повідомлень**: відповіді бота мають логічну структуру, використовують емодзі та розділення інформації по блоках.

Структура головного меню (Приклад візуального представлення головного меню наведено у таблиці 2.4)

Таблиця 2.4 – Візуальна структура головного меню чат-бота

· Розклад занять
· Де знаходиться корпус?
· Отримати довідку
i Часті питання
· Контакти
· Написати звернення

2.7 Структура діалогів і сценарії обробки запитів

Оскільки чат-бот має обслуговувати типові звернення студентів, важливо продумати логіку побудови діалогів та сценаріїв взаємодії. Це дозволяє зробити процес комунікації з ботом зрозумілим, швидким і максимально автоматизованим.

Типи сценаріїв обробки запитів:

1. **Прості сценарії (одноетапні)**
2. Наприклад:

   ? «*Коли починається перерва?*» → ✓«*Пара №2 закінчується об 11:35. Перерва – 10 хв.*»

3. **Сценарії з вибором (меню)**
4. Запит: «*Хочу подивитись розклад*» Бот:

   ➔ · Обери курс

   ➔ ·· Обери групу

   ➔ · Виводиться розклад

5. **Умовно-динамічні сценарії**

6. Наприклад, якщо користувач вводить «*Чи можна відновити стипендію?*» – бот визначає ключові слова та перенаправляє до FAQ або надає відповідь із посиланням на регламент.

Приклади структур діалогів (таблиця 2.5):

Таблиця 2.5 – Приклади сценаріїв обробки типових запитів користувача

Запит користувача	**Дії бота**
«Коли починається друга пара?»	Виводить час початку та кінця другої пари
«Розклад»	Просить вибрати курс → групу → видає розклад
«Як дістатись до корпусу №4?»	Надсилає адресу, опис маршруту та посилання на Google Maps
«Як подати звернення?»	Відповідає з шаблоном: «Вкажіть ПІБ, суть звернення та контакт»
Незрозумілий текст	Відповідає: «На жаль, не зрозумів запит. Спробуйте скористатися меню»

2.8 Висновки до розділу 2

У другому розділі було здійснено комплексне проєктування інформаційної системи — чат-бота для автоматизації взаємодії студентів факультету із деканатом. Визначено вимоги до функціональності, структури бази знань, інтерфейсу користувача, а також підібрано оптимальні технологічні засоби реалізації.

Проведено аналіз ключових потреб студентів, серед яких — оперативність отримання відповідей, зручність доступу до розкладу, інформації про розташування корпусів, а також можливість звернення до адміністрації в онлайн-режимі. З огляду на ці потреби сформовано перелік функціональних вимог до чат-бота та побудовано відповідну структуру його логіки.

Було обґрунтовано вибір **гібридної архітектури** чат-бота з переважанням rule-based підходу. Така архітектура дозволяє ефективно обробляти типові запити, при цьому залишаючи можливість подальшої інтеграції NLP-технологій для обробки складніших звернень.

Окрему увагу приділено **побудові бази знань** — основи, на якій базується функціонування бота. Розроблено структуру з типових запитів, відповідей, категорій інформації та сценаріїв діалогів. Базу знань передбачено в реалізації у форматі JSON або простих реляційних структур на основі SQLite, що забезпечує простоту редагування та підтримки.

Було детально описано інтерфейс користувача — Telegram-бот із зручним меню, інструкціями та підтримкою клавіш швидкого доступу. Запропоновано приклад головного меню, а також наведено таблиці діалогів і реакцій бота на типові звернення.

Також обґрунтовано вибір **технологій реалізації**: Python, Telegram Bot API, бібліотек **python-telegram-bot**, а також (опційно) інструментів для NLP-модулів. Ці технології є відкритими, популярними серед розробників, добре документованими та легко масштабованими.

Таким чином, результати другого розділу формують повноцінну проєктну основу для реалізації інформаційної системи. У наступному розділі буде здійснено безпосередню **реалізацію запропонованих рішень**, тестування функціоналу чат-бота та оцінка ефективності його роботи.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ TELEGRAM-БОТА

3.1 Встановлення середовища розробки

Для реалізації Telegram-бота було обрано мову програмування Python версії 3.10 та середовище розробки IDLE, що входить у стандартну дистрибуцію. Встановлення Python здійснюється з офіційного сайту https://www.python.org, після чого можна розпочати написання коду без потреби в додаткових IDE. Результат перевірки встановлення Python зображено на **рисунку 3.1**

Для коректної роботи Telegram-бота було встановлено бібліотеку pyTelegramBotAPI (альтернатива: telebot), яка дозволяє легко створювати обробники подій і надсилати повідомлення користувачам. Результат встановлення бібліотеки зображено на **рисунку 3.2**

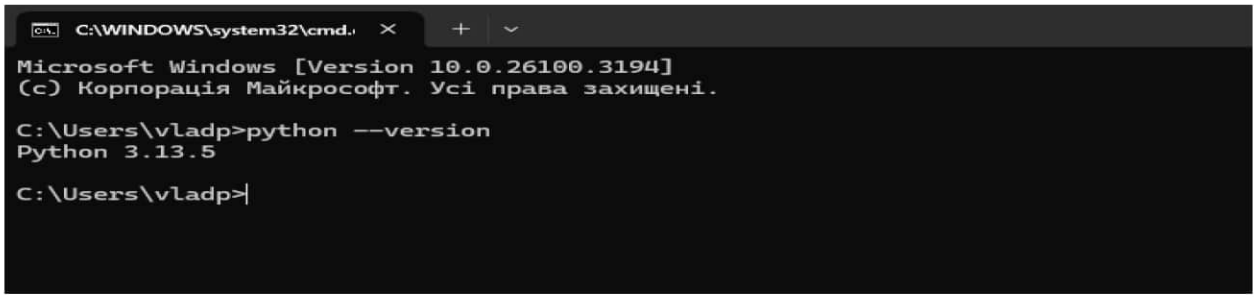

Рисунок 3.1 – Перевірка встановлення Python у командному рядку Windows

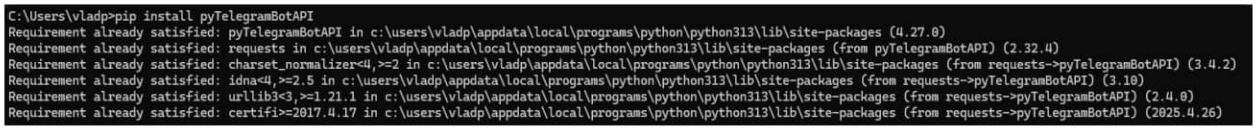

Рисунок 3.2 – Встановлення бібліотеки pyTelegramBotAPI через командний рядок

3.2 Реєстрація та налаштування Telegram-бота

Для створення Telegram-бота використано офіційний сервіс **@BotFather**. У процесі реєстрації було виконано наступні кроки:

1. Відправлено команду **/start**
2. Вибрано опцію **Create a new bot**
3. Задано ім’я бота та його username

4. Отримано унікальний токен для доступу до API

Результат створення Telegram-бота та отримання токена API зображено на **рисунку 3.3**

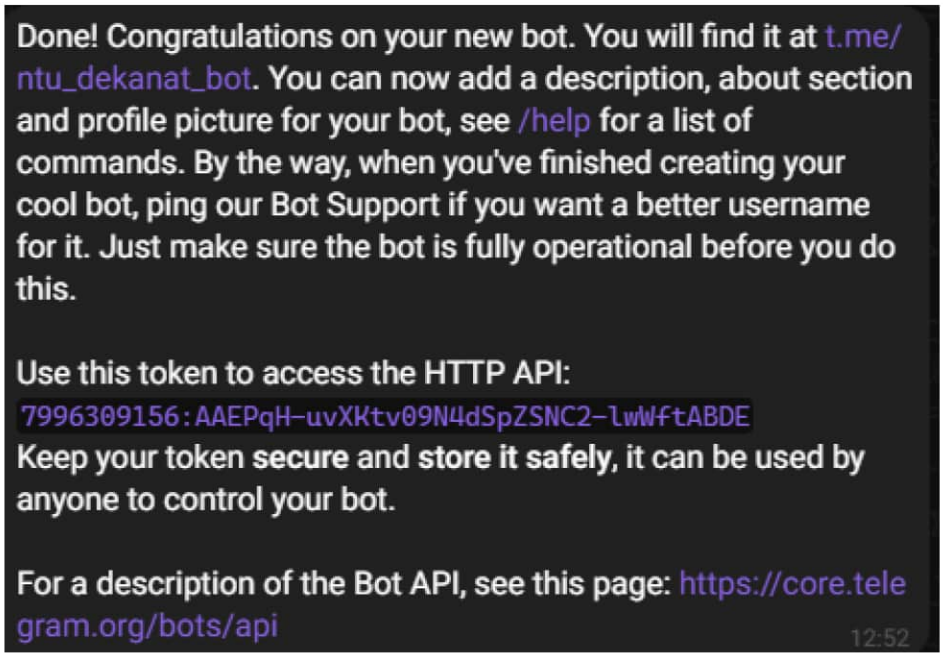

Рисунок 3.3 – Створення Telegram-бота за допомогою сервісу BotFather

3.3 Обробка типових запитів

Розроблений Telegram-бот обробляє найпоширеніші звернення студентів факультету інформаційних технологій. Кожна команда реалізується через окремий обробник, що виконує відповідну дію на основі натиснутої кнопки.

1.Обробка команди «**/start**»

Команда **/start** ініціалізує роботу чат-бота. Після її введення користувач отримує привітальне повідомлення та головне меню з кнопками для зручного вибору типових запитів.

Відповідний фрагмент коду:

```
@bot.message_handler(commands=['start'])
def send_welcome(message):
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
    btn1 = types.KeyboardButton("· Розклад занять")
    btn2 = types.KeyboardButton("· Де знаходиться корпус?")
    btn3 = types.KeyboardButton("· Отримати довідку")
```

```
    btn4 = types.KeyboardButton("ℹ Часті питання")
    btn5 = types.KeyboardButton("· Контакти")
    btn6 = types.KeyboardButton("· Написати звернення")
    markup.add(btn1, btn2)
    markup.add(btn3, btn4)
    markup.add(btn5, btn6)
    bot.send_message(
        message.chat.id,
        "Вітаю! Це бот факультету ФІТ. Оберіть потрібний розділ ↓",
        reply_markup=markup
    )
```

с

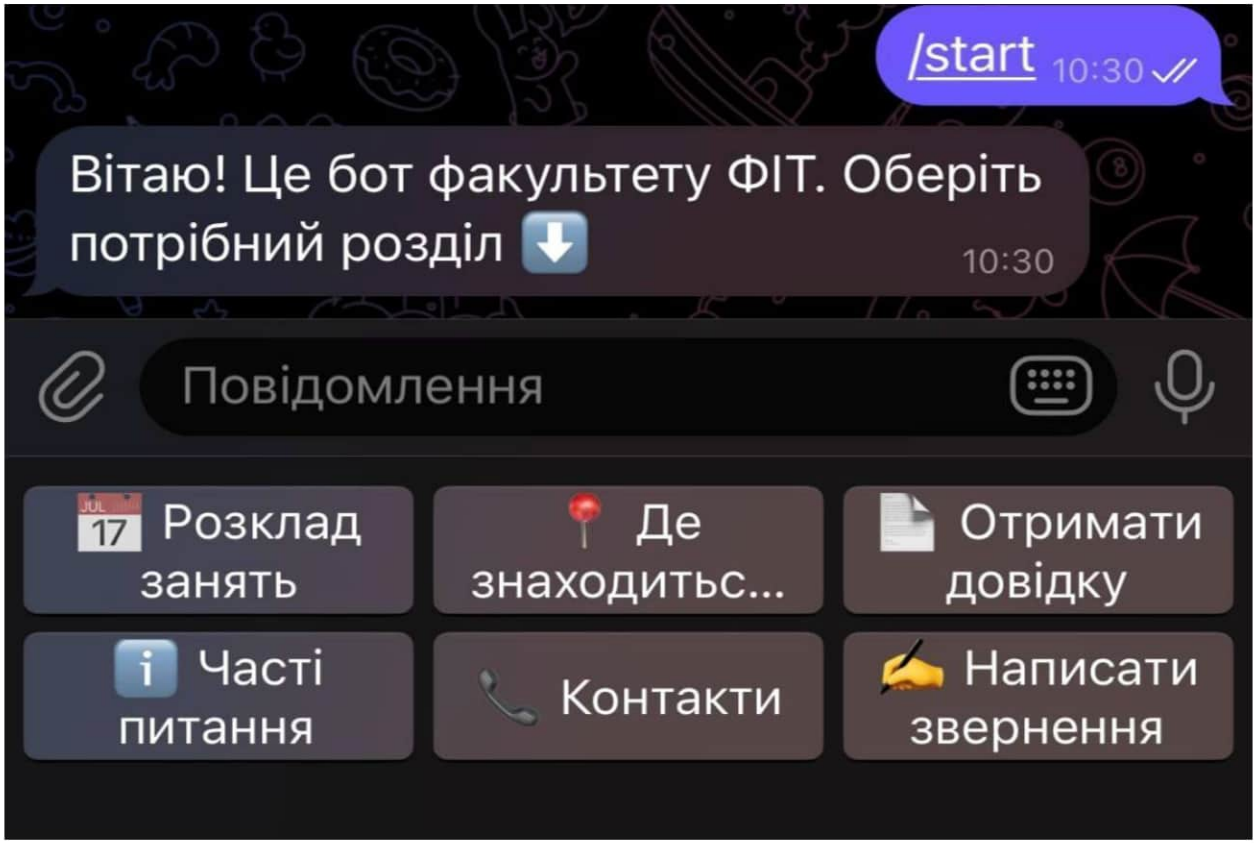

Рисунок 3.4 – Відображення головного меню після введення команди /start

2.Обробка команди «**/help**»

Команда **/help** надає студенту текстову інструкцію про доступні функції чат-бота. Це текстова альтернатива графічному меню.

Відповідний фрагмент коду:

```
@bot.message_handler(commands=['help'])
def send_help(message):
    help_text = (
        "· Я можу надати таку інформацію:\n"
        "· Розклад занять\n"
        "· Де знаходиться корпус?\n"
        "· Як отримати довідку\n"
        "i Часті питання\n"
        "· Контакти деканату\n"
        "· Написати звернення\n\n"
        "_Використай меню або надішли відповідну команду._"
    )
    bot.send_message(message.chat.id, help_text, parse_mode='Markdown')
```

Пояснення:

- Повідомлення включає структурований список з emoji, що підвищує зручність читання.
- Використовується **parse_mode='Markdown'** для стилізації (курсиву, жирного тощо).
- Цей блок служить довідкою, якщо користувач не хоче натискати кнопки.

Результат надсилання відповіді від бота зображено на **рисунку 3.5**

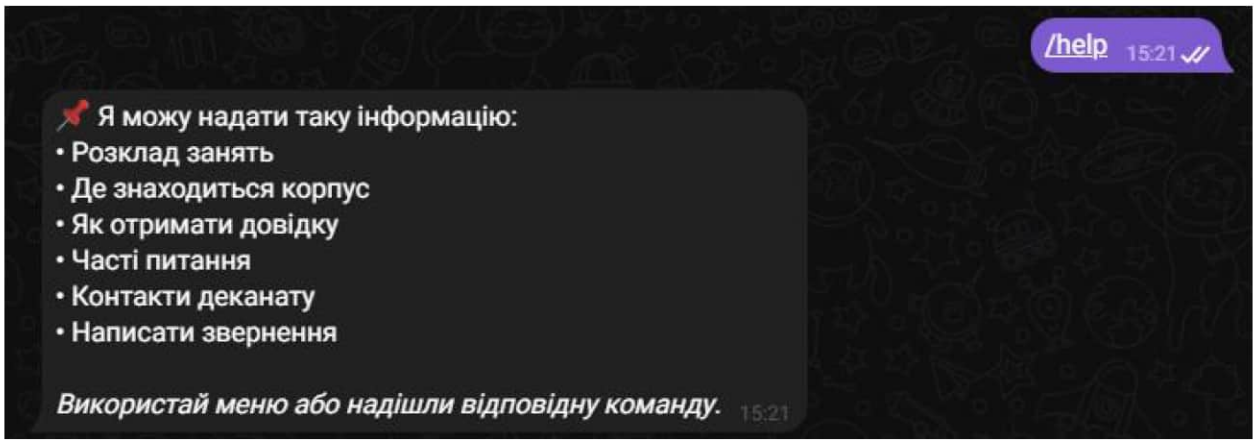

Рисунок 3.5 – Вивід допоміжної інформації після введення команди **/help**

3.Обробка запиту «· Де знаходиться корпус?»

Ця функція надає студенту зображення з картою розташування корпусів НТУ «Дніпровська політехніка» та текстовий опис розташування деканату.

Відповідний фрагмент коду:

```
@bot.message_handler(func=lambda message: message.text == "□ Де знаходиться корпус?")
def send_map(message):
    with open("21.jpg", "rb") as photo:
        bot.send_photo(
            message.chat.id,
            photo,
            caption="□ Кампус НТУ «Дніпровська політехніка»\nКорпус ФІТ: просп. Яворницького, 19, корпус 3, каб. 11"
```

Пояснення:

- ❖ Відкривається локальне зображення 21.jpg (мапа кампусу), що зберігається в тій же директорії, що й файл bot.py.
- ❖ bot.send_photo(...) надсилає зображення з коментарем, що містить адресу корпусу факультету інформаційних технологій.

Результат надсилання відповіді від бота зображено на **рисунку 3.6**

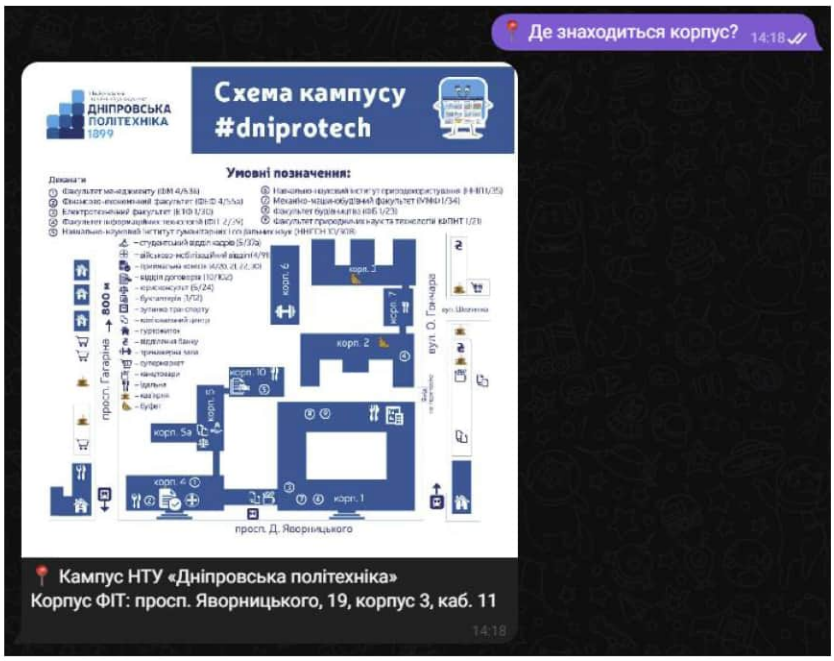

Рисунок 3.6 – Вивід карти розташування корпусів за запитом користувача

4.Обробка запиту «· Отримати довідку»

Бот надає студенту PDF-файл з інформацією щодо оформлення академічної довідки або зразком довідки. Це особливо зручно для тих, хто не має змоги звернутися до деканату особисто.

Відповідний фрагмент коду:

```
@bot.message_handler(func=lambda message: message.text == "· Отримати довідку")
def send_document(message):
    with open("академічна_довідка.pdf", "rb") as pdf:
        bot.send_document(
            message.chat.id,
            pdf,
            caption="· Ось інформація про академічну довідку ·"
```

Пояснення:

- Команда реагує на натискання кнопки «· Отримати довідку».
- Бот відкриває локальний файл академічна_довідка.pdf та надсилає його користувачу з відповідним підписом.

- ❖ Такий функціонал дозволяє швидко поширювати офіційні документи, без потреби в особистому зверненні.

Результат надсилання відповіді від бота зображено на **рисунку 3.7**

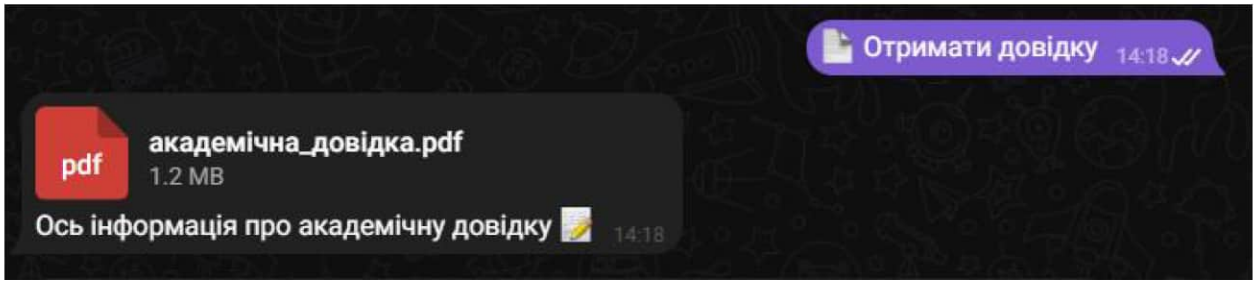

Рисунок 3.7 – Приклад відповіді бота з наданням довідки у форматі PDF

5.Обробка запиту «· Розклад занять»

Цей запит є одним із найчастіших серед студентів. Бот надає актуальне посилання на розклад занять, опублікований на офіційному сайті університету.

Відповідний фрагмент коду:

```
@bot.message_handler(func=lambda message: message.text == "· Розклад занять")
def send_schedule(message):
    bot.send_message(
        message.chat.id,
        "· Розклад занять доступний за посиланням:\nhttps://old.nmu.org.ua/ua/content/students/schedule/"
```

Пояснення:

- ❖ Використовується простий обробник з перевіркою на текст повідомлення.
- ❖ Бот надсилає гіперпосилання на зовнішній ресурс, де можна переглянути або завантажити актуальний розклад занять.

- Такий підхід дозволяє уникнути дублювання інформації та завжди мати актуальні дані.

Результат надсилання відповіді від бота зображено на **рисунку 3.8**

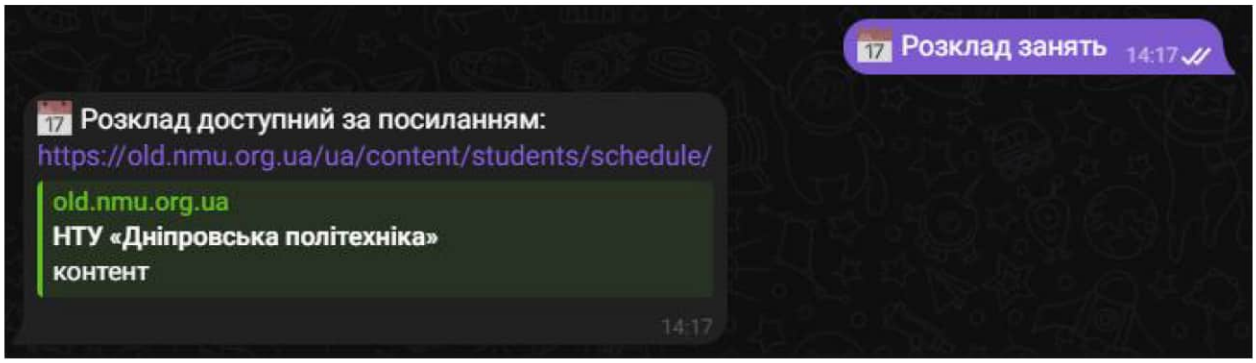

Рисунок 3.8 – Приклад відповіді бота з посиланням на розклад занять

6.Обробка запиту «· Контакти»

Цей функціонал дозволяє студенту миттєво отримати всю необхідну контактну інформацію деканату факультету інформаційних технологій: електронну пошту, телефон секретаря, адресу та ПІБ декана.

Відповідний фрагмент коду:

```
@bot.message_handler(func=lambda message: message.text == "· Контакти")
def send_contacts(message):
    bot.send_message(
        message.chat.id,
        "·· Декан факультету ІТ: Ірина Михайлівна Удовик\n"
        "· Email: udovyk.i.m@nmu.one\n"
        "· Секретар: +38 (099) 651-07-35\n"
        "· Приймальна комісія: (067) 447-51-52, (099) 271-40-80\n"
        "· Адреса: м. Дніпро, просп. Яворницького, 19, корпус 3,1 поверх"
```

Пояснення:

- Відповідь надається у форматі текстового повідомлення з чітко структурованими контактами.

- ❖ Включено найважливішу інформацію для швидкого звернення: пошта, телефони, адреса.
- ❖ Це зручно для студентів, які потребують термінової консультації або мають додаткові запити.

Результат надсилання відповіді від бота зображено на **рисунку 3.9**

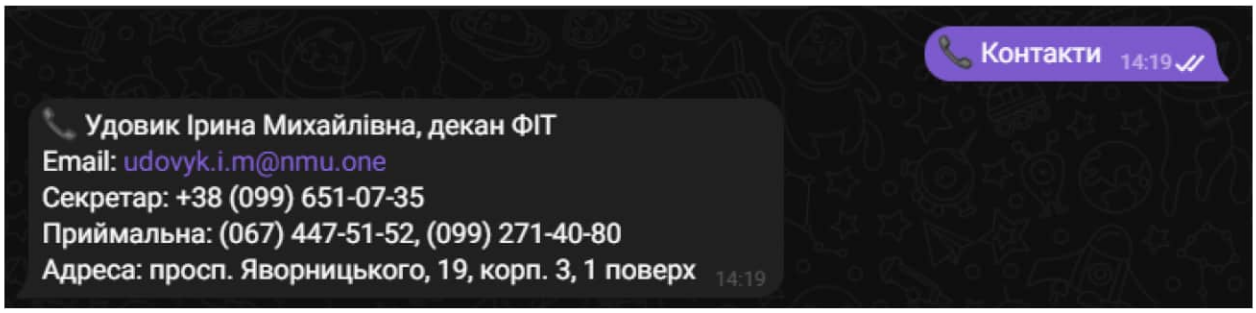

Рисунок 3.9 – Відповідь бота з контактами деканату

7.Обробка запиту «· Написати звернення»

Цей розділ дозволяє студенту зв'язатися з адміністрацією факультету за допомогою Telegram-бота, надіславши власне повідомлення. Така функція є корисною для передачі індивідуальних запитів або звернень.

Відповідний фрагмент коду:

```
@bot.message_handler(func=lambda message: message.text == "· Написати звернення")
def request_feedback(message):
    bot.send_message(
        message.chat.id,
        "· Напишіть своє звернення у відповідь на це повідомлення. Ми розглянемо його найближчим часом."
```

Пояснення:

- ❖ Бот реагує на натискання кнопки «· Написати звернення».
- ❖ Надсилає інструкцію користувачеві з проханням написати власне звернення у вільній формі.
- ❖ На базовому рівні бот не обробляє текст звернення, але воно може зберігатися або пересилатися адміністрації (у майбутніх версіях).

Результат надсилання відповіді від бота зображено на **рисунку 3.10**

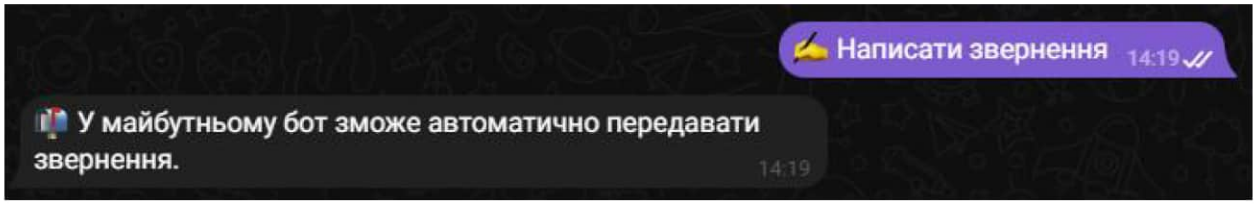

Рисунок 3.10 – Приклад відповіді бота з інструкцією щодо написання звернення

3.4 Висновки до розділу

У третьому розділі було детально розглянуто процес розробки та реалізації Telegram-бота для автоматизації взаємодії між студентами факультету інформаційних технологій та деканатом НТУ «Дніпровська політехніка».

Було визначено архітектуру системи, описано основні модулі, а також реалізовано ключові функції: надання розкладу занять, контактної інформації, можливості отримати академічну довідку, ознайомлення з розташуванням корпусу та створення звернень. Інтерфейс користувача реалізовано у вигляді зручного меню з кнопками, що значно полегшує навігацію та підвищує доступність сервісу.

Також у розділі наведено приклади фрагментів коду кожного функціонального блоку, що демонструє практичну реалізацію логіки роботи чат-бота. Візуальні ілюстрації підтверджують працездатність розробленої системи в реальному середовищі Telegram.

Результати реалізації засвідчують ефективність запропонованого підходу для підвищення якості обслуговування студентів та зменшення навантаження на співробітників деканату.

РОЗДІЛ 4. ТЕСТУВАННЯ ТА ОЦІНКА ЯКОСТІ СИСТЕМИ

4.1 Мета тестування

Метою тестування чат-бота є перевірка його працездатності, відповідності функціональним вимогам, стабільності під час взаємодії з користувачами, а також зручності інтерфейсу для студентів факультету інформаційних технологій.

Особливу увагу приділено правильності обробки типових запитів: надання контактної інформації, доступу до розкладу занять, розташування корпусів, а також можливості отримання довідок.

Тестування дозволяє виявити помилки на ранньому етапі, оцінити зручність використання бота та визначити напрями його подальшого вдосконалення.

4.2 Методи тестування

Для перевірки коректності роботи Telegram-бота було застосовано такі основні методи тестування:

1. **Функціональне тестування** — перевірка відповідності кожної реалізованої функції вимогам. Тестувалися всі кнопки меню, реакція на команди /start, /help, запити на довідку, надсилання зображень тощо.
2. **Тестування користувацьких сценаріїв** — моделювалися реальні ситуації, з якими може стикатися студент: запит розкладу занять, перегляд контактів, звернення до деканату.
3. **Тестування граничних випадків** — перевірка реакції бота на некоректні або незаплановані запити, наприклад введення випадкового тексту.

Загалом тестування проводилося у реальному середовищі Telegram з використанням мобільних і десктопних пристроїв.

4.3 Тестові сценарії

У таблтці 4.1 наведено результати тестування функціональності чат-бота.

Таблиця 4.1 – Результати тестування функціональності чат-бота

№	Сценарій	Дія користувача	Очікувана відповідь бота	Резуль-тат
1	Запуск бота	/start	Привітання та відображення меню	Успішно
2	Виклик довідки	Натискання кнопки «Довідка»	Надсилання PDF-файлу з академічною довідкою	Успішно
3	Перегляд розкладу	Натискання кнопки «Розклад»	Виведення посилання на актуальний розклад занять	Успішно
4	Перевірка розташування корпусу	Натискання кнопки «Де знаходиться корпус?»	Надсилання зображення зі схемою корпусів	Успішно
5	Отримання контактів	Натискання кнопки «Контакти»	Виведення e-mail, телефону та адреси деканату	Успішно
6	Отримання допомоги	/help	Інструкція щодо користування ботом	Успішно
7	Випадкова команда	Введення «123abc»	Повідомлення: «Вибач, я поки не розумію це повідомлення…»	Успішно
8	Надсилання звернення	Натискання кнопки «Написати звернення»	Повідомлення з подякою та перенаправлення до адміністрації	Успішно
9	Повторне натискання тієї ж кнопки	Двічі натискання «Розклад»	Повторне надсилання посилання на розклад	Успішно
10	Перевірка	Використання	Вся функціональність	Успішно

	всіх ф-цій на мобільному	бота зі смартфона	збережена, коректне відображення кнопок	

Всі заплановані сценарії були успішно протестовані.

Чат-бот коректно реагує на типові команди, відображає потрібну інформацію та взаємодіє з користувачем згідно з очікуваними сценаріями.

Жодних критичних помилок виявлено не було, бот стабільно працює в середовищі Telegram з мобільного та десктопного пристроїв.

4.4 Аналіз результатів тестування

Результати тестування засвідчили, що реалізований чат-бот повністю виконує заявлені функції. Усі ключові сценарії були успішно перевірені: бот стабільно приймає команди, правильно реагує на користувацькі запити, надсилає зображення та документи, а також забезпечує інформування студентів факультету у зручній та доступній формі.

Особливо важливим є те, що система коректно обробляє як стандартні звернення через кнопки, так і нестандартні текстові повідомлення, забезпечуючи зворотний зв’язок у разі невірного введення. Бот адаптований до мобільного та десктопного використання, не залежить від ОС користувача, що розширює його доступність.

Функціональність реалізована з урахуванням реальних потреб студентів, що підтверджено під час практичного тестування.

4.5 Висновки до розділу

У ході тестування було підтверджено працездатність і функціональну відповідність чат-бота поставленим вимогам. Система успішно пройшла перевірку за основними сценаріями взаємодії зі студентами факультету інформаційних технологій, продемонструвавши стабільну роботу, зручний інтерфейс та оперативність відповідей.

Отримані результати свідчать про готовність бота до використання в навчальному процесі для автоматизації комунікації між студентами та адміністрацією факультету. Надалі можливе розширення функціоналу та інтеграція з внутрішніми сервісами ВНЗ.

ЗАГАЛЬНІ ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було досліджено актуальні підходи до створення інтелектуальних систем автоматизованого обслуговування користувачів, зокрема чат-ботів, що базуються на нейронних мережах. На основі проведеного аналізу визначено проблеми у взаємодії студентів з деканатом, які можуть бути вирішені шляхом впровадження телеграм-бота.

У першому розділі було здійснено огляд концепції чат-ботів, типів їх архітектури, технологій обробки природної мови та інструментів розробки, що дозволило сформувати теоретичну базу для проєктування системи. У другому розділі проаналізовано потреби студентів факультету ФІТ, визначено функціональні вимоги до системи, побудовано структуру бази знань та обґрунтовано вибір інструментів реалізації.

У третьому розділі реалізовано прототип телеграм-бота з використанням мови Python та бібліотеки python-telegram-bot. Було розроблено користувацький інтерфейс, налаштовано логіку обробки запитів, забезпечено відповіді у вигляді текстів, PDF-документів та зображень.

У четвертому розділі проведено тестування бота, яке підтвердило його стабільну роботу, відповідність функціональним вимогам та зручність у використанні.

Результати роботи свідчать про доцільність застосування подібних чат-ботів для оптимізації комунікації у внутрішніх процесах вищих навчальних закладів. У майбутньому можливим є розширення системи, додавання авторизації, інтеграція з CRM або студентським кабінетом.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ковальов В.В. Нейронні мережі та глибинне навчання. — Київ: Наукова думка, 2020. — 312 с.
2. Васильєв С.А. Технології штучного інтелекту в інформаційних системах. — Харків: ХНУРЕ, 2021. — 276 с.
3. Goldberg Y. Neural Network Methods in Natural Language Processing. — Morgan & Claypool, 2017.
4. Vaswani A. et al. Attention is All You Need // NeurIPS, 2017. Hugging Face Documentation — https://huggingface.co/docs
5. Telegram Bot API Documentation [Електронний ресурс] — https://core.telegram.org/bots/api (дата звернення: 12.04.25)
6. Python Telegram Bot (python-telegram-bot) [Електронний ресурс] — https://docs.python-telegram-bot.org/ (дата звернення: 12.04.25)
7. Rasa NLU [Електронний ресурс] — https://rasa.com/docs/rasa/ (дата звернення: 12.04.25)
8. OpenAI GPT documentation [Електронний ресурс] — https://platform.openai.com/docs (дата звернення: 12.04.25) Stack Overflow [Електронний ресурс] — https://stackoverflow.com/ (дата звернення: 12.04.25)
9. Medium articles (chatbot architecture, NLP, transformers) [Електронний ресурс] — https://medium.com (дата звернення: 12.04.25)
10. Офіційний сайт НТУ «Дніпровська політехніка» [Електронний ресурс] — https://nmu.org.ua/ (дата звернення: 12.04.25)

YouTube канали:

11. **Tech with Tim** [Електронний ресурс] — *"How to Create a Telegram Bot using Python!"* https://www.youtube.com/watch?v=1a4w1M89Pzs (дата звернення: 12.04.25)
12. **Rasa YouTube Channel** [Електронний ресурс] — *"Building a chatbot with Rasa – Tutorial Series"* https://www.youtube.com/c/RasaHQ (дата звернення: 12.04.25)

ДОДАТОК А

Фрагменти коду Telegram-бота, реалізованого на Python

```
# □ Імпорт необхідних бібліотек
import telebot
from telebot import types
bot = telebot.TeleBot("TOKEN")
# □ Обробка команди /start
@bot.message_handler(commands=['start'])
def send_welcome(message):
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
    btn1 = types.KeyboardButton("□ Розклад занять")
    btn2 = types.KeyboardButton("□ Де знаходиться корпус?")
    btn3 = types.KeyboardButton("□ Отримати довідку")
    btn4 = types.KeyboardButton("ℹ Часті питання")
    btn5 = types.KeyboardButton("□ Контакти")
    btn6 = types.KeyboardButton("□ Написати звернення")
    markup.add(btn1, btn2, btn3, btn4, btn5, btn6)
    bot.send_message(message.chat.id,
                     "Вітаю! Це бот факультету ФІТ. Оберіть потрібний розділ □",
                     reply_markup=markup)
# □ Команда /help
```

```
@bot.message_handler(commands=['help'])
def send_help(message):
    help_text = (
        "□ Я можу надати таку інформацію:\n"
        "• Розклад занять\n"
        "• Де знаходиться корпус\n"
        "• Як отримати довідку\n"
```

Продовження ДОДАТКУ А

```
        "• Часті питання\n"
        "• Контакти деканату\n"
        "• Написати звернення"
    )
    bot.send_message(message.chat.id, help_text, parse_mode='Markdown')
```

Обробка кнопки "Де знаходиться корпус?"

```
# □ Обробка кнопки "Де знаходиться корпус?"
@bot.message_handler(func=lambda message: message.text == "□ Де знаходиться корпус?")
def send_map(message):
    with open("21.jpg", "rb") as photo:
        bot.send_photo(
            message.chat.id,
            photo,
            caption="□ Кампус НТУ «Дніпровська політехніка»\nКорпус ФІТ: просп. Яворницького, 19, корпус 3, каб. 11"
        )
```

Обробка інших запитів (кнопки з меню)

```
# □ Обробка інших кнопок
@bot.message_handler(func=lambda message: True)
def handle_text(message):
    if message.text == "□ Розклад занять":
```

bot.send_message(message.chat.id, "□ Розклад доступний за посиланням:\nhttps://old.nmu.org.ua/ua/content/students/schedule/")

elif message.text == "□ Отримати довідку":

with open("академічна_довідка.pdf", "rb") as pdf:

bot.send_document(message.chat.id, pdf, caption="Ось інформація про академічну довідку □")

Продовження ДОДАТКУ А

elif message.text == "**i** Часті питання":

bot.send_message(message.chat.id, "**i** Часті питання включають: як подивитись розклад, як отримати довідку, як звертатись до деканату.")

elif message.text == "□ Контакти":

bot.send_message(message.chat.id, "□ Удовик Ірина Михайлівна, декан ФІТ\nEmail: udovyk.i.m@nmu.one\nСекретар: +38 (099) 651-07-35\nПриймальна: (067) 447-51-52, (099) 271-40-80\nАдреса: просп. Яворницького, 19, корп. 3, 1 поверх")

elif message.text == "□ Написати звернення":

bot.send_message(message.chat.id, "□ У майбутньому бот зможе автоматично передавати звернення.")

else:

bot.send_message(message.chat.id, "□ Вибач, я поки не розумію це повідомлення. Обери варіант з меню.")

Завершення ДОДАТКУ А

Запуск бота

```
# Запуск бота
bot.polling()
```

ДОДАТОК Б

Скріншоти роботи чат-бота Telegram

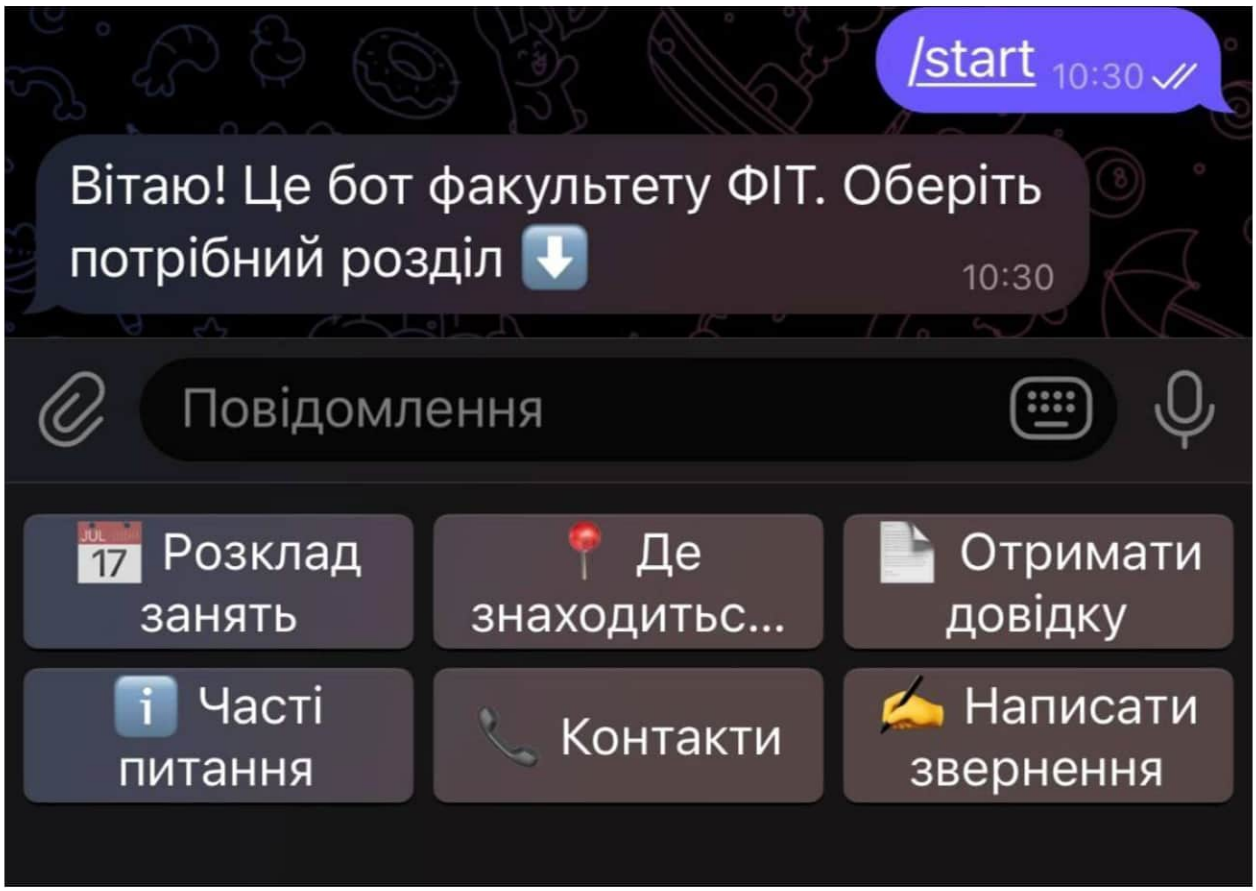

Рисунок Б.1 – Вітальне повідомлення чат-бота після запуску команди /start

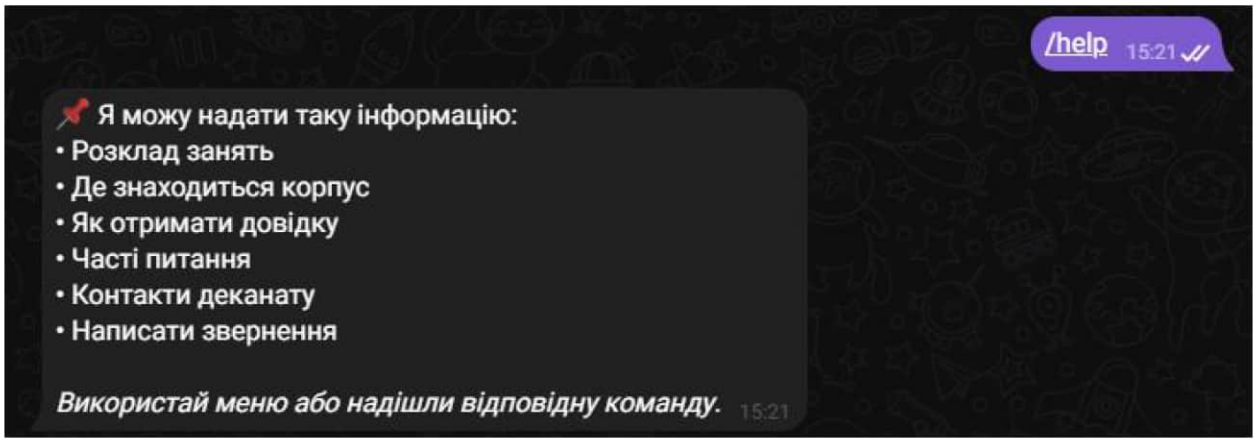

Рисунок Б.2 – Команда /help

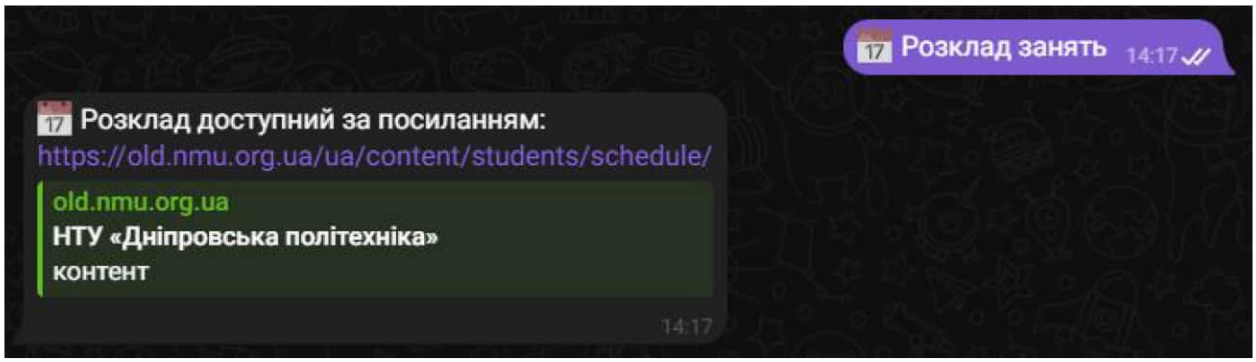

Рисунок Б.3 – кнопка Розклад занять

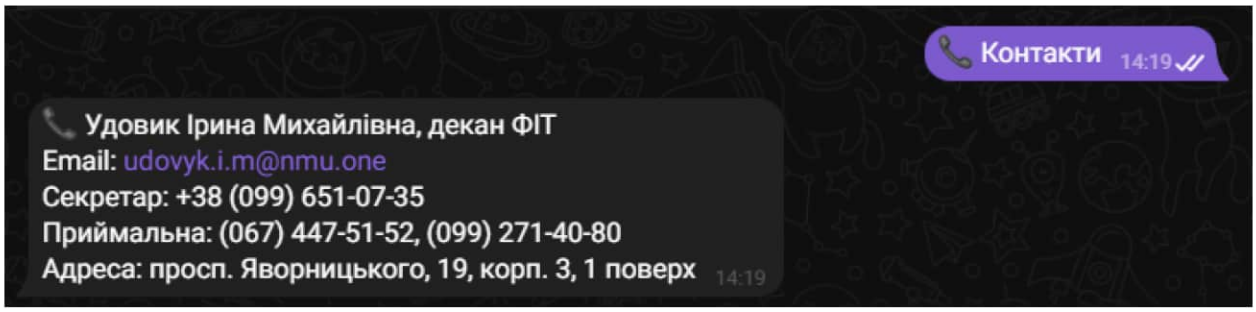

Рисунок Б.4 - кнопка Контакти

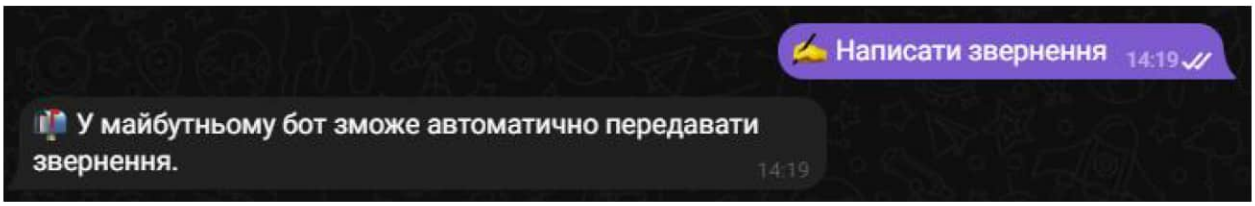

Рисунок Б.5 - кнопка Написати звернення

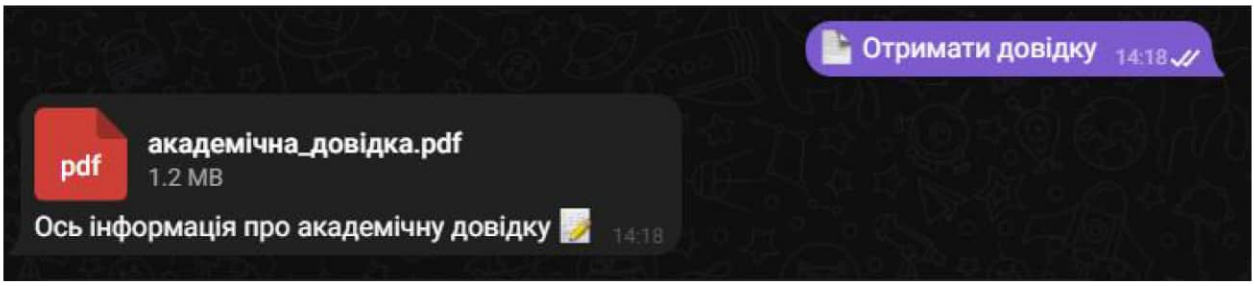

Рисунок Б.6 - кнопка Отримати довідку

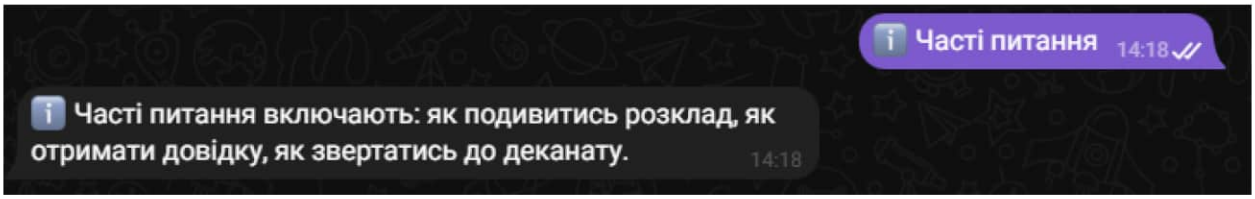

Рисунок Б.7 - кнопка Часті питання

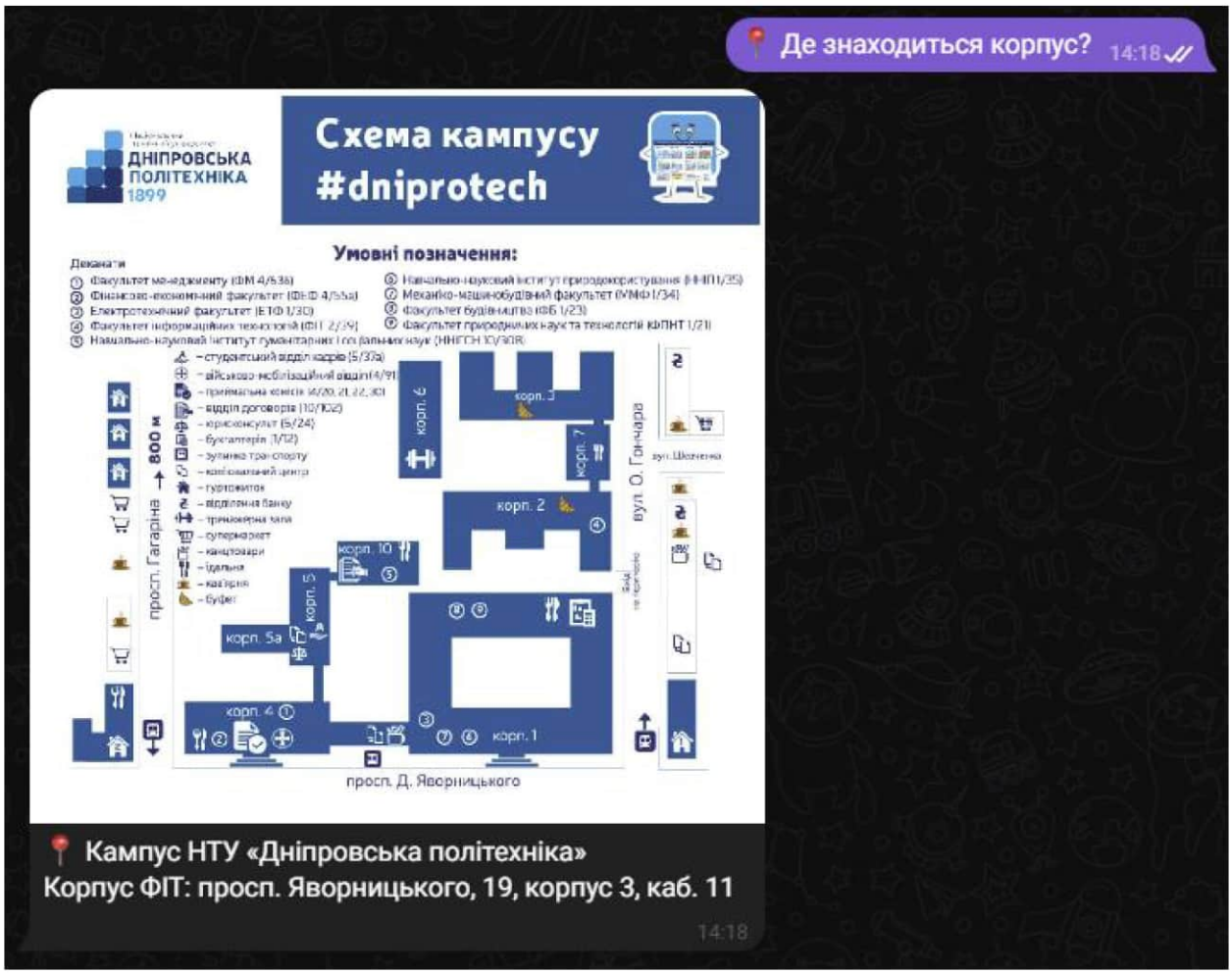

Рисунок Б.8 - кнопка Де знаходиться корпус?