МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ЦЕНТР ЗАОЧНОЇ, ДИСТАНЦІЙНОЇ ТА ВЕЧІРНЬОЇ ФОРМ НАВЧАННЯ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК
СЕКЦІЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРОЕКТУВАННЯ

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему «Інформаційний чат-бот месенджеру Telegram для супроводження проведення курсів навчального центру секції інформаційних технологій проектування кафедри комп'ютерних наук»
за спеціальністю 122 «Комп'ютерні науки та інформаційні технології», освітньо-професійна програма «Інформаційні технології проектування»

Виконавець роботи: студент групи Ітдн-61С Міросєді Ігорь Андрійович

Кваліфікаційна робота бакалавра
захищена на засіданні ЕК
з оцінкою _____«___»_____2020 р.

Науковий керівник _____ к.т.н., доц., Парфененко Ю.В.
(підпис) (науковий ступінь, вчене звання, прізвище та ініціали)

Голова комісії _____ _____
(підпис) (науковий ступінь, вчене звання, прізвище та ініціали)

Засвідчую, що у цій дипломній роботі немає
запозичень з праць інших авторів
без відповідних посилань.

Студент _____
(підпис)

Суми-2020

Сумський державний університет
Центр заочної, дистанційної та вечірньої форм навчання
Кафедра комп'ютерних наук
Секція інформаційних технологій проектування
Спеціальність 122 «Комп'ютерні науки та інформаційні технології»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
Зав. секцією ІТП

_____ В. В. Шендрик
«__» _____ 2020 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА СТУДЕНТУ

Міросєді Ігорь Андрійович

1 Тема роботи Інформаційний чат-бот месенджеру Telegram для супроводження проведення курсів навчального центру секції інформаційних технологій проектування кафедри комп'ютерних наук

керівник роботи Парфененко Юлія Вікторівна ,

затверджені наказом по університету від «21» травня 2020 р. № 0608-III

2 Строк подання студентом роботи « 6 » червня 2020 р.

3 Вхідні дані до роботи технічне завдання на розробку інформаційного чат-боту в месенджері Telegram

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1) Аналіз існуючої екосистеми чат-ботів для додатку Telegram, 2) Аналіз методів створення чат-ботів для навчальних центрів, 3) Моделювання роботи чат-боту, 4) Розробка чат-боту

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) мета та задачі, дослідження аналогів, функціональне моделювання, моделювання бази даних, моделювання варіантів використання, архітектура чат-боту, розробка модулів чат-боту, реалізація моделі бази даних, створення дистрибутиву, інтеграція у Telegram.

6. Консультанти розділів роботи:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7 Дата видачі завдання 01.10.2019

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Оформлення планування робіт	До 08.03.2020	
2	Оформлення технічного завдання	До 15.03.2020	
3	Проведення аналізу предметної області	До 22.03.2020	
4	Проведення структурно-функціонально моделювання процесів	До 10.04.2020	
5	Розробка додатку	До 11.05.2020	
6	Тестування додатку	До 17.05.2020	
7	Здача пояснювальної записки та файлів розробленого проекту	До 01.06.2020	

Студент _____ Міросєді І.А.
(підпис)

Керівник _____ Парфененко Ю.В.
(підпис)

РЕФЕРАТ

Тема кваліфікаційної роботи бакалавра «Інформаційний чат-бот месенджеру Telegram для супроводження проведення курсів навчального центру секції інформаційних технологій проектування кафедри комп'ютерних наук».

Метою кваліфікаційної роботи бакалавра є створення інформаційної системи у вигляді чат-боту для додатку Telegram, яка могла би використовуватися студентами курсів НКНМЦ ІТП та людьми у них зацікавленими.

У ході виконання кваліфікаційної роботи бакалавра було розроблено три частини чат-боту:

- модуль роботи з базою даних;
- серверна частина;
- графічний інтерфейс серверної частини;

Результатом проведеної роботи є розроблений чат-бот месенджеру Telegram для супроводження проведення курсів навчального центру секції інформаційних технологій проектування кафедри комп'ютерних наук.

Кваліфікаційна робота містить 99 сторінок, 12 таблиць, 56 ілюстрацій, список літератури із 20 найменувань, 4 додатки.

Ключові слова: чат-бот, Telegram, Python, супроводження курсів, інформаційна система.

ЗМІСТ

ВСТУП 6
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 7
1.1 Дослідження актуальності проблеми 7
1.2 Аналіз існуючих чат-ботів Telegram 9
2 ПОСТАНОВКА ЗАДАЧІ ПРОЕКТУ 20
2.1 Мета та задачі 20
2.2 Вибір засобів реалізації 20
3 МОДЕЛЮВАННЯ ЧАТ-БОТУ 22
3.1 Структурно-функціональне моделювання 22
3.2 Проектування моделі бази даних 25
3.3 Моделювання варіантів використання 29
4 РОЗРОБКА ЧАТ-БОТУ 31
4.1 Розробка архітектури чат-боту 31
4.2 Розробка модулів чат-боту 32
4.3 Реалізація моделі бази даних 39
4.4 Створення дистрибутиву 42
4.5 Інтеграція чат-боту у Telegram 44
ВИСНОВКИ 48
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 49
ДОДАТОК А 52
ДОДАТОК Б 57
ДОДАТОК В 68
ДОДАТОК Г 78

ВСТУП

Сучасні смартфони дозволяють своїм власникам використовувати численні додатки, які забезпечують доступ до різноманітних інформації та послуг за рекордно низький час. Не оминуло це й сферу освіти, що видно з оціночного розміру ринку, проведеного Technavio у квітні 2019 року [1].

Одним із найбільш розповсюджених і скачуваних типів додатків для смартфонів є месенджери [2], які дозволяють поширювати і отримувати новини та інформацію у найкоротші терміни. Більшість месенджерів підтримують роботу з ботами — програмними додатками у екосистемі месенджерів, які надають користувачу доступ до певного функціоналу.

Метою кваліфікаційної роботи бакалавра є розроблення інформаційної системи у вигляді чат-боту для додатку Telegram, який може використовуватися студентами курсів НКНМЦ ІТП та людьми у них зацікавленими.

Щоб досягнути мету проекту, потрібно буде вирішити такі задачі:

- провести аналіз існуючої екосистеми чат-ботів для додатку Telegram;
- провести аналіз методів створення чат-ботів для навчальних центрів;
- виконати моделювання роботи чат-боту;
- розробити чат-бот та провести його тестування.

Практичне значення роботи проявляється у створенні чат-боту для супроводження проведення курсів НКНМЦ ІТП.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Дослідження актуальності проблеми

На сьогодні переважна кількість навчальних центрів має сайт, де відвідувачі можуть дізнатися про центр, доступні курси, контактну інформацію тощо. Не є виключенням є і учбовий центр секції інформаційних технологій проектування кафедри комп'ютерних наук Сумського державного університету НКНМЦ ІТП [3]. Скріншот головної сторінки сайту НКНМЦ ІТП можна побачити на рис. 1.1.

Рисунок 1.1 – Скріншот сайту центру НКНМЦ ІТП

Цей сайт є лише візитівкою центру, на якій розміщені контактні дані та загальна інформація про курси, але немає функціоналу супроводження курсів, наприклад, нагадування слухачам про завдання, які слід виконати.

Функціонал супроводження курсів може бути реалізований у вигляді окремого web- або мобільного додатку. Але обидва підходи можуть зіткнутися з деякою кількістю контраргументів.

Для web-додатку це:

- потреба у окремій інфраструктурі (окремий сервер чи хостинг);
- згідно дослідженням Flurry Analytics, користувачі віддають перевагу мобільним додаткам, а не web-додаткам [4].

Для мобільного додатку це:

- потреба підтримки сумісності з різними операційними системами та їх версіями;
- вимога до користувача встановлювати ще один додаток.

Для обох підходів характерна достатньо висока вартість розробки.

Не можна також не враховувати дослідження The Standish Group, у ході якого було виявлено, що лише приблизно 7% функцій програм та додатків використовується користувачами на постійній основі [5], що може привести до даремно витрачених коштів.

Переважна більшість слухачів курсів користуються месенджерами на мобільних пристроях, тому доцільною є розробка чат-боту саме для месенджера, а не у вигляді окремого мобільного додатку, який потребує встановлення та завантаження.

Враховуючи усе це та зростаючу популярність месенджера Telegram [6], можна зробити висновок, що є актуальною розробка чат-боту супроводження курсів саме для додатку Telegram.

1.2 Аналіз існуючих чат-ботів Telegram

Перед початком розробки був проведений аналіз існуючих чат-ботів. Були обрані наступні 3 чат-боти організацій, які проводять курси:

– “Онлайн Школа Дизайна”;
– “Insta_kurs Бесплатные курсы”;
– “Фотошоп Мастер”.

Нажаль, обрані чат-боти мають обмежений функціонал супроводження курсів, або зовсім його не мають, тому не можуть бути розглянуті як прямі аналоги. Через це, для більш повного аналізу, були обрані також п’ять популярних чат-ботів:

– Alert Bot;
– Cambridge Dictionary;
– Mitsuku;
– Get Media Bot;
– Wikipedia Search Bot.

Інтерфейс роботи з цими чат-ботами показано на рис. 1.1. – 1.9.

Були виділені наступні характеристики, на які необхідно звертати увагу при аналізі чат-ботів:

– наявність обробки натуральної мови — чи вміє бот обробляти натуральну мову користувача, або розуміє лише команди;
– персистентність — здатність бота зберігати інформацію користувача навіть після завершення діалогу;
– наявність inline-режиму — можливість використовувати бота у спільнотах, результат взаємодії доступний усім учасникам;
– простота інтерфейсу — наскільки простий інтерфейс має бот.

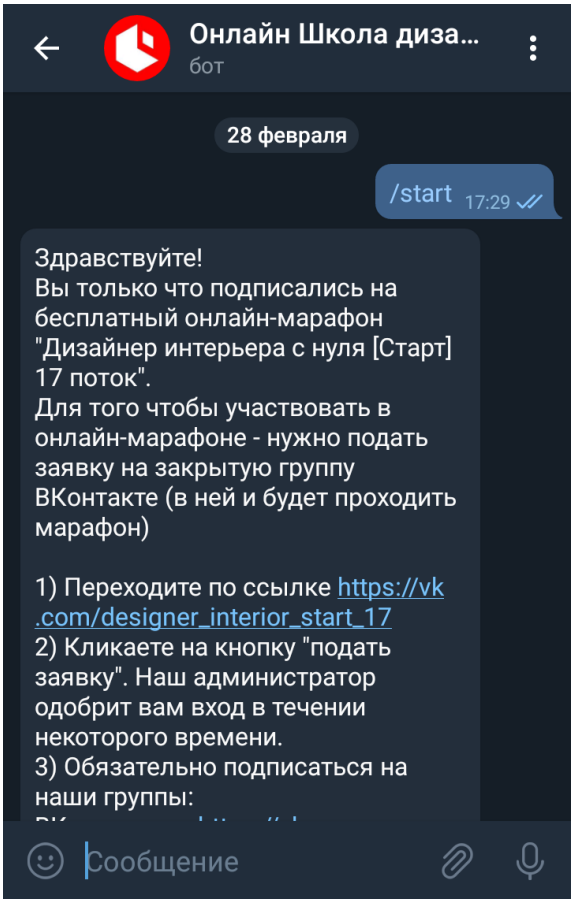

Рисунок 1.2 – Скріншот роботи з “Онлайн Школа Дизайна”

«Онлайн Школа Дизайна» — бот супроводження безкоштовних курсів дизайну інтер’єрів. Дозволяє користувачу ознайомитися з інформацією про курс та умовами вступу, та використовується для масової розсилки повідомлень, яка, судячи з усього, викликається вручну. Має простий інтерфейс з лише одною командою для початку роботи. Бот не має inline-режиму і персистентності. Приклад роботи з ботом наведений на рис 1.2.

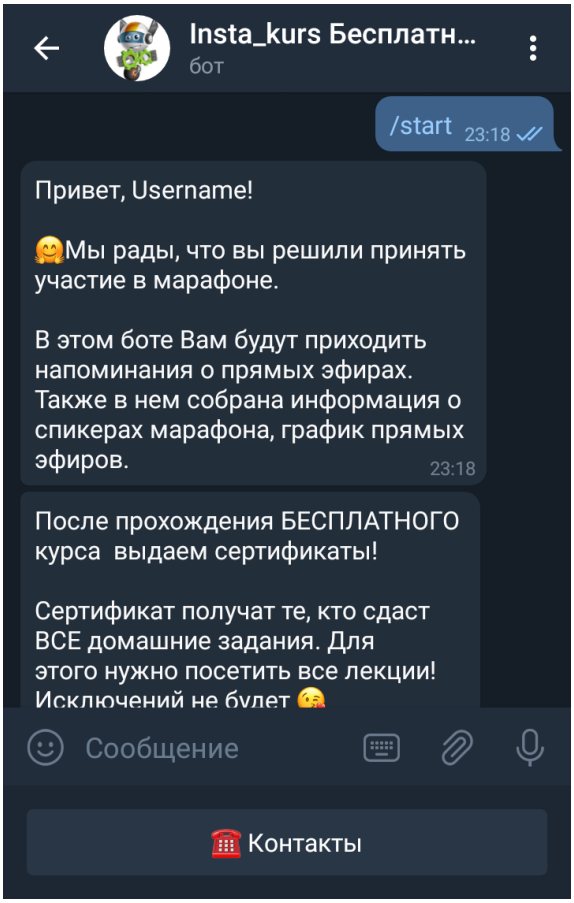

Рисунок 1.3 – Скріншот роботи з “Insta_kurs Бесплатные курсы”

«Insta_kurs Бесплатные курсы» — бот супроводження безкоштовних курсів створення та розкрути Instagram-каналів. Дозволяє користувачу ознайомитися з інформацією про курс, та використовується для масової розсилки повідомлень про початок прямих ефірів. Має простий інтерфейс з лише одною кнопкою «Контакти». Бот не має inline-режиму, а персистентність, судячи з усього, використовується лише для запам’ятовування імені користувача. Приклад роботи з ботом наведений на рис 1.3.

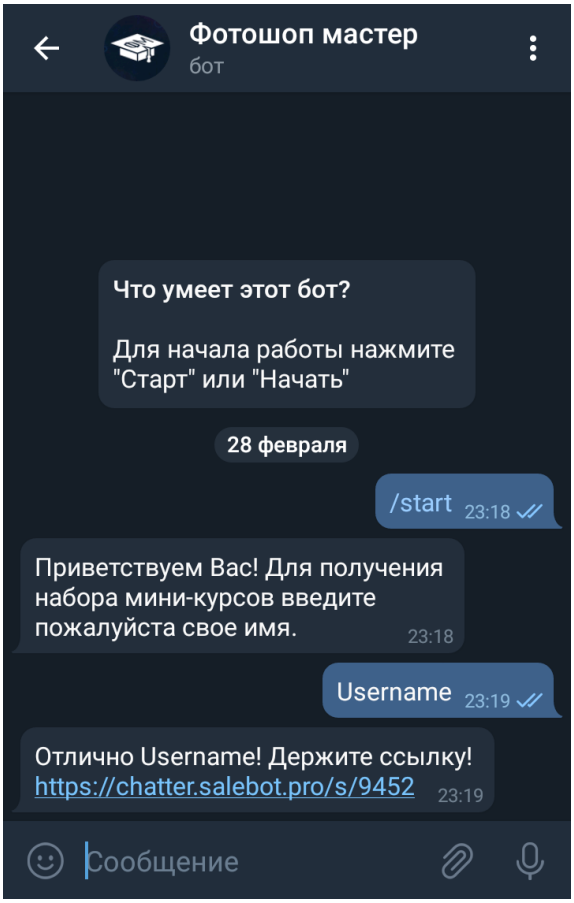

Рисунок 1.4 – Скріншот роботи з “Фотошоп Мастер”

«Фотошоп Мастер» — бот супроводження курсів фото- та відеообробки. Використовується лише для надання посилання на сайт центру. Має простий інтерфейс з лише одною командою для початку роботи. Бот не має ні персистентності, ні inline-режиму. Приклад роботи з ботом наведений на рис 1.4.

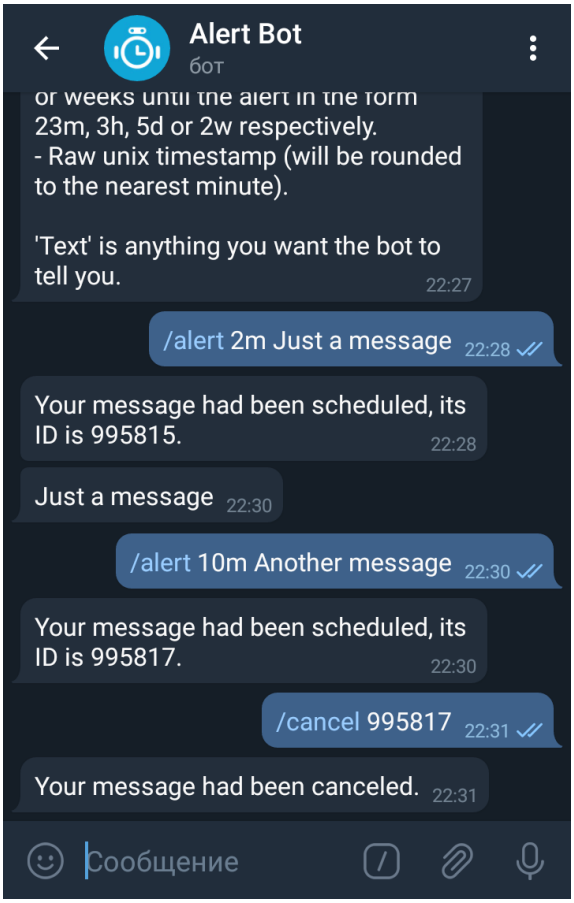

Рисунок 1.5 – Скріншот роботи з Alert Bot

Alert Bot — бот-нагадувач, який присилає задані користувачем повідомлення у обраний час. Має простий інтерфейс, який керується тільки командами. Команд усього три, /alert для створення нагадування, /cancel для видалення нагадування, та /help для показу довідки. Користувач може створювати декілька нагадувань, кожне із яких має свій ідентифікаційний номер. Бот не має inline-режиму і працює тільки у вікні діалогу з користувачем. Приклад роботи з ботом наведений на рис 1.5.

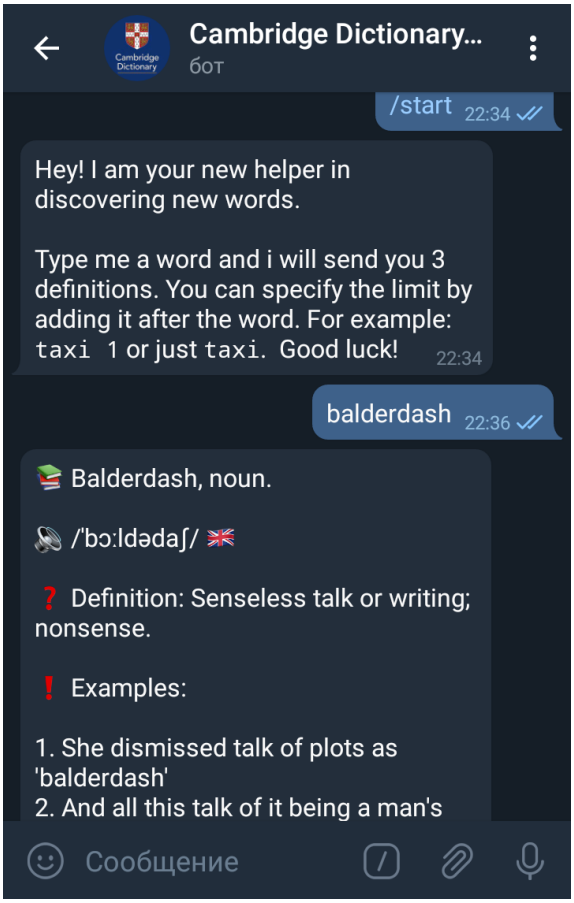

Рисунок 1.6 – Скріншот роботи з Cambridge Dictionary Bot

Cambridge Dictionary Bot — бот, здатний проводити пошук в Кембриджському словнику англійської мови. Має надзвичайно простий інтерфейс, який складається лише із двух команд, /search для пошуку цікавого користувачу слова, та /random для видачі випадкового слова. Бот показує визначення слова, його вимову за допомогою фонетичної транскрипції, та декілька прикладів використання слова у реченні. Бот не має inline-режиму чи персистентності. Приклад роботи з ботом наведений на рис 1.6.

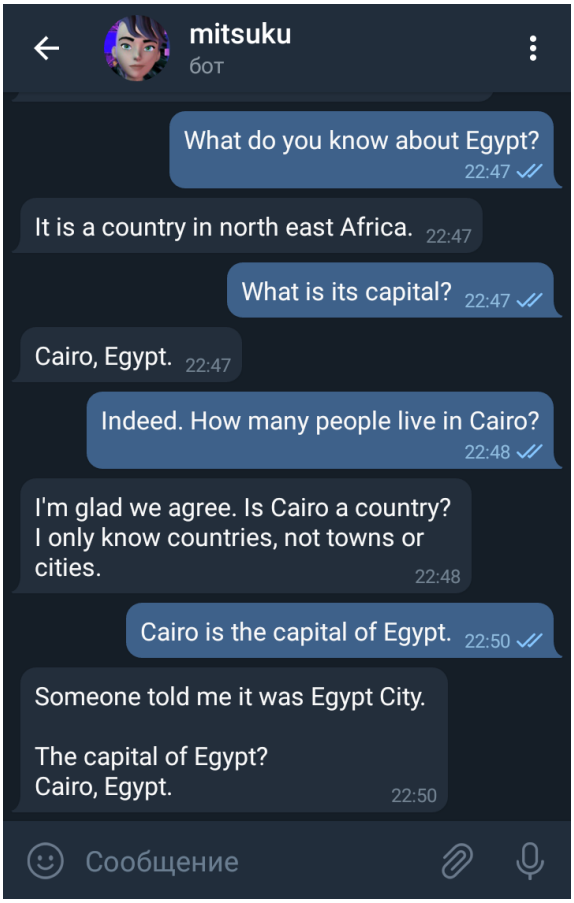

Рисунок 1.7 – Скріншот роботи з Mitsuku

Mitsuku — розмовний чат-бот, який розробляється з 2003 року [7] та чотири рази одержав премію Льобнера [8]. Інтерфейс лише текстовий, чат-бот не має команд, а тільки обробляє натуральну мову повідомлень користувача. Mitsuku вміє:

– розповідати жарти та історії;
– читати поеми та гороскопи;
– показувати картинки та веб-сайти;
– показувати прогноз погоди для заданих користувачем місць;
– розповідати факти про задані користувачем дати.

Inline-режиму не має. Приклад роботи з чат-ботом наведений на рис 1.7.

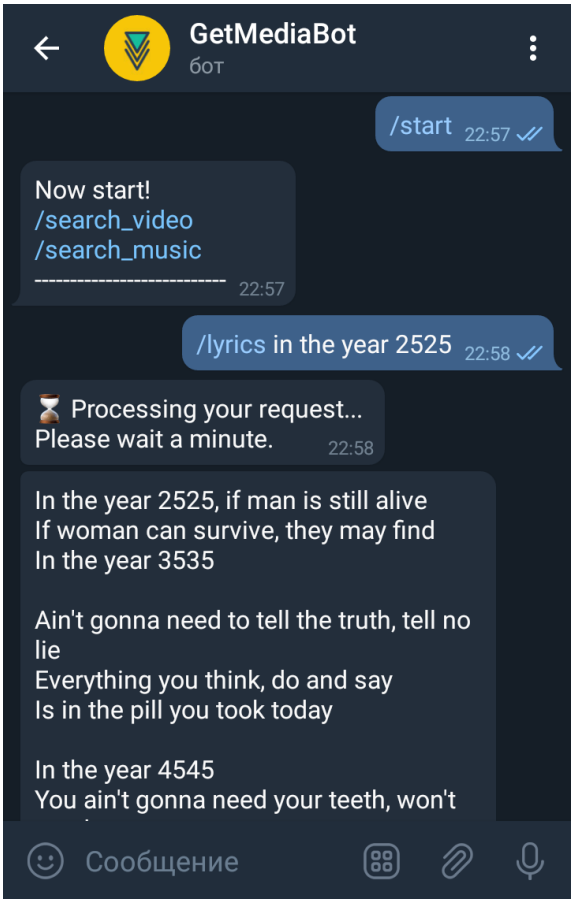

Рисунок 1.8 – Скріншот роботи з Get Media Bot

Get Media Bot — бот, здатний шукати та завантажувати музику, відео та тексти пісень. Має достатньо складний інтерфейс з великою кількістю кнопок та функцій. Бот має inline-режим, який дозволяє ділитися посиланнями з іншими учасниками спільноти. Бот має персистентні налаштування та статистику. Велика кількість функцій та громіздкий інтерфейс значно знижують зручність користування. Приклад роботи з ботом наведений на рис 1.8.

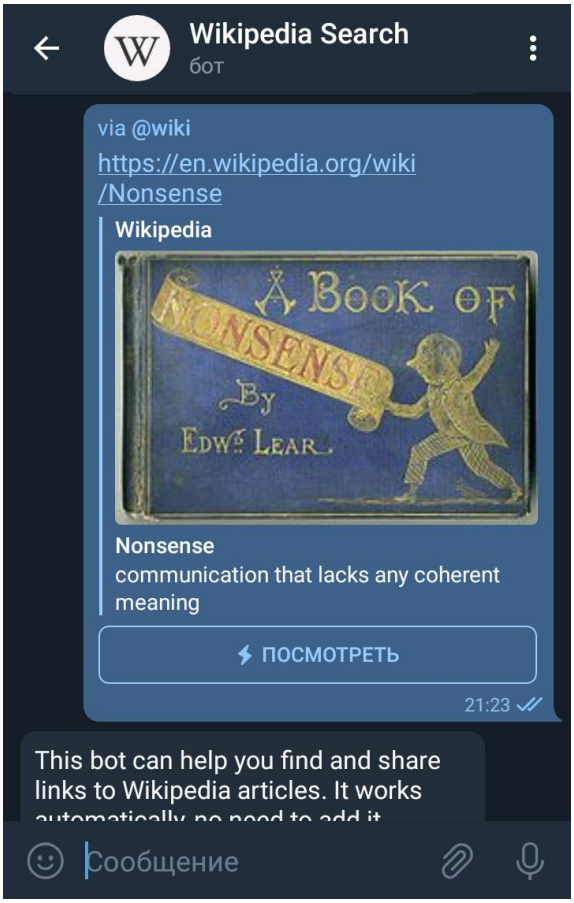

Рисунок 1.9 – Скріншот роботи з Wikipedia Search Bot

Wikipedia Search Bot — бот, який дозволяє здійснювати пошук у Вікіпедії. Має украй простий інтерфейс без команд. Бот працює виключно в inline-режимі, що дозволяє використовувати його у будь-якій ситуації. Бот не має персистентності. Приклад роботи з ботом наведений на рис 1.9.

Результати дослідження по кожному із критеріїв були занесені в табл. 1.1.

Таблиця 1.1— Порівняння характеристик ботів

Критерії	Обробка натуральної мови	Персистентність	Inline-режим	Простота інтерфейсу
«Онлайн Школа Дизайна»	–	–	–	+
«Insta_kurs Бесплатные курсы»	–	+	–	+
«Фотошоп Мастер»	–	–	–	+
Alert Bot	–	+	–	+
Cambridge Dictionary Bot	–	–	–	+
Mitsuku	+	+	–	–
Get Media Bot	–	+	+	–
Wikipedia Search Bot	–	–	+	+

Завдяки проведенню аналізу було виявлено, що при створенні чат-бота для супроводження проведення курсів НКНМЦ ІТП треба мати на увазі наступне:

- велика кількість слабо пов’язаних функцій призводить до зменшення зручності інтерфейсу;
- навіть бот-переможець змагань, заснованих на тесті Тьюринга, має надзвичайно багато проблем з обробкою натуральної мови.

З оглядкою на проведений аналіз, можна визначитися з бажаними характеристиками чат-бота для даного проекту:

- обробка натуральної мови, якщо і повинна бути присутньою, має бути простою та обмеженою у масштабі, користувач не повинен писати боту об’ємні повідомлення для виклику простих функцій;
- чат-бот повинен мати властивість персистентності, зберігати дані про користувача;
- чат-бот навряд чи буде використовуватися у спільнотах, тому inline-режим не є бажаним;
- простота інтерфейсу має бути первинним пріоритетом розробки, чат-бот не повинен мати ускладнений інтерфейс із багатьма рівнями вкладеності.

2 ПОСТАНОВКА ЗАДАЧІ ПРОЕКТУ

2.1 Мета та задачі

Метою проекту є створення чат-бота для додатку Telegram для супроводження проведення курсів НКНМЦ ІТП.

На основі аналізу, проведеного у попередньому розділі, можна зробити наступні висновки:

- чат-бот має бути вузькоспеціалізованим, що спростить не тільки процес розробки й тестування, а і використання;
- використання природної мови для керування чат-ботом є достатньо незручним, особливо у випадку вирішення утилітарних задач, таких як пошук по існуючому сайту чи словнику, або створенні нагадувань.

Враховуючи це, можна скласти список бажаних функцій.

- пошук курсів за ключовими словами;
- видача інформації про наступні завдання і строки їх виконання у курсах за прямим запитом користувача;
- автоматичне нагадування про строки виконання завдань.

Технічне завдання на розробку чат-бота представлене у додатку А. Планування робіт з розробки чат-бота наведене у додатку Б.

2.2 Вибір засобів реалізації

Для реалізації проекту була обрана мова програмування Python, інтерпретована мова високого рівня загального призначення. Python дозволяє

швидко створювати прототипи програмного забезпечення та проводити подальші ітерації. Основними перевагами цієї мови є:

- мультипарадигменість;
- простота синтаксису;
- велика кількість навчаючих матеріалів;
- розвинута екосистема сторонніх модулів та бібліотек;
- велика кількість open-source проектів, що полегшую вивчення ідіом мови та вирішення поширених задач.

Для вирішення проблем, пов’язаних зі створенням чат-боту була обрана бібліотека python-telegram-bot, яка імплементує не тільки API Telegram, а й містить декілька високорівневих класів для спрощення процесу розробки [9].

Для реалізації персистентності була обрана реляційна система керування базами даних SQLite. Ця самодостатня безсерверна система, яка не потребує налаштування, є альтернативою класичним клієнт-серверним РСКБД для локального зберігання інформації. SQLite часто використовується як вбудована база даних у веб-браузерах, додатках для смартфонів, тощо [10].

3 МОДЕЛЮВАННЯ ЧАТ-БОТУ

3.1 Структурно-функціональне моделювання

Першим етапом моделювання має бути побудова функціональної моделі системи, яка допоможе визначити та описати функції та процеси системи. Для цього буде використана методологія IDEF0 (ICAM Definition for Function Modeling), яка має декілька рівнів деталізації [11].

Спочатку створюється модель рівня А-0, найбільш загального із рівнів, яка має один функціональний блок. Основним процесом є створення інформаційного чат-боту для супроводження проведення курсів. Вхідними елементами процесу є технічне завдання та токен API Telegram. Елементами контролю є документація Python та Telegram. Елементами механізму є замовник, розробник, мова програмування Python, СКБД SQLite та додаток Telegram. Вихідним елементом є працюючий чат-бот. Діаграму рівню А-0 представлено на рис. 3.1.

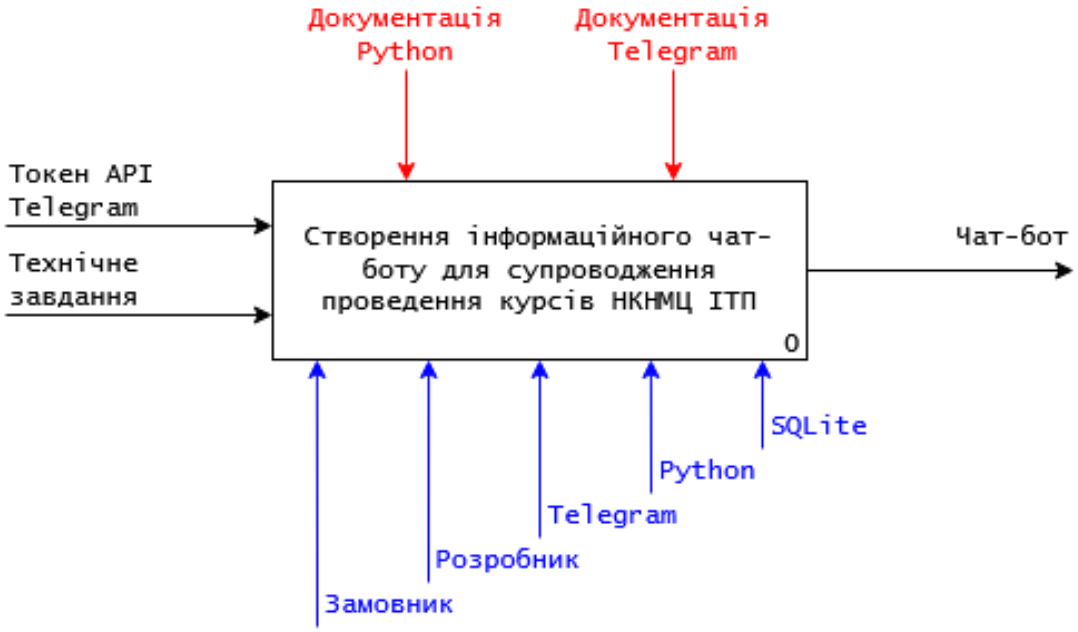

Рисунок 3.1 — Діаграма рівню А-0

Для деталізації опису системи проводиться подальша функціональна декомпозиція системи. Процес №0, «Створення інформаційного чат-боту для супроводження проведення курсів НКНМЦ ІТП», був розбитий на три субпроцеси:

– аналіз предметної області;

– практична реалізація;

– тестування та здача проекту.

Діаграму декомпозиції основного процесу можна побачити на рис. 3.2.

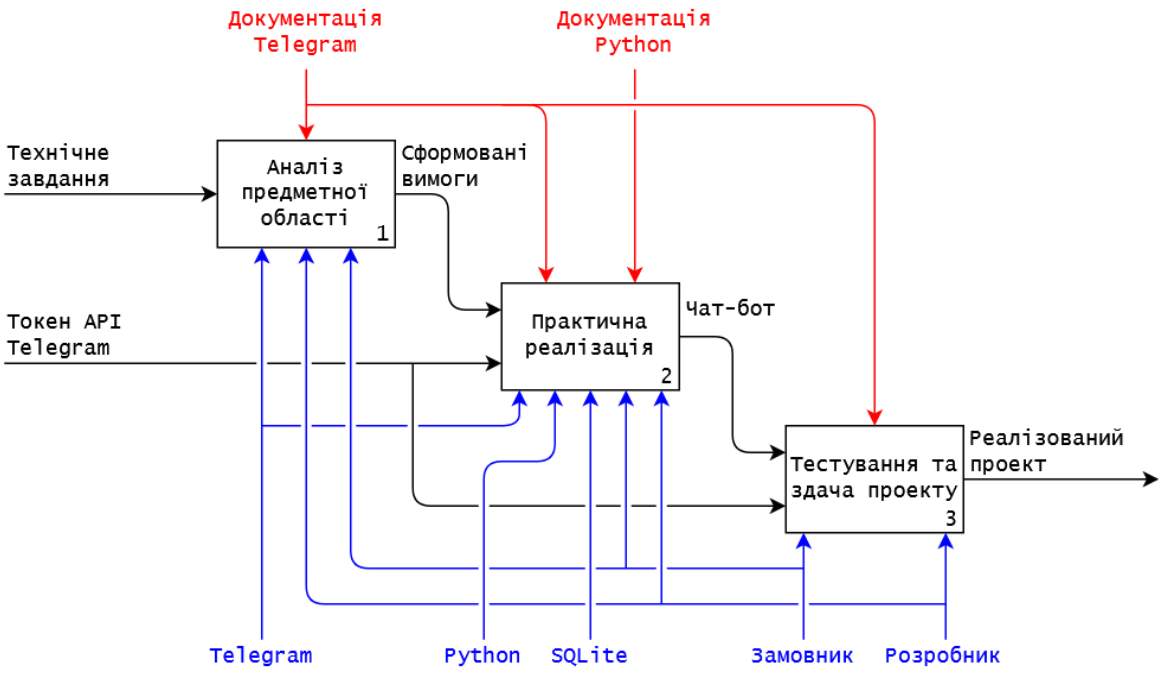

Рисунок 3.2 — Діаграма A0

Процес №1, «Аналіз предметної області», складається з пошуку аналогічних ботів, комплексного аналізу їх недоліків та переваг та формування остаточних вимог. Єдиним вхідним елементом субпроцесу є технічне завдання. Елементом контролю є документація Telegram. Елементами механізму є розробник та замовник, додаток Telegram. Вихідним елементом є сформовані вимоги. Діаграму декомпозиції процесу №1 зображено на рис. 3.3.

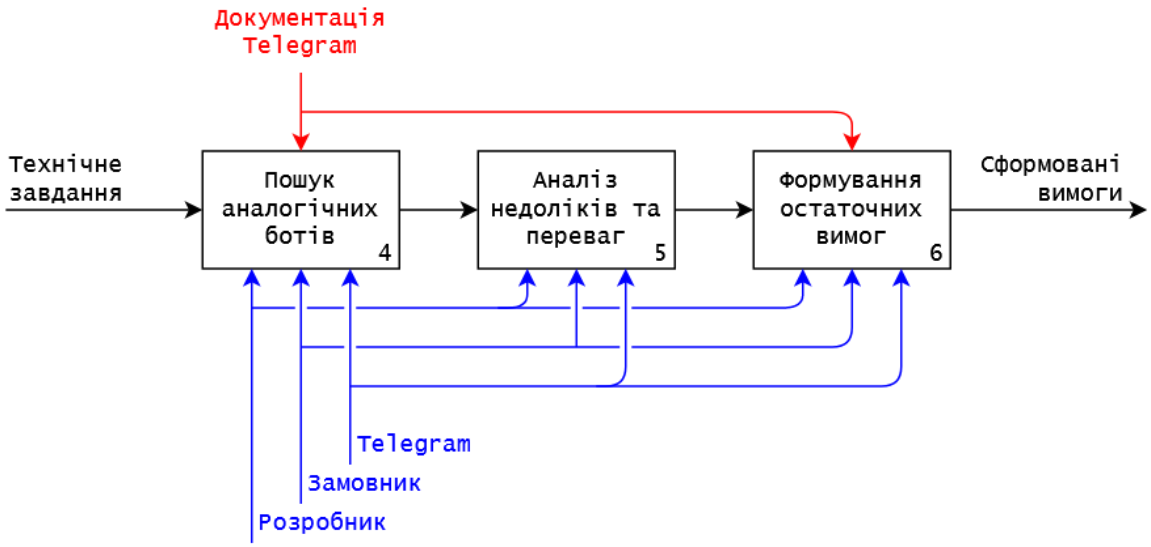

Рисунок 3.3 — Діаграма A1

Процес №2, «Практична реалізація», складається з розробки прототипу, структури бази даних та серверної частини. Вхідним елементом субпроцесу є сформовані у першому субпроцесі вимоги. Елементами контролю є документація Telegram та Python. Елементами механізму є розробник, Python, додаток Telegram та СКБД SQLite. Вихідним елементом є готовий чат-бот. Діаграму декомпозиції процесу №2 зображено на рис. 3.4.

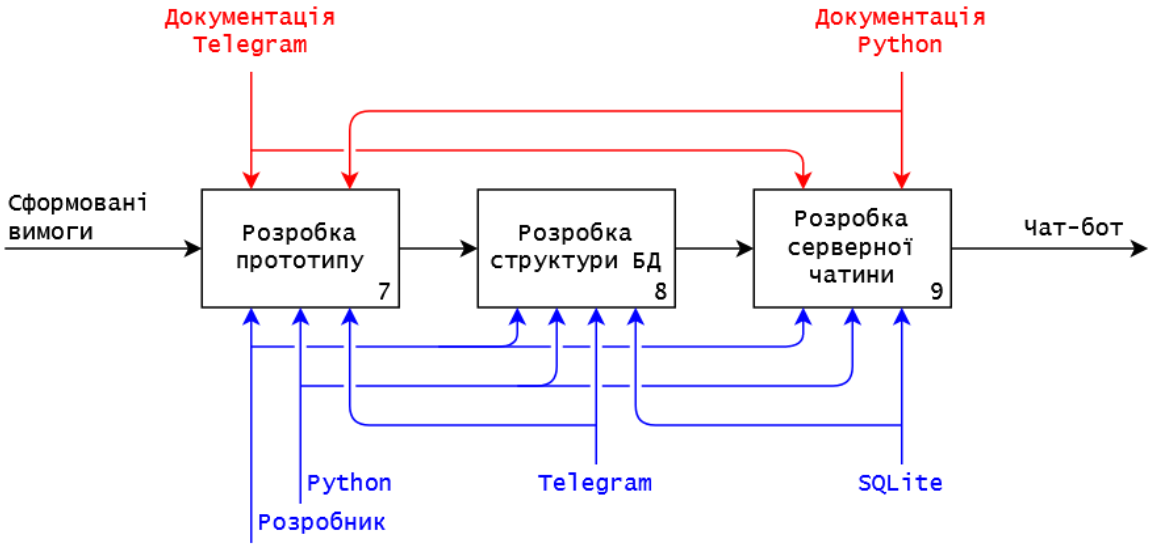

Рисунок 3.4 — Діаграма A2

Процес №3, «Тестування та здача проекту», складається з тестування чат-боту, розробки документації та здачі проекту в експлуатацію. Вхідним елементом субпроцесу є створенний у другому субпроцесі чат-бот. Елементами контролю є документація Telegram та документація Python. Елементами механізму є розробник, замовник, додаток Telegram, Python та СКБД SQLite. Вихідним елементом є реалізований проект. Діаграму декомпозиції процесу №3 зображено на рис. 3.5.

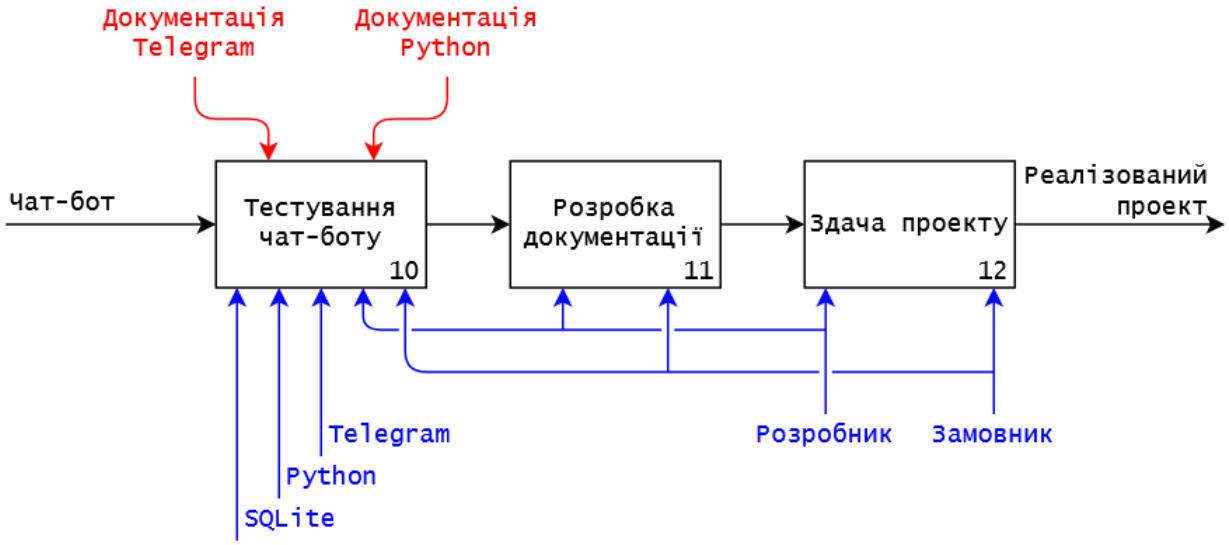

Рисунок 3.5 — Діаграма А3

3.2 Проектування моделі бази даних

Для опису структури інформації, що зберігається, буде використовуватися ER-модель (модель «сутність–зв'язок»).

Першим кроком створення ER-моделі інформаційної системи є побудова логічної (сутнісної) моделі, яка відображає логічне представлення даних за допомогою трьох основних компонентів: сутності, атрибуту та зв’язку [12]. Для створення логічної моделі спочатку треба визначити сутності та їх атрибути:

- сутність «Курс» (Course) має атрибути: ідентифікатор (course_id), назву (name), ціну (price), тривалість у годинах (hours), посилання на курс (url) та теги (tags);
- сутність «Завдання» (Homework) має атрибути: ідентифікатор (homework_id), курс, до якого належить це завдання (course), опис (description) та термін виконання (due);
- сутність «Студент» (Student) має атрибути: ідентифікатор (student_id), ім’я (first_name), прізвище (last_name), номер телефону (phone_number), ідентифікатор у додатку Telegram (telegram_id) та булевий флаг, який показує, чи бажає студент отримувати сповіщення (notify).

Наступним кроком є визначення зв’язків між визначеними сутностями та їх властивостями — силою, потужністю, типом участі сутності, ступеню зв’язку:

- зв’язок «Курс–Завдання» є сильним (завдання не може існувати без курсу), one-to-many (завдання може належати лише до одного курсу, курс може мати декілька завдань), optional–mandatory (курс може не мати завдань, завдання повинно належати якомусь курсу), бінарним;
- зв’язок «Курс–Студент» є слабким (існування курсів та студентів не пов’язане), many-to-many (студент може бути слухачем декількох курсів одночасно), optional–optional (студент може не бути слухачем жодного курсу, курс може не мати слухачів), бінарним.

Наступним кроком створення ER-моделі інформаційної системи є побудова фізичної (табличної) моделі, яка містить визначення усіх виділених у логічній моделі об’єктів у термінах кінцевої СКБД.

У даному випадку єдиною суттєвою зміною, яку треба внести для переходу від логічної моделі до фізичної, є заміна одного зв’язку типу many-to-many «Курс–Студент» двома зв’язками one-to-many за допомогою проміжної таблиці “student_course”.

Варто відмітити, що обраний метод зберігання тегів курсів, який не дозволяє привести таблицю «courses» навіть до першої нормальної форми, є запланованим. Основною операцією, пов’язаною з тегами, є об’єднання, і така форма зберігання дозволяє мінімізувати складність запитів без зменшення швидкодії [13].

Для візуалізації створеної фізичної ER-моделі створюється діаграма, яку можна побачити на рис. 3.6.

Атрибути таблиць та їх типи описані у табл. 3.1–3.4.

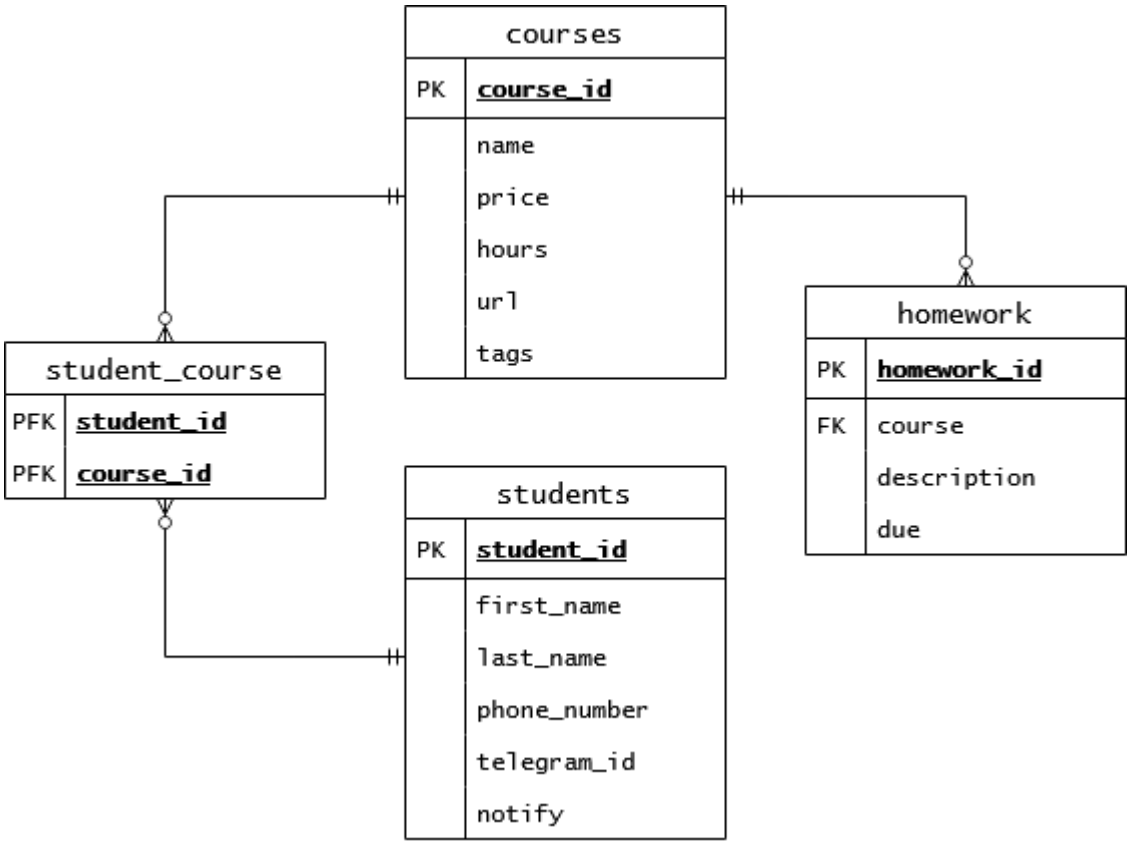

Рисунок 3.6 — Діаграма фізичної частини ER-моделі

Таблиця 3.1 — Структура таблиці «student_course»

Атрибут	Тип даних	Обмеження	Опис
student_id	INTEGER	PK, FK	Ідентифікатор студента
course_id	INTEGER	PK, FK	Ідентифікатор курсу

Таблиця 3.2 — Структура таблиці «courses»

Атрибут	Тип даних	Обмеження	Опис
course_id	INTEGER	PK	Ідентифікатор курсу
name	TEXT	NOT NULL, UNIQUE	Назва курсу
price	REAL	NOT NULL	Ціна проходження курсу
hours	INTEGER	NOT NULL	Тривалість курсу у годинах
url	TEXT	NOT NULL, UNIQUE	Посилання на сайт
tags	TEXT	NOT NULL	Теги, розділені комою

Таблиця 3.3 — Структура таблиці «students»

Атрибут	Тип даних	Обмеження	Опис
student_id	INTEGER	PK	Ідентифікатор студента
first_name	TEXT	NOT NULL	Ім'я студента
last_name	TEXT	NOT NULL	Прізвище студента
phone	TEXT	NOT NULL, UNIQUE	Номер телефону студента
telegram_id	TEXT		Ідентифікатор студента у Telegram
notify	INTEGER	DEFAULT 1	Флаг, який показує, чи бажає студент отримувати сповіщення

Таблиця 3.4 — Структура таблиці «homework»

Атрибут	Тип даних	Обмеження	Опис
homework_id	INTEGER	PK	Ідентифікатор завдання
course	INTEGER	FK, NOT NULL	Курс, до якого належить
description	TEXT		Опис завдання
due	TEXT	NOT NULL	Термін виконання (ISO 8601)

3.3 Моделювання варіантів використання

Для побудови діаграми варіантів використання був проведений аналіз можливих взаємодій користувача із системою. Це дозволило виділити трьох акторів, які можуть взаємодіяти з чат-ботом:

- неавторизований користувач (НК);
- авторизований користувач (АК);
- адміністратор (АД).

Був також складений список можливих операцій, які доступні кожному з акторів:

- авторизація (НК);
- запит інформації про НКНМЦ ІТП (НК, АК);
- пошук курсів за ключовими словами (НК, АК);
- отримання переліку курсів користувача (АК);
- отримання термінів виконання завдань (АК);
- автоматичне нагадування про терміни виконання завдань (АК);
- масова розсилка повідомлень (АК, АД);
- редагування записів БД (АД).

На основі отриманих результатів було створено діаграму варіантів використання, яку можна побачити на рис. 3.7.

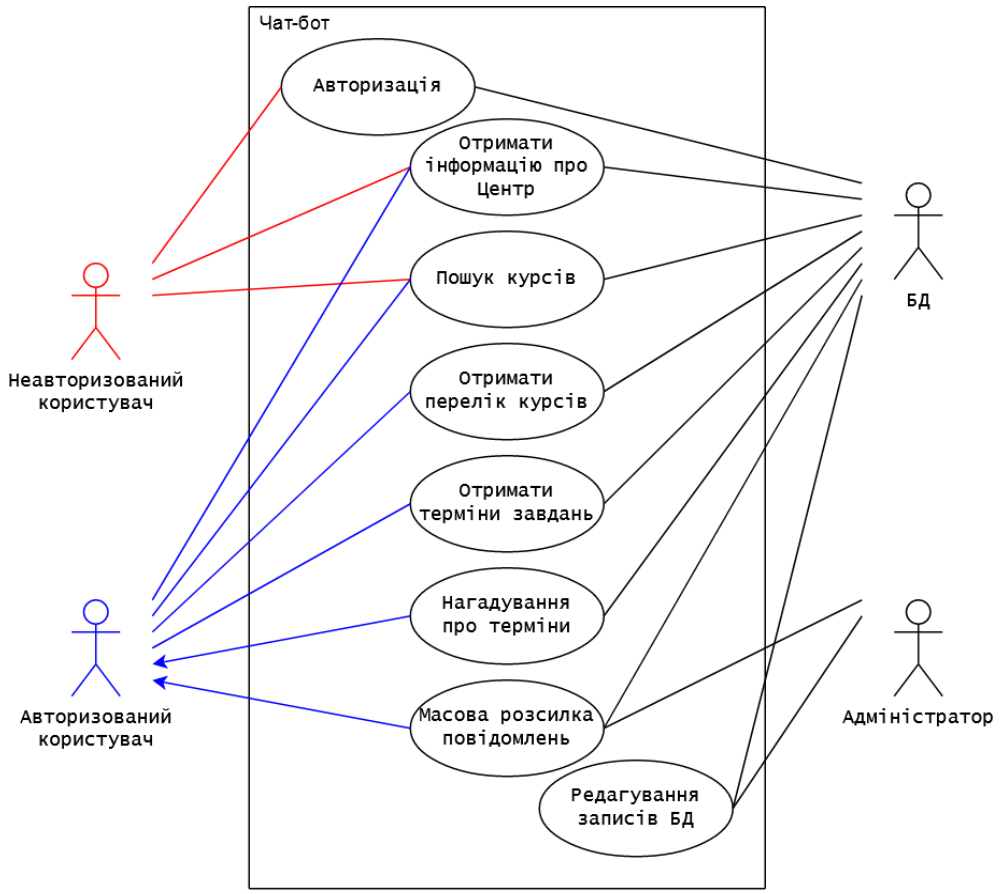

Рисунок 3.7 — Діаграма варіантів використання чат-боту

4 РОЗРОБКА ЧАТ-БОТУ

4.1 Розробка архітектури чат-боту

Для реалізації запланованого функціоналу чат-бот було вирішено розділити на три логічні частини:

– модуль роботи з базою даних, «BotDatabase», який впровадить шар абстракції між серверною частиною та базою даних;
– серверна частина, «BotServer», яка відповідатиме за взаємодію з користувачем;
– графічний інтерфейс серверної частини, «BotGUI», який буде використовуватися адміністратором для керування серверною частиною.

Графічне зображення архітектури чат-боту представлене на рис. 4.1.

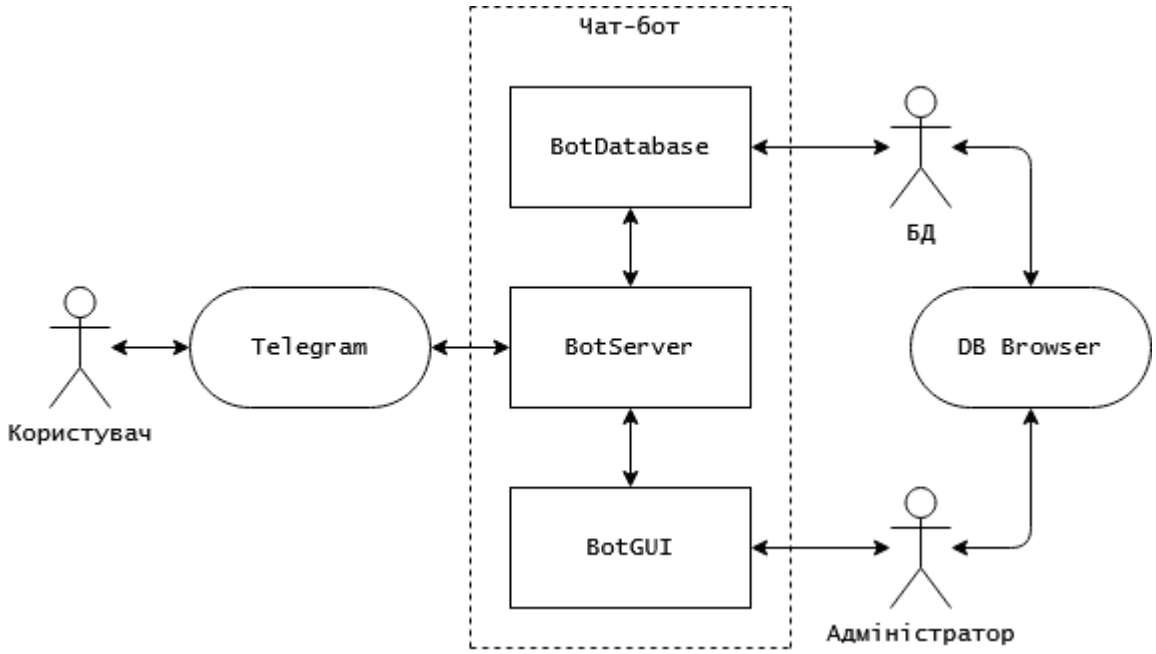

Рисунок 4.1 — Архітектура чат-боту

4.2 Розробка модулів чат-боту

Перед початком розробки та тестування чат-боту необхідно отримати авторизаційний токен для доступу до API. Зробити це можна звернувшись до бота BotFather з командою «/newbot» у самому додатку Telegram. Приклад цього показаний на рис. 4.2.

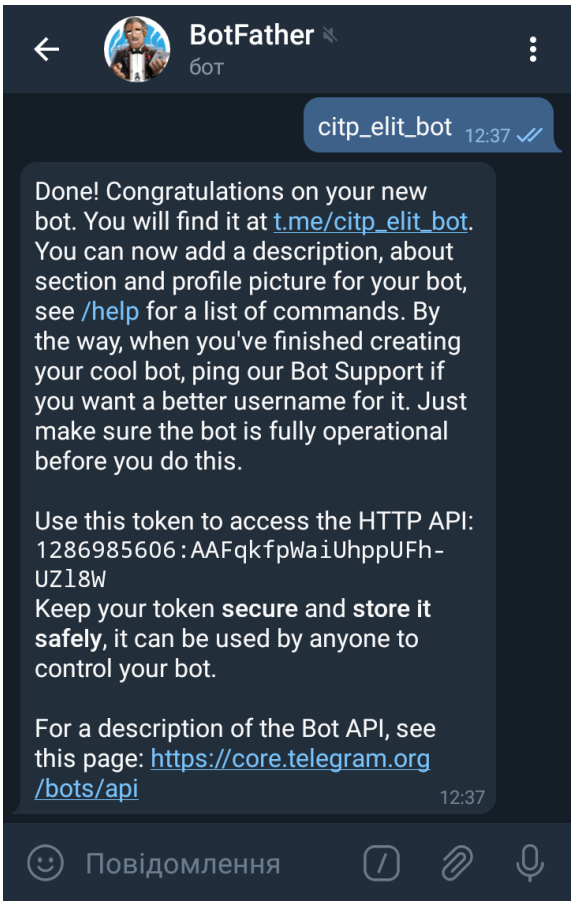

Рисунок 4.2 — Отримання токену Telegram API

Першим був розроблений клас «BotDatabase», розміщений у файлі «database.py». Основною особливістю, яку потрібно враховувати, є проблематичність використання одного з’єднання з базою даних багатьма потоками

одночасно [14]. Для вирішення цієї проблеми достатньо створювати підключення окремо для кожного потоку. Методи класу «BotDatabase» представлені у табл. 4.1.

Таблиця 4.1 — Методи класу «BotDatabase»

Метод	**Опис**
load_strings(language)	Завантажує локалізовані строки для подальшого використання
is_user_registered(telegram_id)	Перевіряє, чи є користувач з заданим ID у системі
is_phone_in_database(phone_number)	Перевіряє, чи є користувач з заданим номером телефону у системі
update_telegram_id(phone_number, telegram_id)	Оновлює значення ID для користувача із заданим номером телефону
search_courses_by_tags(tags)	Здійснює пошук курсів за заданим набором тегів
find_active_courses(telegram_id)	Повертає перелік курсів користувача
find_pending_homework(telegram_id)	Повертає перелік завдань курсів користувача з заданим ID, які очікують виконання
has_notify_flag(telegram_id)	Перевіряє, чи встановлений флаг notify у користувача з заданим ID
set_notify_flag(telegram_id, value)	Встановлює чи очищує флаг notify у користувача з заданим ID
get_notifees()	Повертає перелік користувачів зі встановленим флагом notify

Наступним був розроблений клас «BotServer», розміщений у файлі «server.py». Основним питанням є вибір методу отримання повідомлень від користувачів. Існує два можливих варіанти:

– опитування (polling), при якому сервер чат-боту періодично посилає на сервери Telegram запит та завантажує отримані повідомлення;
– вебхук (webhook), при якому сервери Telegram автоматично пересилають повідомлення на заздалегідь указану адресу за допомогою HTTPS.

Діаграми послідовності для обох механізмів показані на рис. 4.3–4.4.

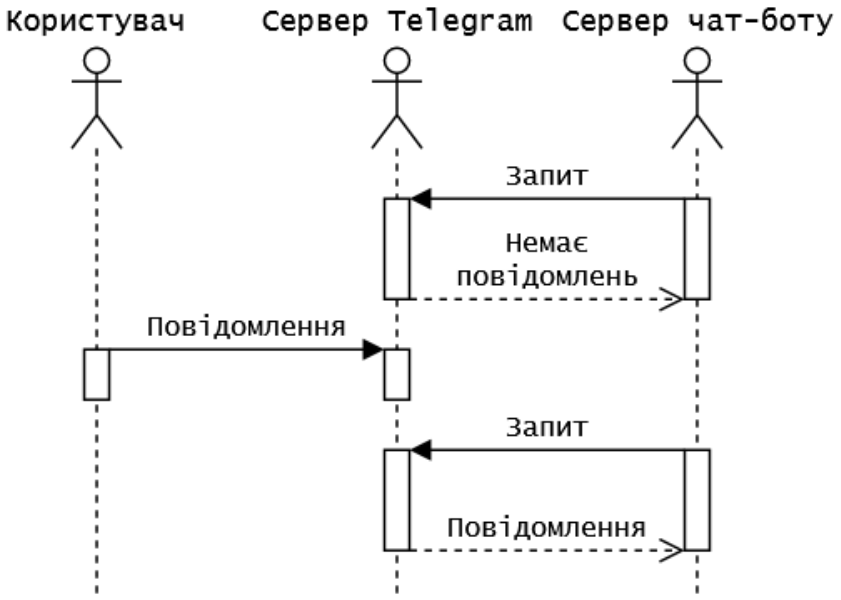

Рисунок 4.3 — Діаграма послідовності для механізму опитування

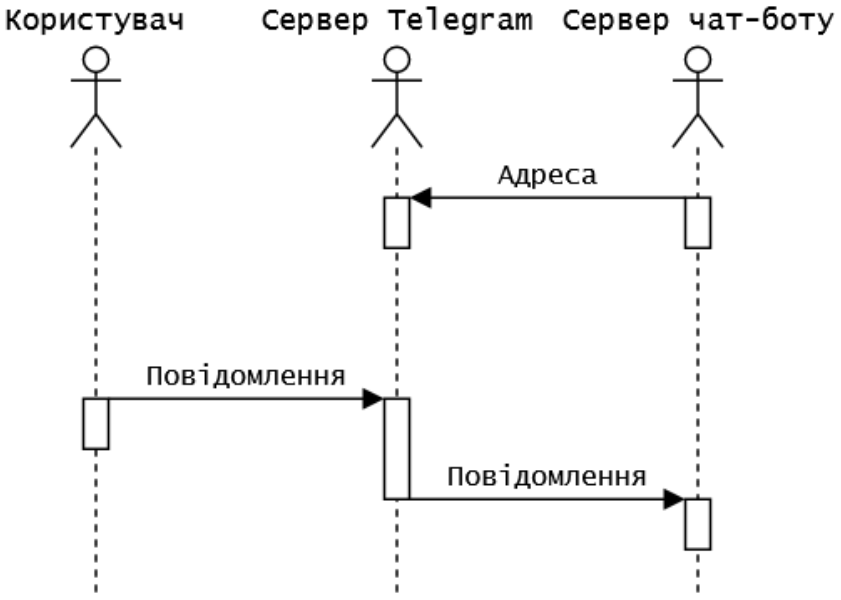

Рисунок 4.4 — Діаграма послідовності для механізму вебхуку

На перший погляд механізм вебхуку може здатися найкращім вибором, але треба врахувати, що він потребує наявності не тільки публічної ір-адреси, на яку будуть передаватися повідомлення, а й дійсного SSL-сертифіката [15]. Для того, щоб зменшити кількість вимог до цільового обладнання, був обраний другий метод, опитування. Це також дозволить розміщувати сервер чат-боту у мережах, які використовують механізм трансляції адрес (NAT), без будь-яких ускладнень системи [16].

Методи класу «BotServer» представлені у табл. 4.2.

Таблиця 4.2 — Методи класу «BotServer»

Метод	**Опис**
l10n(string_id)	Повертає локалізовану строку з заданим ID
generate_keyboards()	Генерує клавіатури для зареєстрованих та незареєстрованих користувачей
load_handlers()	Генерує хендлери діалогу
get_keyboard(user_id)	Повертає клавіатуру, яку повинен бачити користувач із заданим ID
get_return_state(user_id)	Повертає стан, у який повинен потрапити користувач із заданим ID
handle_greeting(update, context)	Обробляє діалог «Привітання»
handle_about(update, context)	Обробляє діалог «Про Центр»
handle_auth(update, context)	Обробляє діалог «Авторизація»
handle_search(update, context)	Обробляє діалог «Пошук курсів»
handle_search_query(update, context)	Обробляє отримані у діалозі «Пошук курсів» дані
handle_search_cancel(update, context)	Обробляє скасування діалогу «Пошук курсів»

Продовження таблиці 4.2 — Методи класу «BotServer»

build_course_list(courses)	Форматує заданий перелік курсів
handle_courses(update, context)	Обробляє діалог «Курси»
build_homework_list(homework)	Форматує заданий перелік завдань
handle_pending(update, context)	Обробляє діалог «Завдання»
handle_notifications(update, context)	Обробляє діалог «Керування нагадуваннями»
handle_unknown(update, context)	Обробляє невірні команди
handle_error(update, context)	Обробляє помилки
send_notifications(context)	Розсилає нагадування
enable_notifications(utc_time)	Вмикає нагадування у заданий час
disable_notifications()	Вимикає нагадування
mass_send(message)	Розсилає повідомлення
start()	Вмикає сервер
stop()	Вимикає сервер

Наступним кроком є створення графічного інтерфейсу для більш зручного керування серверною частиною чат-бота. Для цього була обрана бібліотека Tkinter, яка представляє інтерфейс до інструментарію створення графічних інтерфейсів Tk у мові програмування Python [17]. Був розроблений клас «BotGUI», який розміщений у файлі «gui.py».

Після запуску програми, користувач бачить перед собою головне вікно із трьома вкладками. Перша вкладка, «Стан роботи», яка відкрита за замовчуванням, керує статусом сервера чат-боту. Вона складається з двох частин:

– журнал, у який автоматично заносяться події, пов’язані з роботою чат-бота, та їх відмітка про час;
– кнопки керування статусом, за допомогою яких запускається та зупиняється сервер.

Вкладка «Стан роботи» зображена на рис. 4.5.

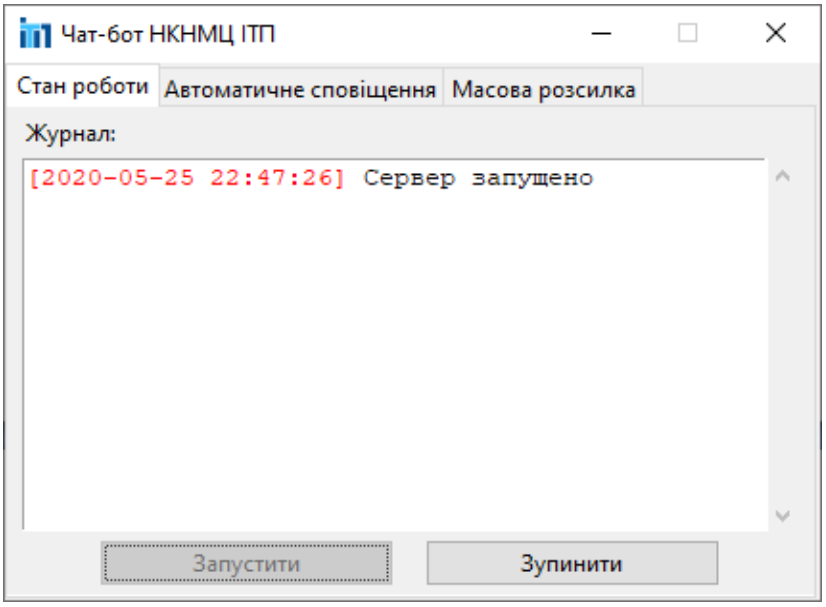

Рисунок 4.5 — Вигляд вкладки «Стан роботи»

Друга вкладка, «Автоматичне сповіщення», яка доступна лише якщо сервер запущений, відповідає за нагадування користувачам про терміни виконання завдань. Вона складається з чек боксу, який вмикає та вимикає цю функцію. Якщо він установлений, з’являється доступ до елементів контролю часу нагадування, які розташовані нижче. Вони складаються з двох спінерів, для годин та хвилин відповідно, та кнопки, яка завершує процес вибору часу.

Вкладка «Автоматичне сповіщення» зображена на рис. 4.6.

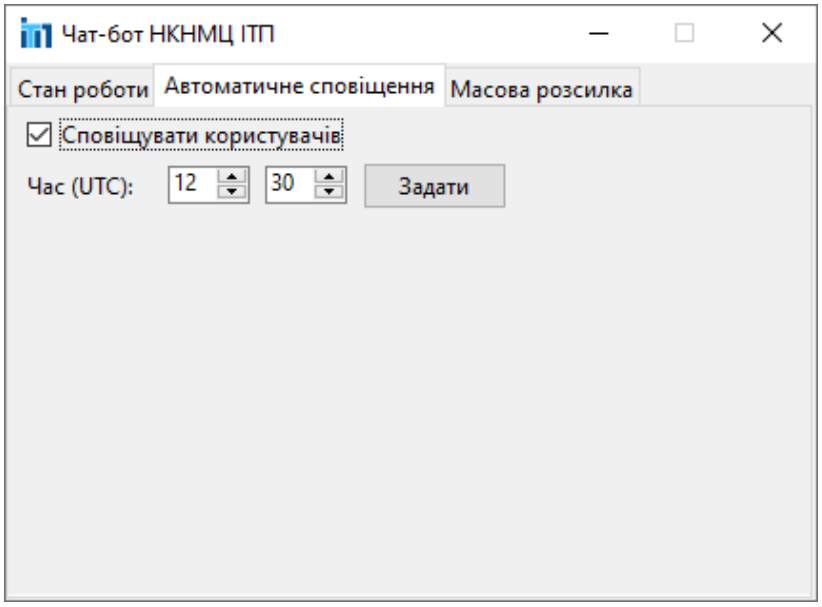

Рисунок 4.6 — Вигляд вкладки «Автоматичне сповіщення»

Третя вкладка, «Масова розсилка», яка також доступна лише якщо сервер запущений, надає можливість адміністратору розсилати користувачам новини та важливі повідомлення. Вона складається з текстового поля для вводу повідомлення та кнопки для його відправки.

Вкладка «Масова розсилка» зображена на рис. 4.7.

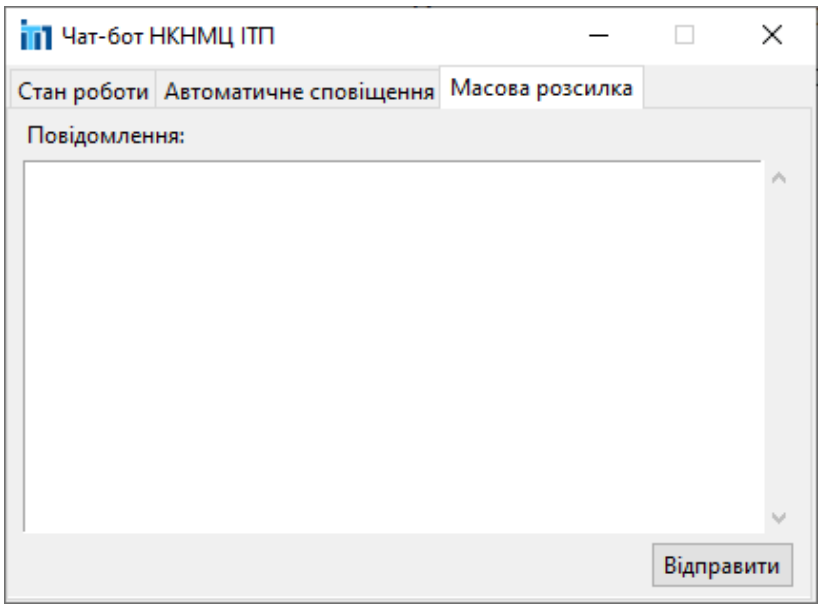

Рисунок 4.7 — Вигляд вкладки «Масова розсилка»

Методи класу «BotGUI» представлені у табл. 4.3.

Таблиця 4.3 — Методи класу «BotGUI»

Метод	Опис
update_log(message)	Додає задане повідомлення у журнал
receive_error(update, context)	Отримує помилку від класу BotServer
pressed_start()	Обробляє натискання кнопки «Запустити»
pressed_stop()	Обробляє натискання кнопки «Зупинити»
changed_notify()	Обробляє зміну флагу повідомлень
changed_time(value, widget)	Обробляє зміну даних у спінерах зміни часу
get_time()	Повертає введений час у вигляді строки
pressed_update()	Обробляє натискання кнопки «Задати»
pressed_bulk()	Обробляє натискання кнопки «Відправити»
on_close()	Обробляє подію закриття вікна

Для об’єднання трьох описаних класів у єдину систему та надання точки входу у програму був створений файл «bot.py».

4.3 Реалізація моделі бази даних

Обрана СКБД має декілька особливостей, які потрібно враховувати при створюванні таблиць. Наприклад, SQLite підтримує не весь стандарт мови SQL [18]. Варто також зазначити, що у SQLite типів даних лише 5 [19]:

- NULL, для представлення відповідного значення;
- INTEGER, цілочисельний тип з автоматичним підбором розрядності від 1 до 8 байтів;

– REAL, тип з плаваючою комою з розрядністю 8 байтів;

– TEXT, строковий тип з підтримкою UTF-8 та UTF-16;

– BLOB, бінарний тип для представлення довільних даних.

Структура створених таблиць у DB Browser зображена на рис. 4.8–4.11.

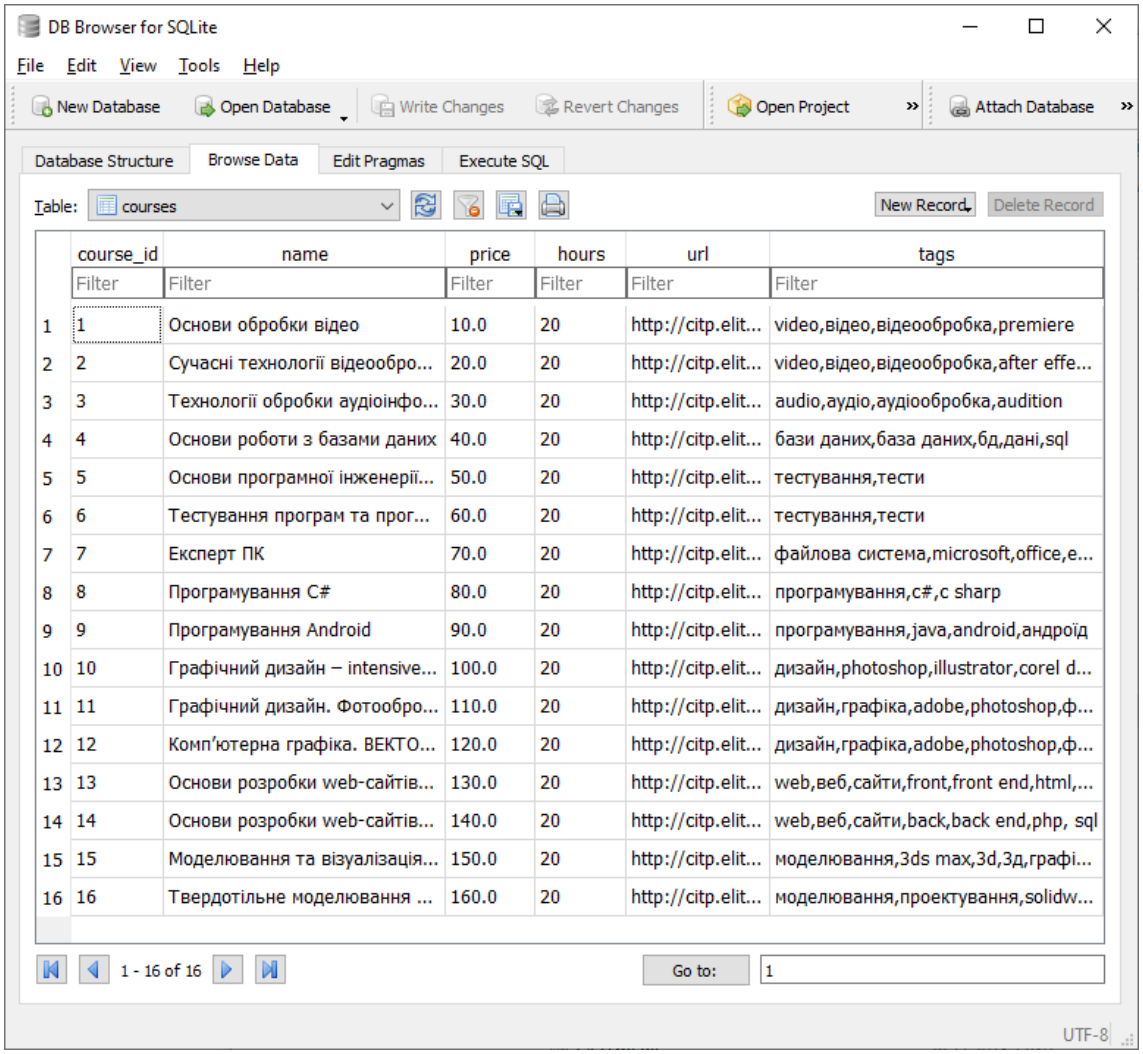

Рисунок 4.8 — Таблиця «courses»

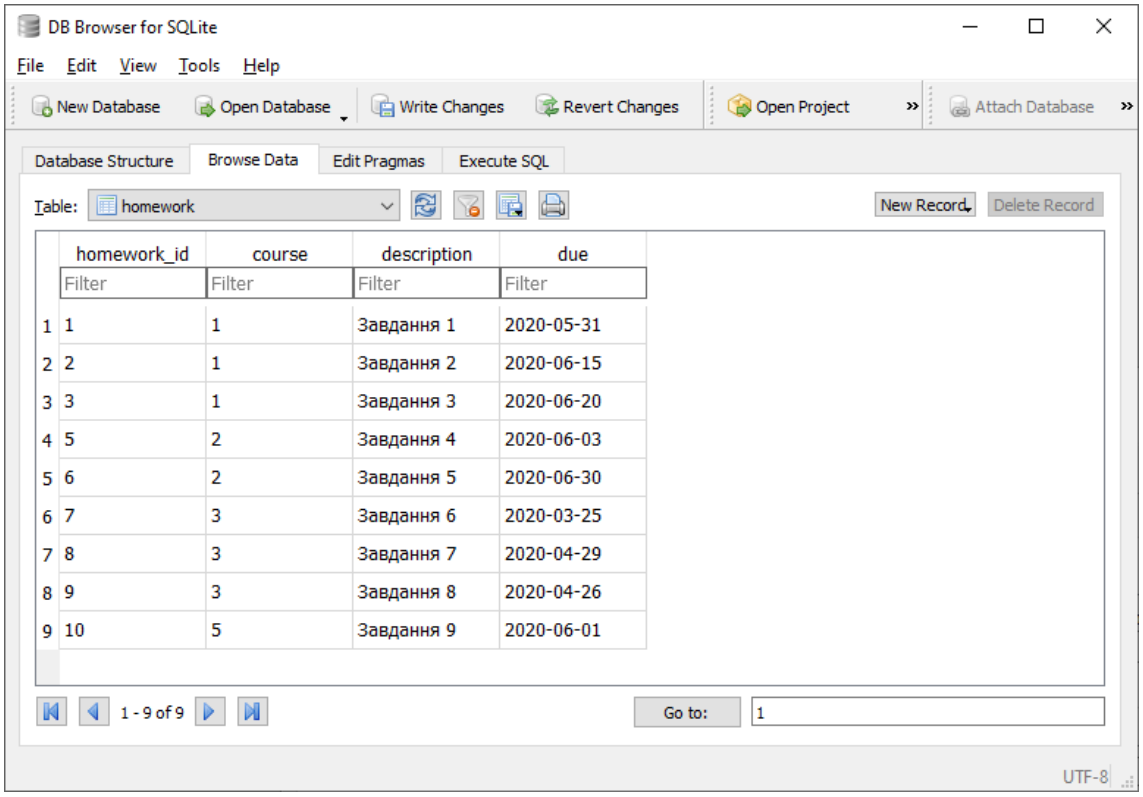

Рисунок 4.9 — Таблиця «homework»

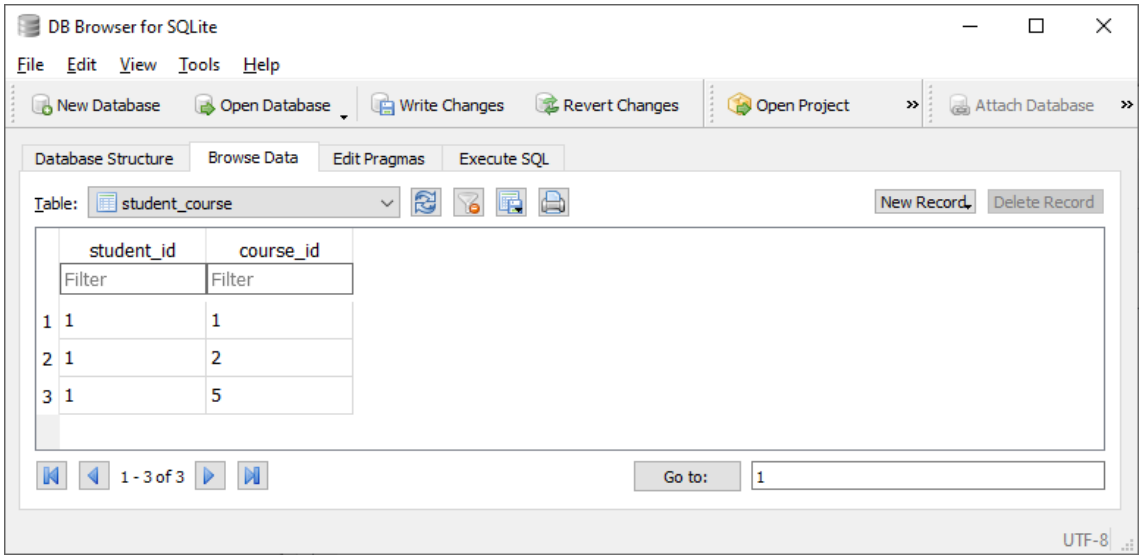

Рисунок 4.10 — Таблиця «student_course»

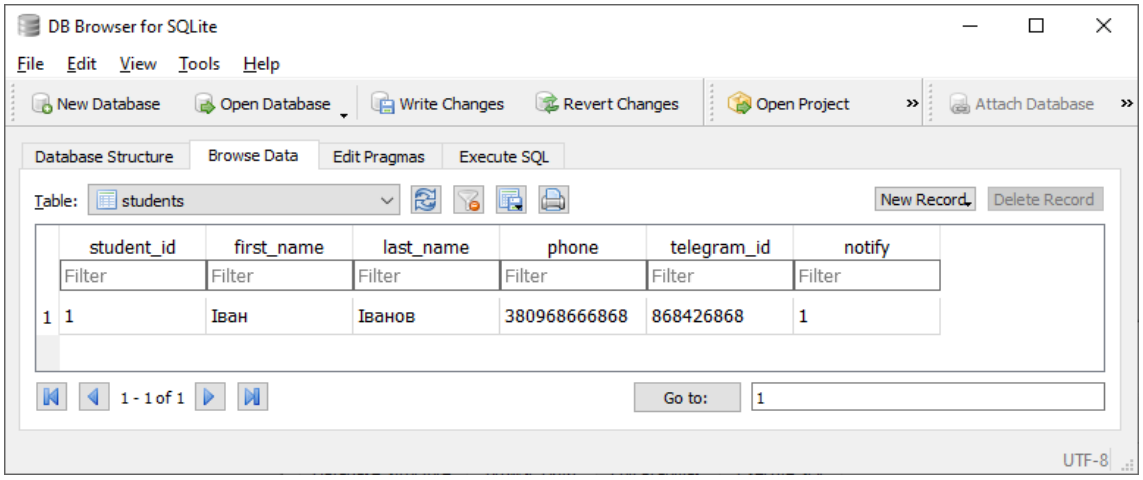

Рисунок 4.11 — Таблиця «students»

4.4 Створення дистрибутиву

До цього етапу розроблена програма існувала у вигляді шести файлів:

– bot.py;

– database.py;

– server.py;

– gui.py;

– database.db;

– itp.ico.

Для запуску і коректної роботи програми необхідно, щоб на цільовому комп'ютері був встановлений інтерпретатор Python з дев'ятьма модулями, створеними сторонніми розробниками:

– python-telegram-bot — для спрощення роботи з API Telegram;

– future — для сумісності між Python 2 та 3;

– tornado — для роботи з мережею;

– decorator — для полегшення роботи з декораторами;

– certifi — для роботи з TLS;

– cryptography — для роботи з криптографією;

– six — для сумісності між Python 2 та 3;

– cffi — для полегшення інтеграції C та Python;

– pycparser — для парсингу C.

Це є досить незручним. Для створення дистрибутиву, який полегшить процес розгортання серверної частини чат-бота на цільовому комп'ютері, була використана утиліта PyInstaller, яка дозволяє упакувати інтерпретатор Python, необхідні модулі та файли самої програми у єдиний виконуваний файл [20]. Проведена операція дозволила зменшити дистрибутив до трьох необхідних файлів:

– bot.exe;

– database.db;

– itp.ico.

Робота утіліти PyInstaller зображена на рис. 4.12.

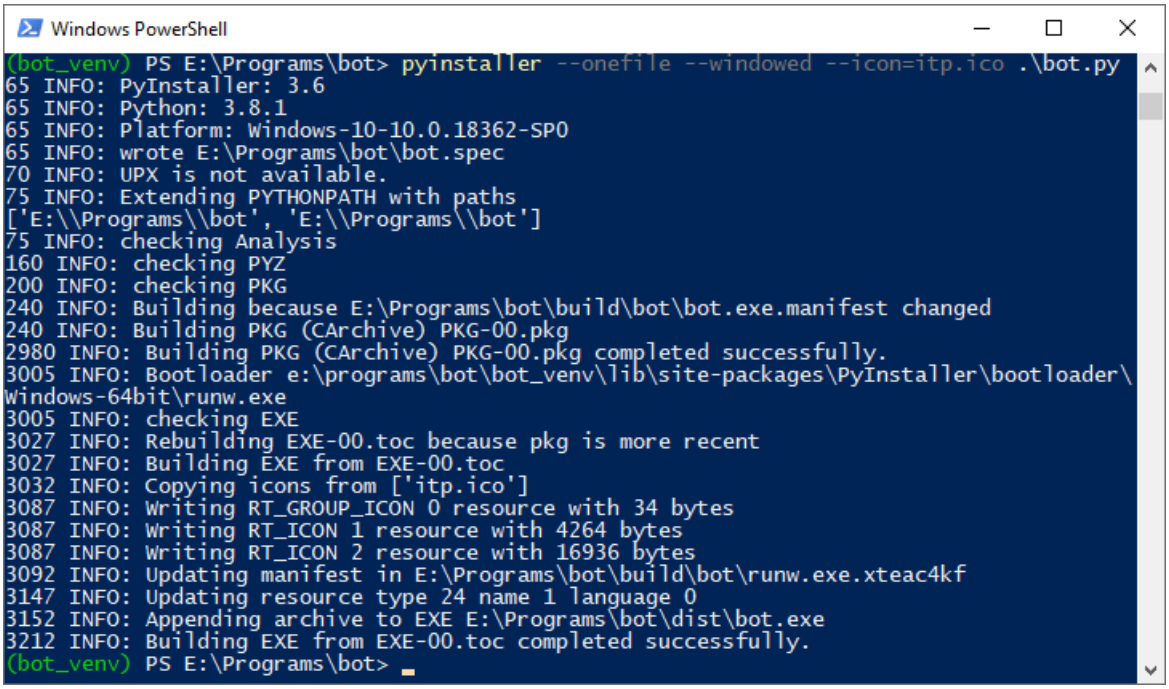

Рисунок 4.12 — Робота утіліти PyInstaller

4.5 Інтеграція чат-боту у Telegram

Завершення розробки дає можливість переходити до наступного кроку, інтеграції боту в Telegram. Для цього спочатку потрібно зробити декілька кроків за допомогою BotFather:

– призначити боту опис за допомогою команди «/setdescription»;
– призначити боту зображення користувача за допомогою команди «/setuserpic»;
– призначити боту перелік доступних команд за допомогою команди «/setcommands».

Цей процес можна побачити на рис. 4.13–4.15.

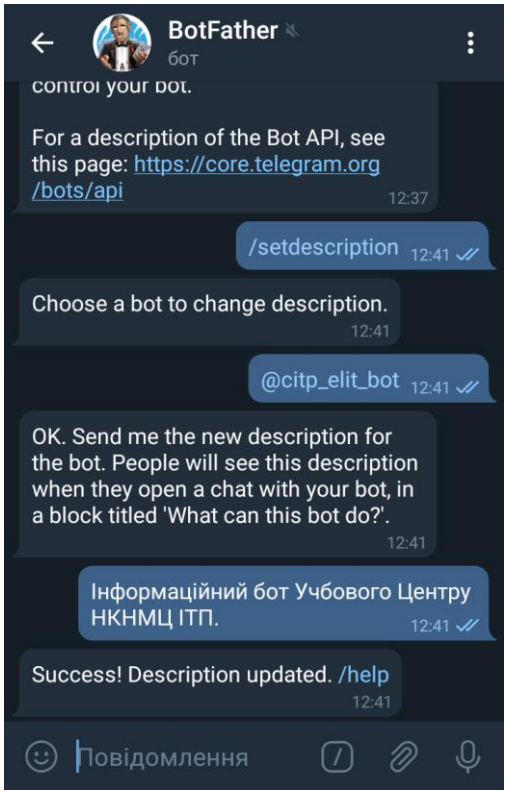

Рисунок 4.13 — Призначення опису

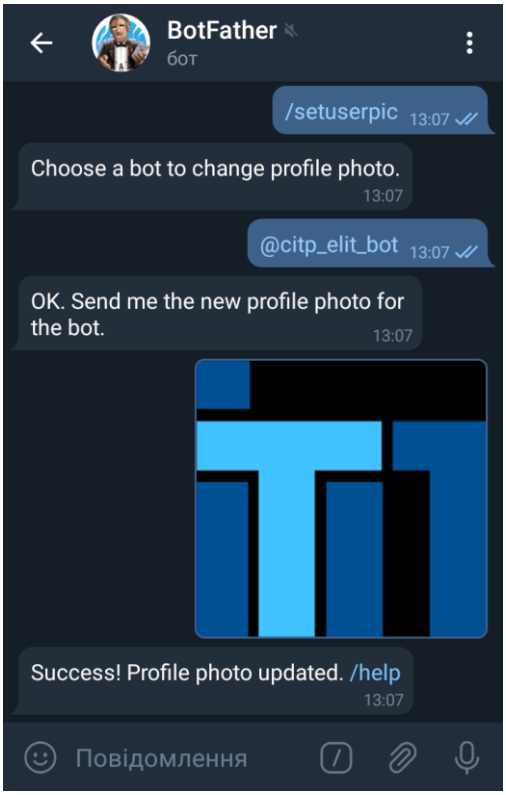

Рисунок 4.14 — Призначення зображення користувача

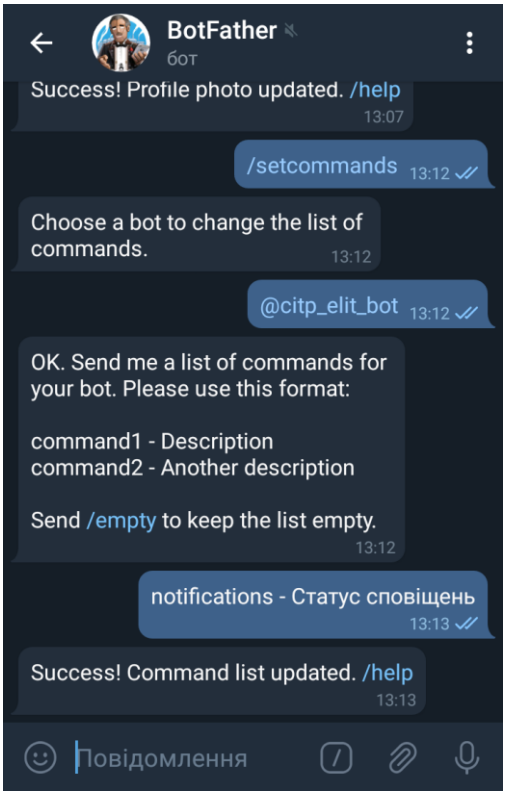

Рисунок 4.15 — Призначення переліку доступних команд

Після цього можна перевірити працездатність створеної системи та переходити до подальшого інтеграційного тестування та створення інструкції користування.

Перевірка працездатності зображена на рис. 4.16–4.17.

Повна інструкція використання бота із прикладами приведена у Додатку В.

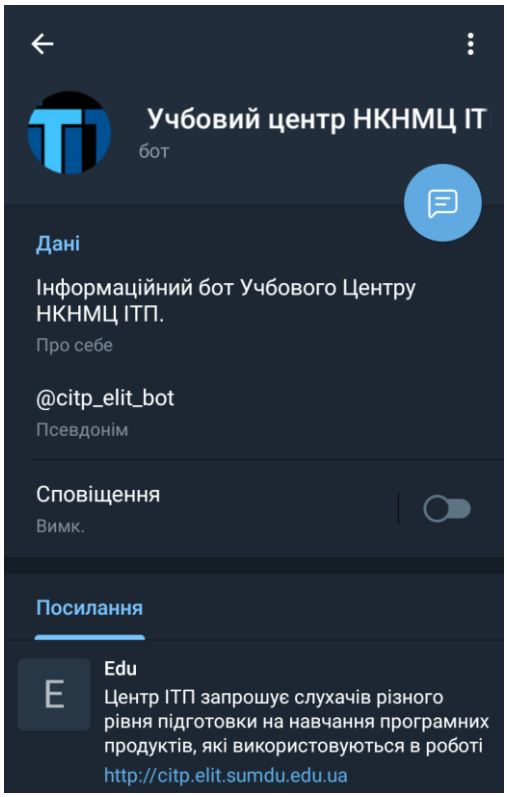

Рисунок 4.16 — Доданий у Telegram бот

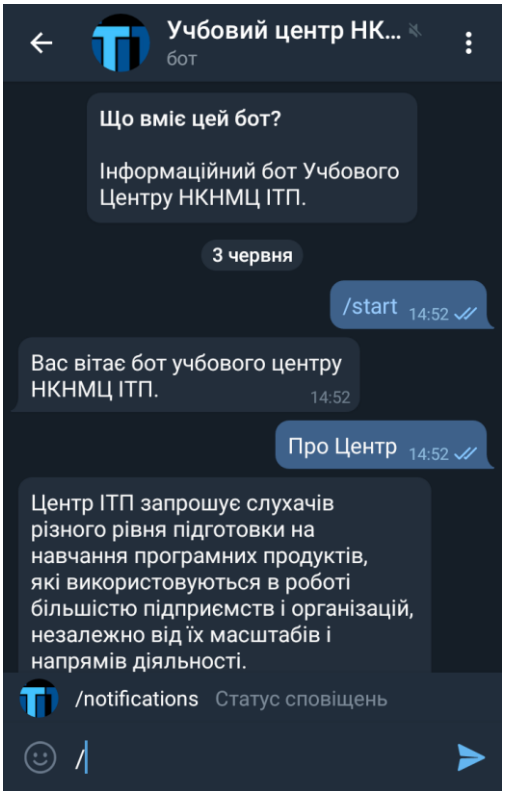

Рисунок 4.17 — Перевірка працездатності боту

ВИСНОВКИ

У ході виконання кваліфікаційної роботи бакалавра було проаналізовано різноманітні сторони побудови інформаційних систем у екосистемі додатку Telegram. Це дозволило виявити недоліки та переваги сторони вже існуючих інформаційних систем такого типу та виявити критичні аспекти побудови зручних інтерфейсів користувача.

На основі аналізу була сформовані мета роботи та обрані технології, які дозволили ефективно вирішити поставлену задачу. Це дозволило створити докладний план робіт та скласти структуру проекту, побудувати WBS та OBS системи та провести первинний аналіз ризиків.

Після цього було проведене моделювання чат-боту, була створена функціональна модель системи, проаналізовані її потреби у зберіганні інформації за допомогою ER-моделі. Була також створена повна діаграма використання чат-боту. Це стало основою процесу розробки та тестування.

Виконання усіх етапів кваліфікаційної роботи дозволило отримати чат-бот, який має наступні функції:

- видача інформації про НКНМЦ ІТП;
- пошук курсів за ключовими словами;
- видача інформації про наступні завдання і строки їх виконання у курсах за прямим запитом користувача;
- автоматичне нагадування про строки виконання завдань;
- масова розсилка повідомлень користувачам.

Чат-бот відповідає заявленим функціональним вимогам і є готовим до впровадження і використання студентами курсів НКНМЦ ІТП та людьми у них зацікавленими.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Global Academic E-Learning Market 2019-2023 [Електронний ресурс]. – Режим доступу: https://www.technavio.com/report/global-academic-e-learning-market-industry-analysis%20&utm_source=pressrelease&utm_medium=bw&utm_campaign=t25_wk39_top&utm_term=RFS&utm_content=IRTNTR31012 – 01.04.2019р. – Назва з титулу екрана.
2. Ranked: The World’s Most Downloaded Apps [Електронний ресурс]. – Режим доступу: https://www.visualcapitalist.com/ranked-most-downloaded-apps/ – 25.01.2020р. – Назва з титулу екрана.
3. ІТП НКНМЦ [Електронний ресурс]. – Режим доступу: http://citp.elit.sumdu.edu.ua/ – 01.02.2020р. – Назва з титулу екрана.
4. Seven Years Into The Mobile Revolution: Content is King… Again [Електронний ресурс]. – Режим доступу: https://www.flurry.com/post/127638842745/seven-years-into-the-mobile-revolution-content-is – 26.08.2015р. – Назва з титулу екрана.
5. Modernization: Clearing A Pathway To Success [Електронний ресурс]. – Режим доступу: https://www.standishgroup.com/sample_research_files/Modernization.pdf – 01.01.2010р. – Назва з титулу екрана.
6. Number of monthly active Telegram users worldwide from March 2014 to March 2018 [Електронний ресурс]. – Режим доступу: https://www.statista.com/statistics/234038/telegram-messenger-mau-users/ – 01.03.2018р. – Назва з титулу екрана.
7. Steve Worswick Interview - Loebner 2013 winner [Електронний ресурс]. – Режим доступу:

https://aidreams.co.uk/forum/index.php?page=Steve_Worswick_Interview_-_Loebner_2013_winner – 23.09.2013р. – Назва з титулу екрана.

8. The Society for the study of Artificial Intelligence and Simulation of Behaviour [Електронний ресурс]. – Режим доступу: https://aisb.org.uk/aisb-events/ – 15.02.2020р. – Назва з титулу екрана.
9. GitHub - python-telegram-bot/python-telegram-bot: We have made you a wrapper you can't refuse [Електронний ресурс]. – Режим доступу: https://github.com/python-telegram-bot/python-telegram-bot – 06.02.2020р. – Назва з титулу екрана.
10. Most Widely Deployed and Used Database Engine [Електронний ресурс]. – Режим доступу: https://sqlite.org/mostdeployed.html – 10.02.2020р. – Назва з титулу екрана.
11. Systems Engineering Fundamentals — Defense Acquisition University Press, 2001. 222 с.
12. What is Entity Relationship Diagram (ERD)? [Електронний ресурс]. – Режим доступу: https://www.visual-paradigm.com/guide/data-modeling/what-is-entity-relationship-diagram/ – 22.01.2019р. – Назва з титулу екрана.
13. Tagsystems: performance tests | Random Howtos [Електронний ресурс]. – Режим доступу: http://howto.philippkeller.com/2005/06/19/Tagsystems-performance-tests/ – 19.06.2005р. – Назва з титулу екрана.
14. sqlite3 — DB-API 2.0 interface for SQLite databases — Python 3.8.3 documentation [Електронний ресурс]. – Режим доступу: https://docs.python.org/3.8/library/sqlite3.html#multithreading – 20.05.2020р. – Назва з титулу екрана.
15. Telegram Bot API [Електронний ресурс]. – Режим доступу: https://core.telegram.org/bots/api#making-requests – 24.04.2020р. – Назва з титулу екрана.

16. Webhooks... Great when you can get them [Електронний ресурс]. – Режим доступу: https://blog.alexellis.io/webhooks-are-great-when-you-can-get-them/ – 26.09.2019р. – Назва з титулу екрана.
17. tkinter — Python interface to Tcl/Tk — Python 3.8.3 documentation [Електронний ресурс]. – Режим доступу: https://docs.python.org/3/library/tkinter.html – 26.05.2020р. – Назва з титулу екрана.
18. Query Language Understood by SQLite [Електронний ресурс]. – Режим доступу: https://www.sqlite.org/lang.html – 25.04.2020р. – Назва з титулу екрана.
19. Datatypes In SQLite Version 3 [Електронний ресурс]. – Режим доступу: https://www.sqlite.org/datatype3.html – 25.04.2020р. – Назва з титулу екрана.
20. PyInstaller Quickstart — PyInstaller bundles Python applications [Електронний ресурс]. – Режим доступу: https://www.pyinstaller.org/ – 14.02.2020р. – Назва з титулу екрана.

ДОДАТОК А
ТЕХНІЧНЕ ЗАВДАННЯ

1. Призначення й мета створення інформаційної системи

1.1 Призначення інформаційної системи

Чат-бот для месенджеру Telegram повинен представляти доступ до інформації про курси та завдання до них НКНМЦ ІТП.

1.2 Мета створення інформаційної системи

Забезпечення оперативного доступу до інформації о курсах НКНМЦ ІТП.

1.3 Мета створення інформаційної системи

У цільовій аудиторії чат-боту можна виділити наступні групи:

- студенти курсів НКНМЦ ІТП;
- люди, зацікавлені курсами НКНМЦ ІТП.

2. Вимоги до інформаційної системи в цілому

2.1 Вимоги до структури й функціонування чат-боту

Інформаційна система повинна бути реалізована у вигляді чат-боту, доступного в додатку Telegram.

2.2 Вимоги до персоналу

Для підтримки додатку персонал повинен мати базові навички роботи з базами даних, загальні навички роботи з персональним комп'ютером.

2.3 Вимоги до збереження інформації

У системі керування чат-ботом повинен бути передбачений механізм резервного копіювання структури й вмісту бази даних. Процедура резервного копіювання повинна проводитися співробітником, відповідальним за підтримку додатку.

2.4 Вимоги до розмежування доступу

База даних не повинна містити конфіденційної інформації. База даних може містити приватну інформацію користувача, а саме ПІБ, номер телефону та ID у додатку Telegram.

Користувачів чат-боту можна розділити на 3 групи відповідно до прав доступу:

- відвідувачі;
- студенти;
- адміністратор.

Відвідувачі мають доступ тільки до інформації про НКНМЦ ІТП та функції пошуку, базової інформації про курси.

Студенти також мають доступ до інформації про курси та статус завдань.

Адміністратор має повний локальний доступ до бази даних та може додавати та видаляти користувачів, курси та завдання.

Розділення відвідувачів та студентів відбувається за допомогою ID користувача у додатку Telegram, а також номеру телефону користувача, отриманого за допомогою вбудованої в Telegram функції відправки номеру телефону.

3. Основні вимоги

3.1 Функціональні вимоги

Чат-бот має забезпечувати виконання наступних функцій:

– пошук курсів за ключовими словами, такими як “аудіо обробка”, “3d моделювання”;
– видача інформації про наступні завдання у курсах за прямим запитом користувача, наприклад, за допомогою команди “/pending”;
– автоматичне нагадування про строки виконання завдань.

3.2 Вимоги до програмного забезпечення

Програмне забезпечення клієнтської частини повинне задовольняти наступним вимогам:

– додаток Telegram.

Програмне забезпечення клієнтської частини повинне задовольняти наступним вимогам:

– Python 3.8.1.

4. Макети інтерфейсу

У цьому розділі представлено декілька макетів користувацького інтерфейсу.

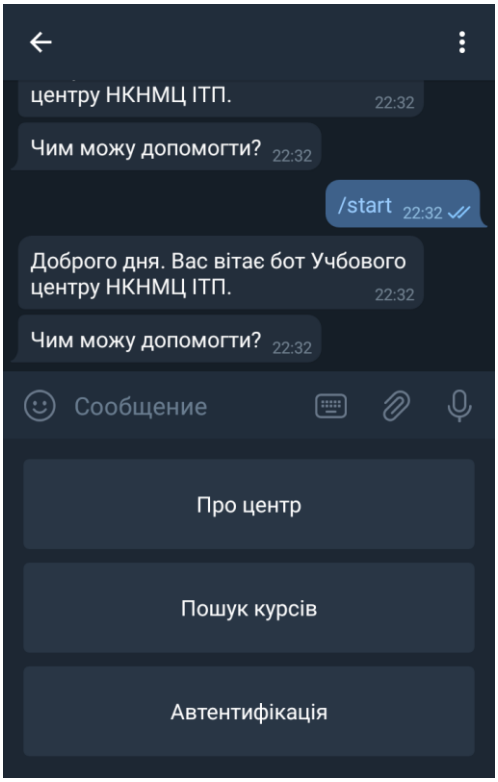

Рисунок А.1 – Макет інтерфейсу для неавторизованого користувача

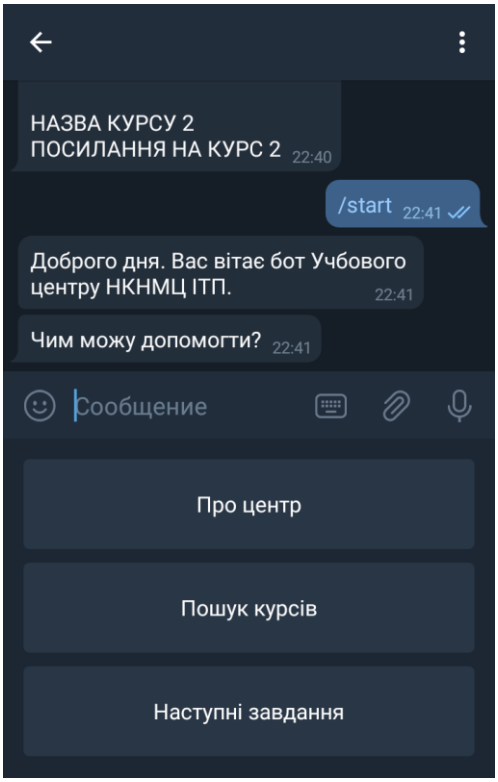

Рисунок А.2 – Макет інтерфейсу для авторизованого користувача

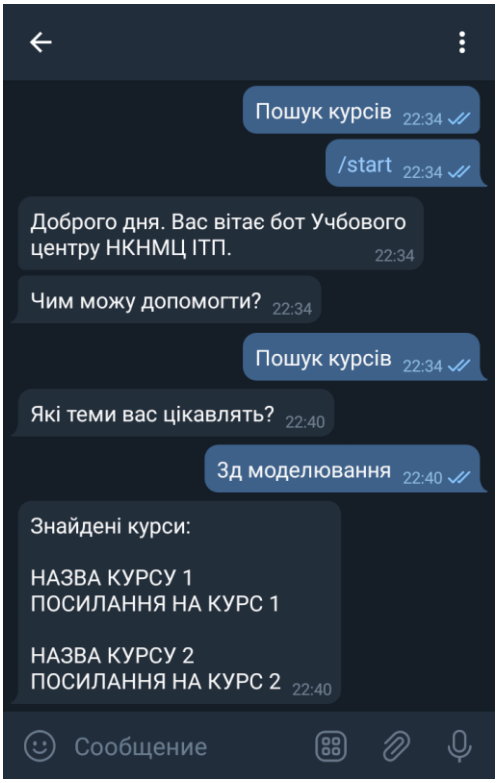

Рисунок А.3 – Макет інтерфейсу пошуку курсів

ДОДАТОК Б
ПЛАНУВАННЯ РОБІТ

1. Ідентифікація ідеї проекту

Метою розроблення інформаційної системи дипломного проекту є підтримка супроводження проведення курсів НКНМЦ ІТП.

Дипломний проект призначений для того, щоб студенти курсів НКНМЦ ІТП мали спосіб отримувати нагадування про завдання до курсів; щоб люди, зацікавлені у курсах НКНМЦ ІТП, мали зручний спосіб шукати цікавлячи їх курси.

Інформаційна система повинна бути реалізована у вигляді чат-боту, доступного в додатку Telegram.

2. Деталізація ідеї проекту

Для деталізації ідей та завдань у менеджменті проектів часто використовується так званий метод SMART — акронім для п’яти критеріїв оцінки кожного етапу проекту.

- **S**pecific — ціль повинна бути ясно та точно визначена, інакше важко буде зосередити на неї сили;
- **M**easurable — ціль повинна бути вимірюваною, щоб можна було відстежувати та оцінювати прогрес розробки;
- **A**chievable — ціль повинна бути досяжною, реалістичною, знаходитися у межах здібностей виконавців;
- **R**elevant — ціль повинна бути релевантною, тобто мати сенс сама по собі або в комплексі з іншими цілями.
- **T**ime-bound — ціль повинна бути обмеженою у часі, мати конкретні строки виконання.

Результати застосування методу SMART до проекту даної роботи можна побачити в табл. Б.1.

Таблиця Б.1 – Деталізація ідеї проекту методом SMART

Критерій	**Опис**
Specific	Розробити чат-бота для супроводження проведення курсів НКНМЦ ІТП
Measurable	З оглядом на некомерційність проекту, результатом є коректна робота та оцінка замовника
Achievable	Виконавець має навички роботи з обраними технологіями та платформами
Relevant	Проект може допомогти людям, зацікавленим у курсах НКНМЦ ІТП
Time-bound	Процес розробки обмежений у часі за допомогою планування

3. Планування структури проекту

Для планування структури використовуються розповсюджені у менеджменті проектів методи WBS та OBS.

WBS розшифровується як Work Breakdown Structure (структура декомпозиції робіт) та використовується для подрібнення деякої цілі на дрібні складові компоненти. Це допомагає зробити організацію діяльності більш вимірюваною та структурованою. Для подрібненого згідно WBS проекту набагато легше створити кошторис чи календарний план робіт та розподілити роботу між учасниками команди.

Одним з основних питань при створенні WBS є визначення рівня деталізації розбиття. Для вирішення цієї та супутніх проблеми існує декілька корисних евристичних правил:

- для більшості малих та середніх проектів достатньо від двох до чотирьох рівнів деталізації;
- ніяка задача на найнижчому рівні розбиття не повинна займати більше ніж 80 годин для виконання;
- ніяка задача на найнижчому рівні розбиття не повинна займати більше ніж довжину звітного періоду.

Розбиття проекту у питанні згідно WBS можна побачити у табл. Б.2 та рис Б.1.

Таблиця Б.2 – WBS проекту

Індекс	**Задача**
1	Планування
1.1	Постановка задачі
1.2	Визначення вимог
1.3	Аналіз аналогів
1.4	Створення ТЗ
1.5	Створення плану робіт
2	Розробка
2.1	Розробка прототипу
2.2	Розробка серверної частини
2.3	Розробка структури БД
2.4	Заповнення БД
2.5	Розробка модулю роботи з БД
2.6	Розробка модулю обробки натуральної мови
2.7	Розробка модулю повідомлень
2.8	Тестування

Продовження таблиці Б.2 – WBS проекту

Індекс	**Задача**
3	Завершення
3.1	Розробка документації
3.2	Здача в експлуатацію

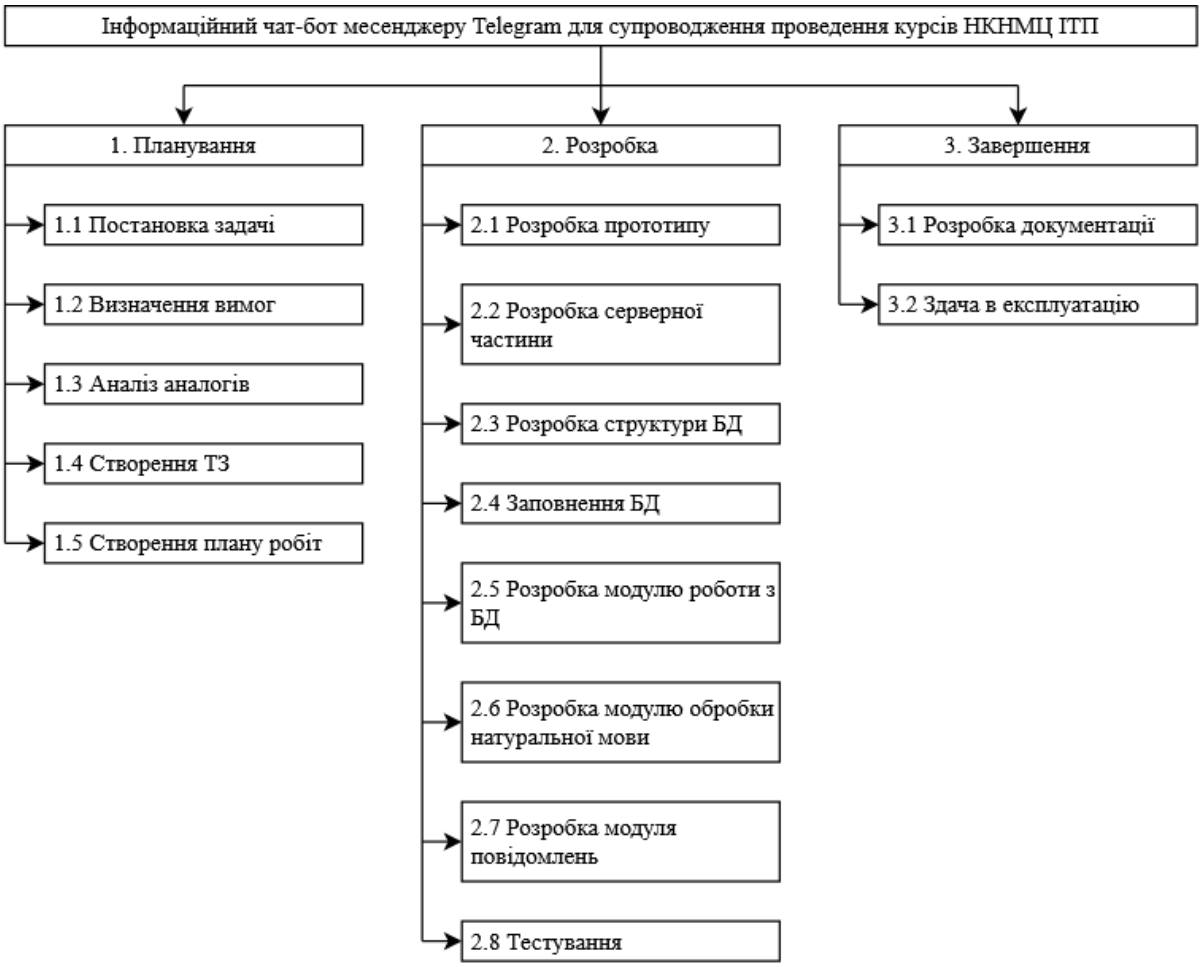

Рисунок Б.1 – WBS проекту

OBS (розшифровується як Organizational Breakdown Structure, структура декомпозиції організації) — ще одна ієрархічна модель, яка майже завжди використовується в комплексі з WBS для розподілення відповідальності за виконання задач найнижчого рівня між виконавцями. Суб’єктами OBS можуть бути

окремі виконавці, команди або структури. Важливо пам’ятати, що кожна задача найнижчого рівня у WBS має мати виконавця у OBS.

Розбиття проекту у питанні згідно OBS можна побачити на рис Б.2.

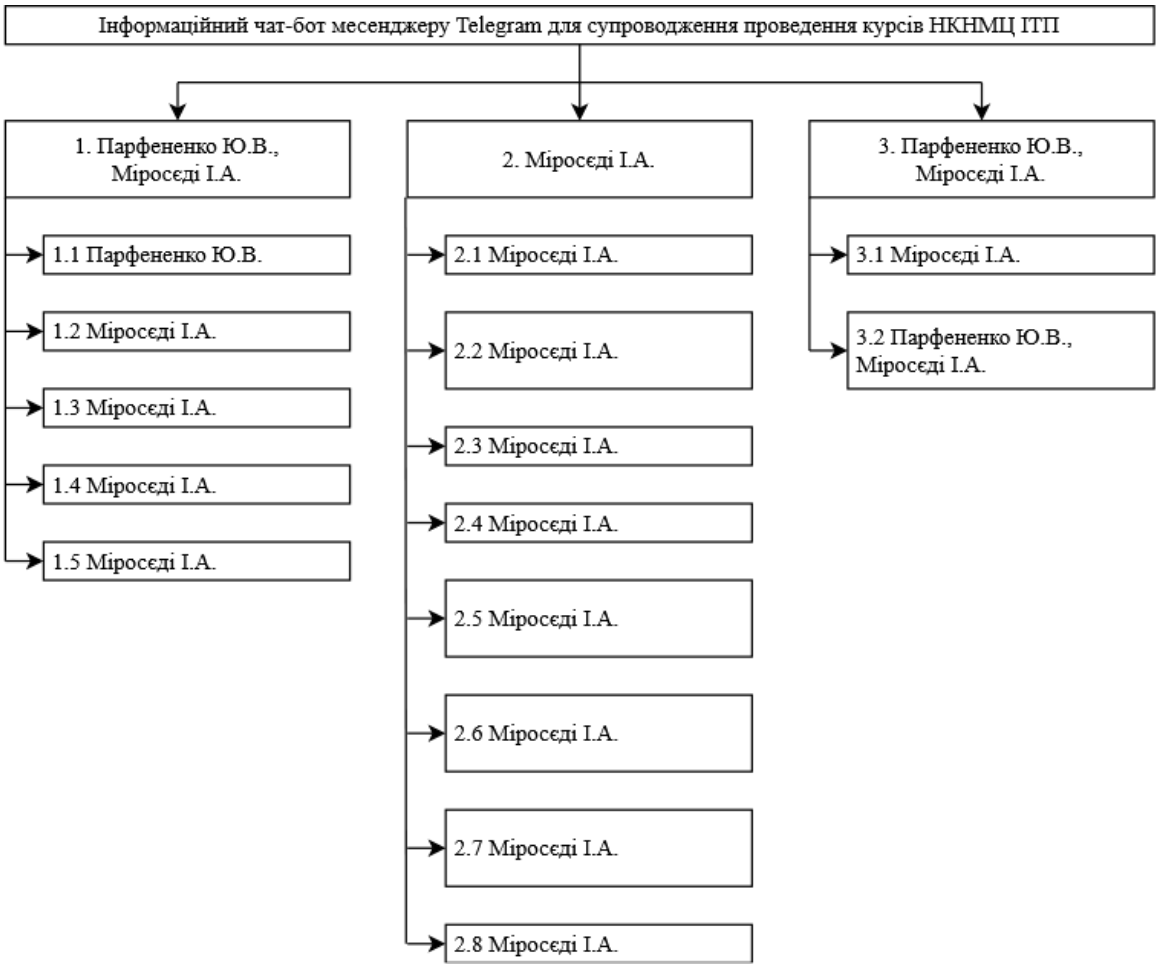

Рисунок Б.2 – OBS проекту

4. Побудова матриці відповідальності

Використовуючи отримані за допомогою WBS та OBS результати можна побудувати матрицю відповідальності. Ця модель встановлює відповідальність кожного учасника проекту за виконання окремих задач.

Найчастіше, згідно з PMBOK Guide, виділяється чотири ступені відповідальності:

- Відповідальний (Accountable) — відповідає за виконання задачі та має право обирати способи її виконання;
- Виконавець (Responsible) — виконує задачу, але не несе відповідальності за вибір методів її вирішення;
- Консультант (Consult before doing) — консультує та контролює якість реалізації;
- Спостерігач (Inform after doing) — спостерігає за ходом реалізації, але не несе ніякої відповідальності.

Завершену матрицю відповідальності для даного проекту можна побачити в табл. Б.3.

Таблиця Б.3 – Матриця відповідальності

Індекс	**Задача**	**Парфененко Ю.В.**	**Міросєді І.А.**
1	Планування		
1.1	Постановка задачі	C	A
1.2	Визначення вимог	I	A
1.3	Аналіз аналогів	I	A
1.4	Створення ТЗ	I	A
1.5	Створення плану робіт	I	A
2	Розробка		
2.1	Розробка прототипу	I	A
2.2	Розробка серверної частини	I	A
2.3	Розробка структури БД	I	A
2.4	Заповнення БД	I	A
2.5	Розробка модулю роботи з БД	I	A
2.6	Розробка модулю обробки натуральної мови	I	A

Продовження таблиці Б.3 – Матриця відповідальності

Індекс	Задача	Парфененко Ю.В.	Міросєді І.А.
2.7	Розробка модулю повідомлень	I	A
2.8	Тестування	I	A
3	Завершення		
3.1	Розробка документації	I	A
3.2	Здача в експлуатацію	C	A

5. Побудова діаграми Ганта

Для швидкого огляду розкладу й прогресу виконання окремих стадій проекту зручно використовувати діаграми Ганта. Вони є різновидом стовпчикових діаграм, де по вертикалі розміщуються актуальні задачі (які вже були відокремлені під час створення WBS), а по горизонталі відміряються інтервали часу. Залежні одна від іншої задачі з'єднуються за допомогою стрілок.

Діаграму Ганта для даного проекту можна побачити на рис. Б.3. Червоні стовпці показують основні етапи проекту, а сині — задачі найнижчого рівня.

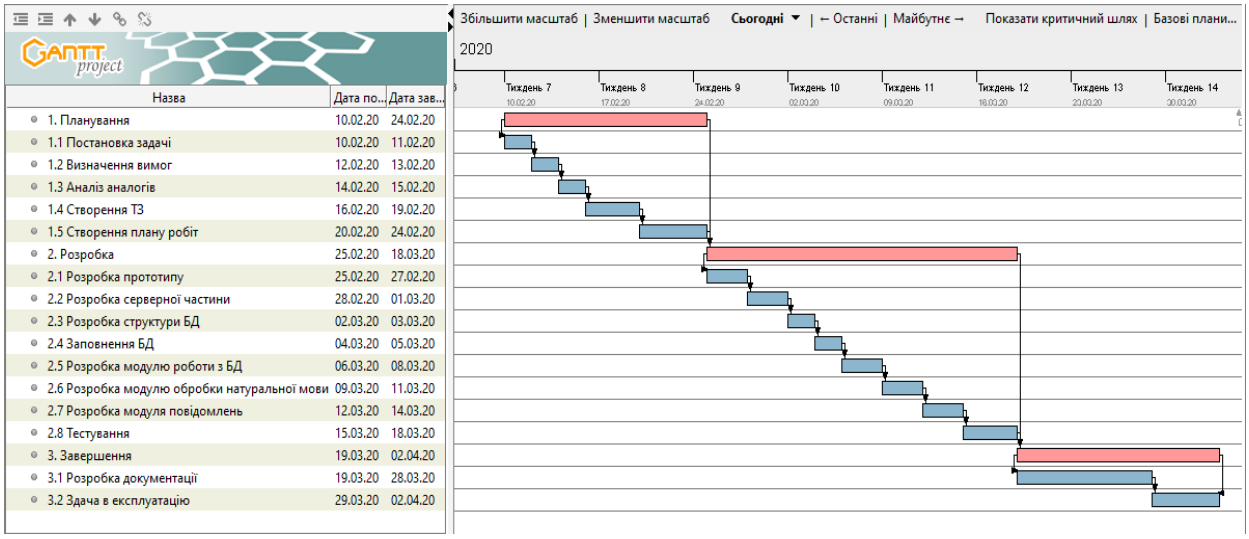

Рисунок Б.3 – Діаграма Ганта

6. Оцінка ризиків

Процес оцінки ризиків призначений для визначення можливих інцидентів у ході реалізації проекту, а також їх ймовірностей та наслідків. Цей процес проводиться на протязі усього життєвого циклу проекту, тому що потенційні ризики постійно змінюються. Існує багато класифікаторів ризиків, але для цієї роботи буде розглянуто лише один, який бере до уваги внутрішню діяльність команди виконавців.

Зовнішні ризики не залежать від діяльності виконавців. До них відносяться політичні, економічні, природні, соціальні ризики, тощо.

Внутрішні ризики напряму залежать від діяльності виконавців. До них належать проектні, технологічні, кадрові, організаційні ризики, тощо.

Для подальшого аналізу використовуються лише внутрішні ризики, тому що їх легше оцінити.

Знайдені у даному проекті ризики можна побачити на рис. Б.4.

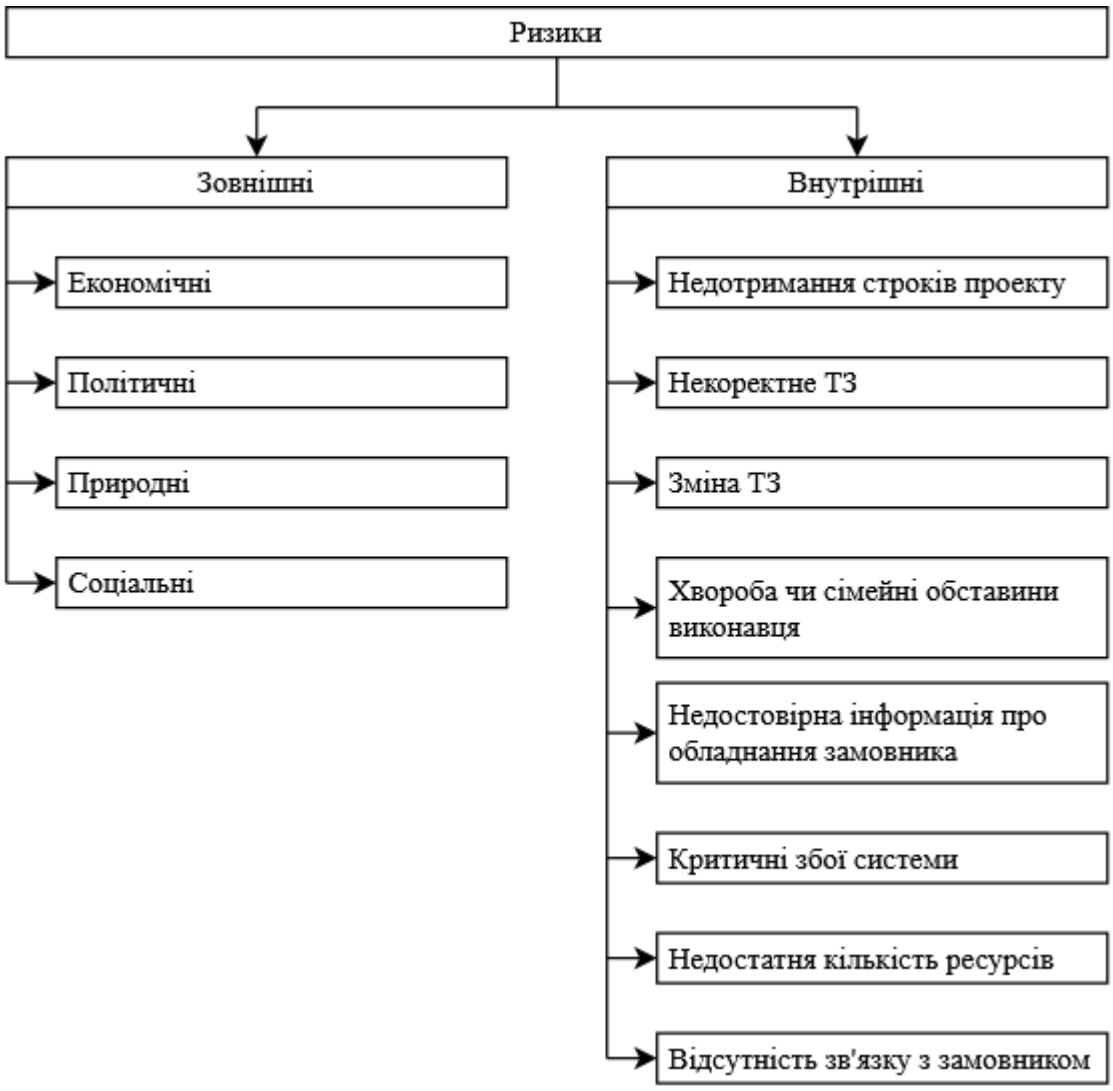

Рисунок Б.4 – Ризики

Одним із кількісних інструментів оцінки ризиків є так звана матриця ризиків. Кожному з виділених ризиків присвоюється два числа, серйозність та ймовірність виникнення.

Серйозність (Severity), в порядку зменшення, ділиться на наступні чотири категорії:

- 4 — катастрофічна;
- 3 — критична;
- 2 — значна;
- 1 — незначна.

Ймовірність (Probability), в порядку зменшення, ділиться на наступні п’ять категорій:

- 5 — надзвичайно висока;
- 4 — висока;
- 3 — середня;
- 2 — низька;
- 1 — вкрай низька.

Обчисливши множення серйозності та ймовірності для кожного із проаналізованих ризиків (нехай отримане число іменується рангом ризику R = S * P), можна виділити ризики у декілька категорій. Дуже часто число категорій беруть рівним чотирьом:

- $1 \leq R \leq 5$ — низький ранг;
- $6 \leq R \leq 10$ — середній ранг;
- $11 \leq R \leq 15$ — високий ранг;
- $16 \leq R \leq 20$ — надзвичайно високий ранг;

Матрицю ризиків для даного проекту можна знайти у табл. Б.4 та на рис. Б.5.

Таблиця Б.4 – Матриця ризиків

Ризик	**Індекс**	**Серйозність**	**Ймовірність**	**Ранг**
Недотримання строків проекту	R1	3	4	Високий
Некоректне ТЗ	R2	2	3	Середній
Зміна ТЗ	R3	3	3	Середній
Хвороба чи сімейні обставини виконавця	R4	4	3	Високий
Недостовірна інформація про обладнання замовника	R5	2	4	Середній

Продовження таблиці Б.4 – Матриця ризиків

Ризик	Індекс	Серйозність	Ймовірність	Ранг
Критичні збої системи	R6	4	4	Надзвичайно високий
Недостатня кількість ресурсів	R7	2	2	Низький
Відсутність зв'язку з замовником	R8	3	2	Середній

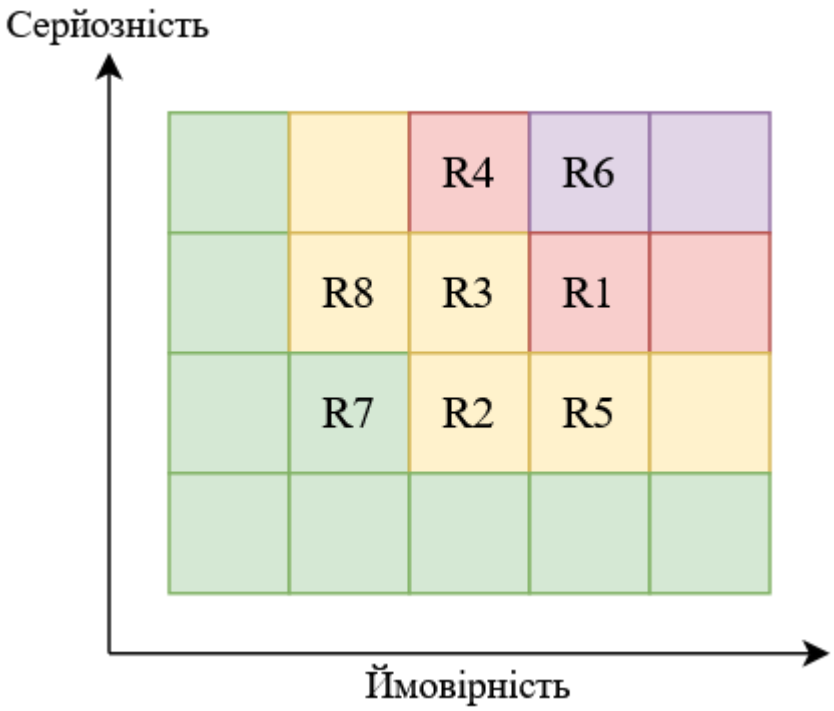

Рисунок Б.5 – Матриця ризиків

ДОДАТОК В
ІНСТРУКЦІЯ ТА ПРИКЛАДИ РОБОТИ

1. Початок роботи

Для початку роботи треба у діалозі з ботом ввести стандартну команду Telegram «/start» або натиснути на «Розпочати» (рис В.1).

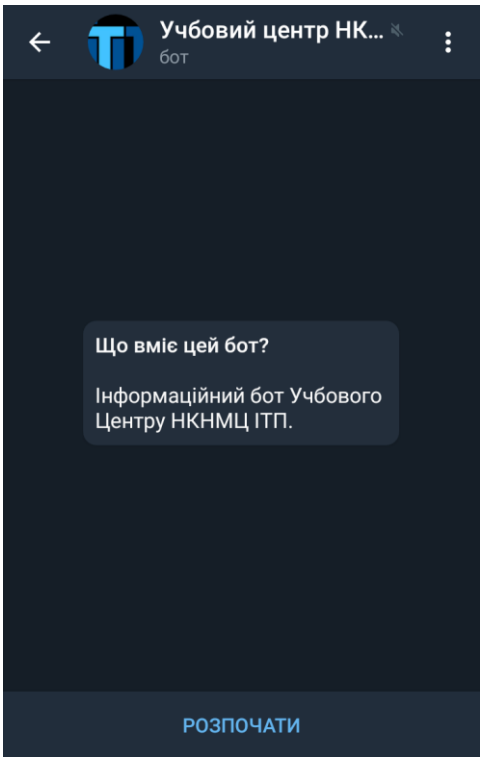

Рисунок В.1 – Початок роботи

2. Неавторизований користувач

Після привітання, неавторизований користувач побачить три доступні кнопки (рис В.2):

– «Про Центр», яка виводить інформацію про НКНМЦ ІТП (рис В.3);

– «Пошук курсів», яка дозволяє шукати курси за ключовими словами (рис В.4). Варто зазначити, що довжина запросу повинна бути більшою за три символи;
– «Авторизація», яка, якщо знайде номер телефону користувача у базі даних, підвищить його права доступу до рівня «авторизований користувач» (рис В.5).

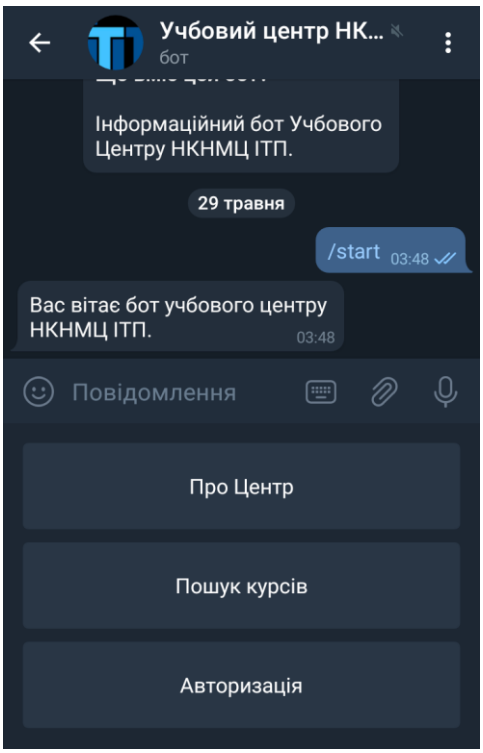

Рисунок В.2 – Кнопки для неавторизованого користувача

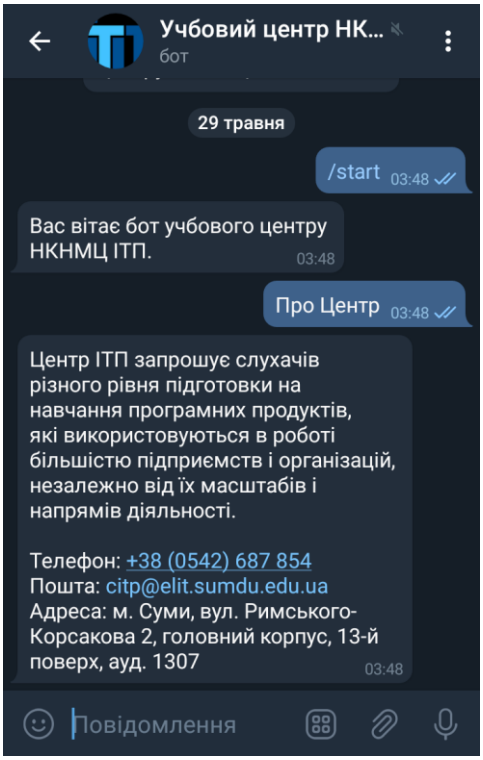

Рисунок В.3 – Отримання інформації про Центр

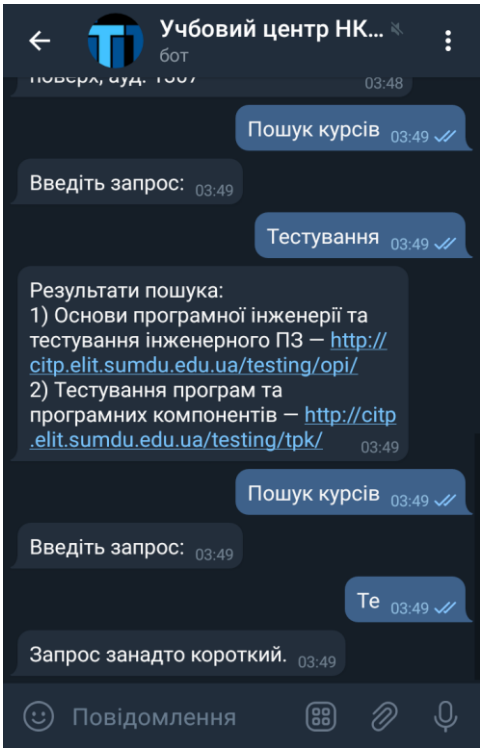

Рисунок В.4 – Приклад пошуку курсів

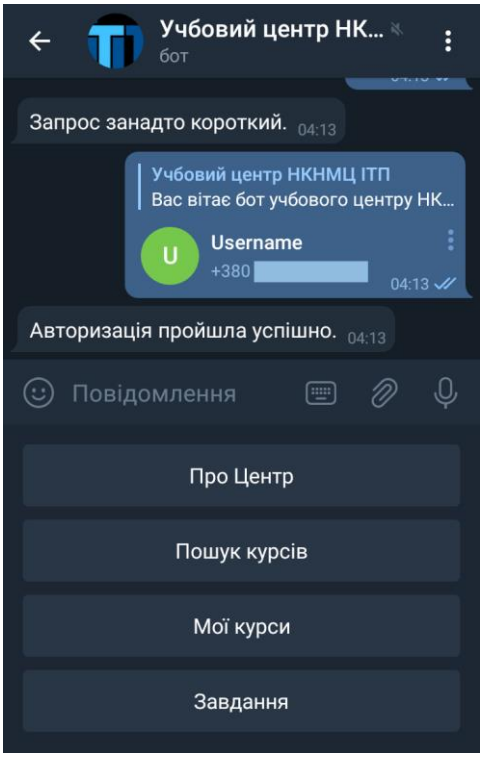

Рисунок В.5 – Авторизація

3. Авторизований користувач

Після привітання або авторизації, авторизований користувач побачить чотири доступні кнопки, дві із яких, «Про Центр» та «Пошук курсів», мають той же функціонал, як і для неавторизованого користувача (рис В.6). Дві інші кнопки мають наступні функції:

– «Мої курси» виводить перелік курсів, які слухає користувач (рис В.7);
– «Завдання» виводить найближчі за терміном здачі завдання із курсів, які слухає користувач (рис В.7).

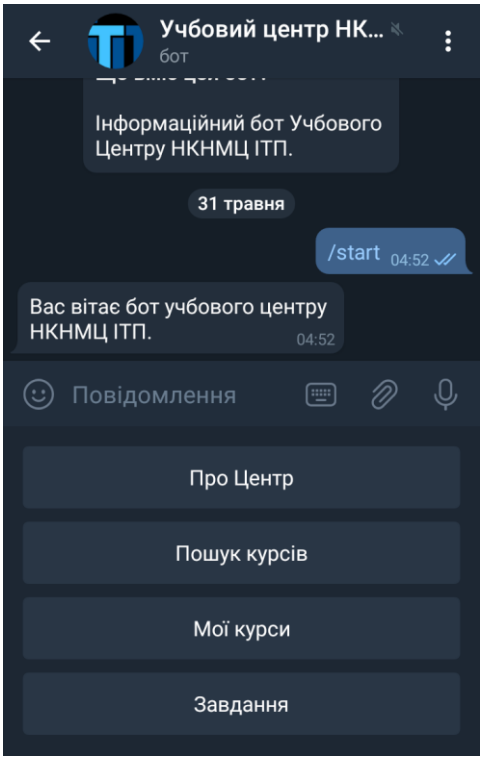

Рисунок В.6 – Кнопки для авторизованого користувача

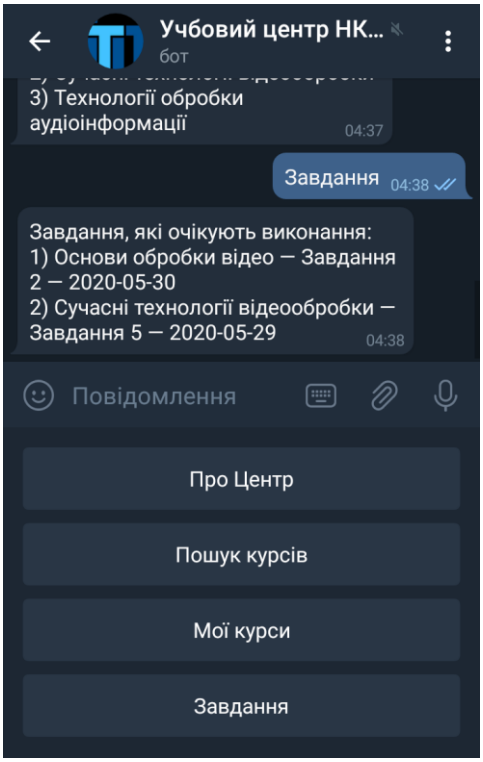

Рисунок В.7 – Отримання термінів виконання завдань

Для керування автоматичними нагадуваннями про терміни виконання завдань авторизований користувач має доступ до команди «/notifications» (рис В.8). Її синтаксис має наступний вигляд:

– «/notifications» дозволяє отримати поточний статус нагадувань;
– «/notifications on» вмикає нагадування;
– «/notifications on» вимикає нагадування.

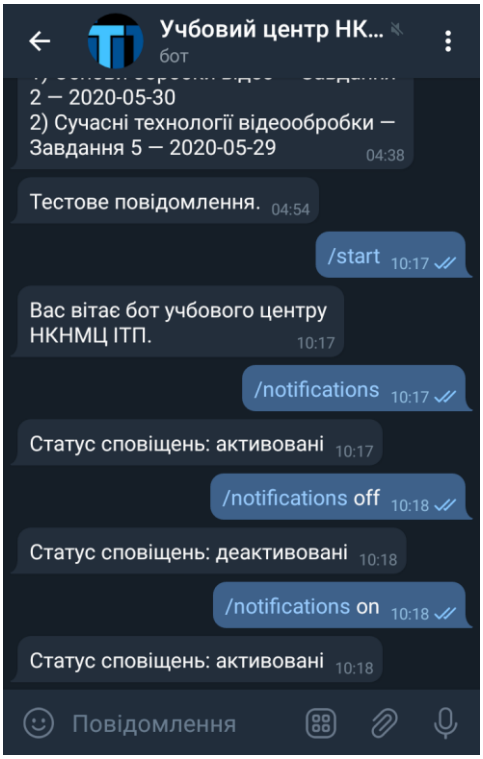

Рисунок В.8 – Керування нагадуваннями

4. Адміністратор

Адміністратор, використовуючи спеціалізований додаток, має доступ до наступних інструментів:

– керування станом серверу та просмотр зафіксованих подій (рис В.9);
– керування механізмом автоматичного нагадування про терміни виконання завдань (рис В.10–В.12). Варто зазначити, що час має бути заданим у UTC.

Це зроблено для збільшення незалежності від часових поясів та переходів між літнім та зимнім часом.

– масова розсилка повідомлень, яка дозволяє оповіщати студентів про важливі новини (рис В.13–В.15).

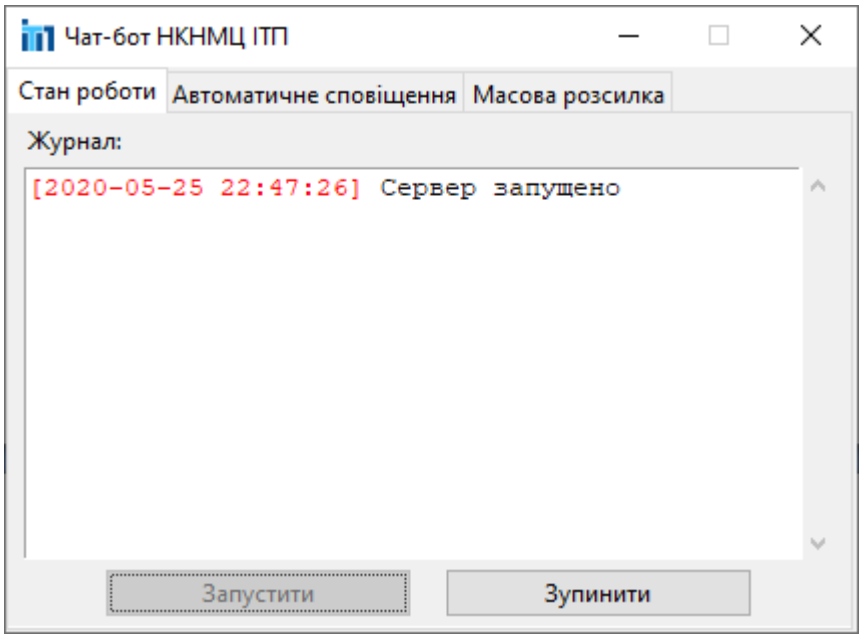

Рисунок В.9 – Журнал та керування станом серверу

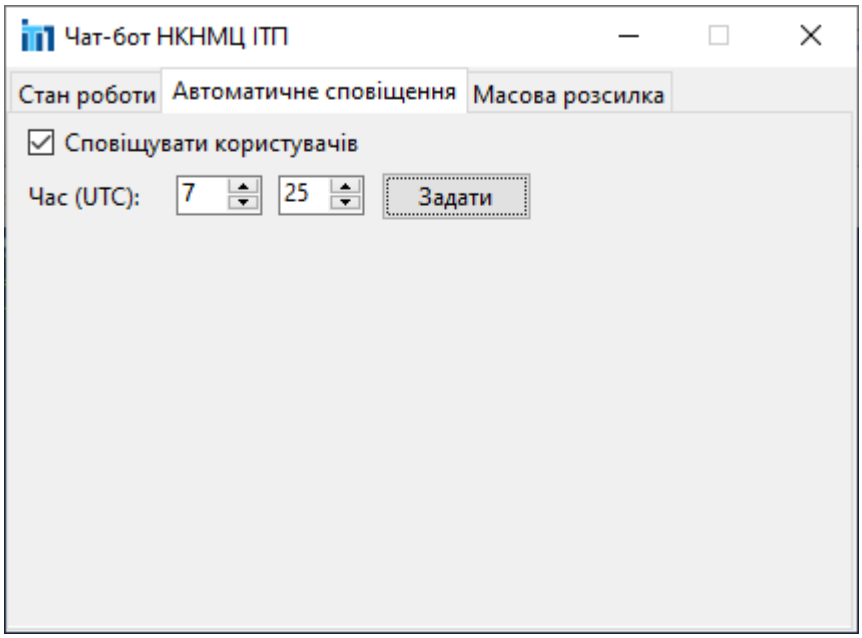

Рисунок В.10 – Автоматичне нагадування

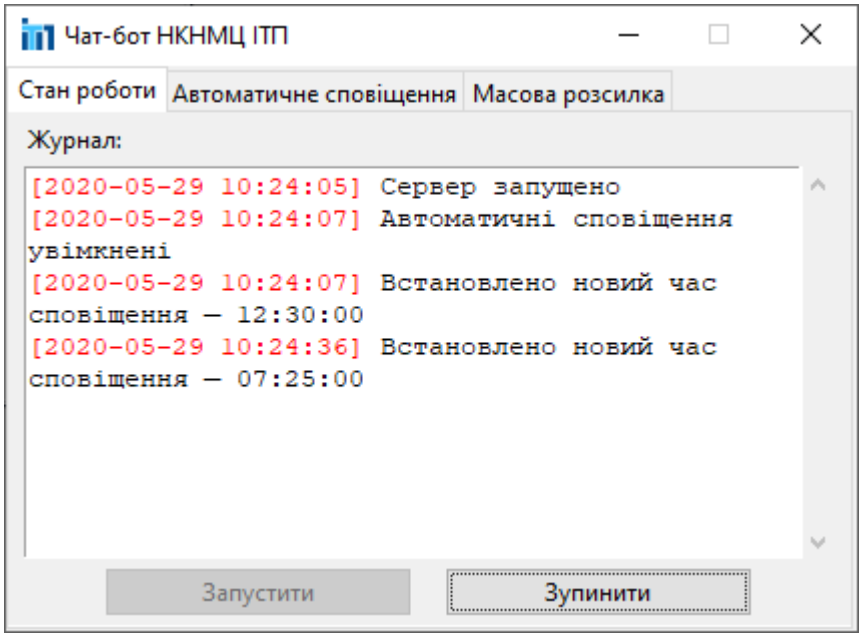

Рисунок В.11 – Автоматичне нагадування (запис у журналі)

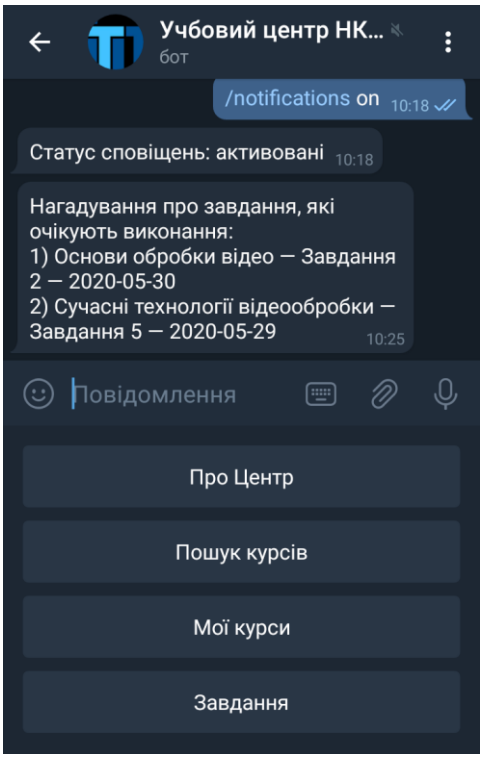

Рисунок В.12 – Автоматичне нагадування (результат)

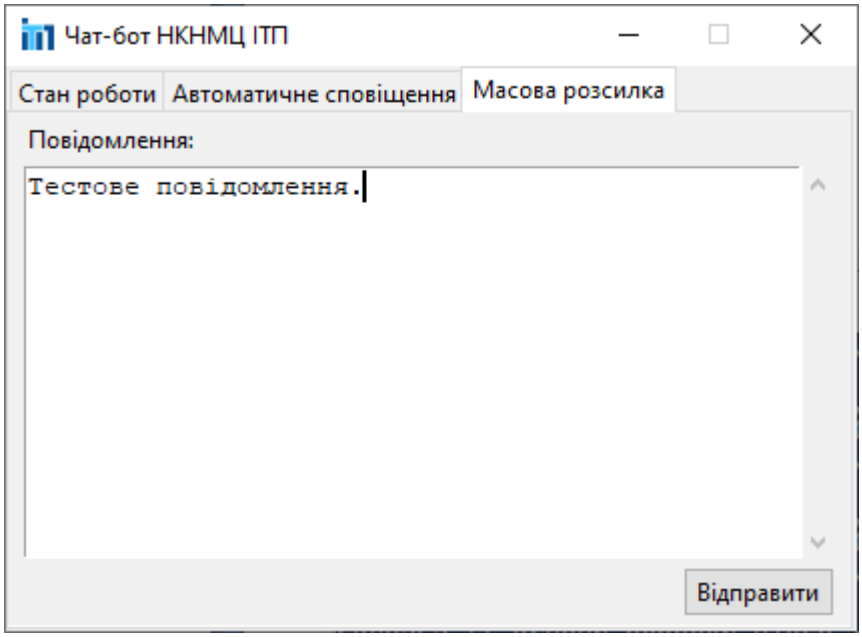

Рисунок В.13 – Масова розсилка

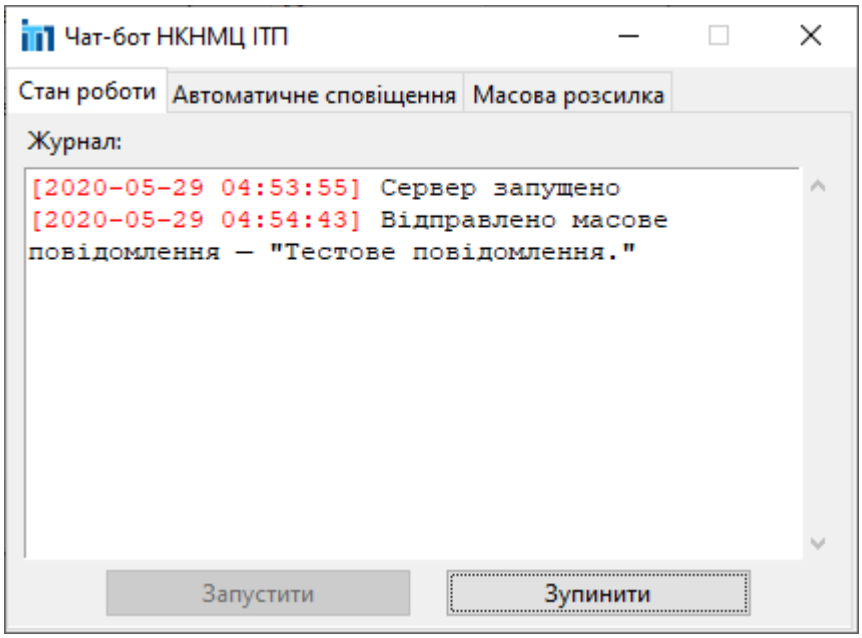

Рисунок В.14 – Масова розсилка (запис у журналі)

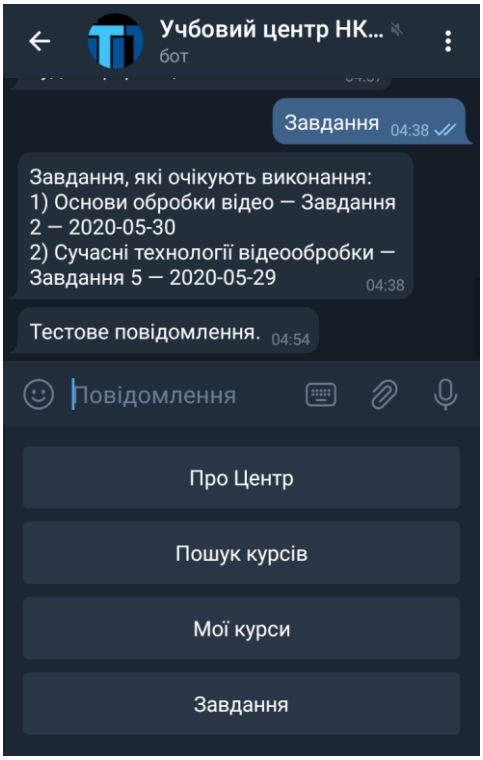

Рисунок В.15 – Масова розсилка (результат)

ДОДАТОК Г
ЛИСТИНГ ПРОГРАМИ

1. Листинг файлу «bot.py»

```
from database import BotDatabase
from server import BotServer
from gui import BotGUI

TOKEN = '1286985606:AAFqkfpWaiUhppUFh-UZl8W_RBhsQFLz2Eg'
DB_NAME = 'database.db'

if __name__ == '__main__':
    bot_database = BotDatabase(DB_NAME)
    bot_server = BotServer(TOKEN, bot_database, 'ua')
    bot_gui = BotGUI(bot_server)
    bot_gui.mainloop()
```

2. Листинг файлу «database.py»

```
class BotDatabase:
    def __init__(self, name):
        self.name = name

    def load_strings(self, language):
        if len(language) != 2:
            return None
```

```
            query = 'SELECT * FROM
l10n_{}'.format(language.lower())
            with sqlite3.connect(self.name) as db:
                return dict(db.execute(query).fetchall())

        def is_user_registered(self, telegram_id):
            query = 'SELECT 1 FROM students WHERE
telegram_id=?'
            with sqlite3.connect(self.name) as db:
                return bool(db.execute(query,
(telegram_id,)).fetchall())

        def is_phone_in_database(self, phone_number):
            query = 'SELECT 1 FROM students WHERE phone=?'
            with sqlite3.connect(self.name) as db:
                return bool(db.execute(query,
(phone_number,)).fetchall())

        def update_telegram_id(self, phone_number,
telegram_id):
            query = 'UPDATE students SET telegram_id=?
WHERE phone=?'
            with sqlite3.connect(self.name) as db:
                return bool(db.execute(query, (telegram_id,
phone_number)).rowcount)

        def search_courses_by_tags(self, tags):
```

```
            filtered_tags = list(filter(lambda s: len(s) >
3, tags))
            if not filtered_tags:
                raise ValueError('query is too short')
            query = ['SELECT name, url FROM courses WHERE
']
            parameters = []
            for i, tag in enumerate(filtered_tags):
                if i > 0:
                    query.append(' OR ')
                query.append('tags LIKE ?')
                parameters.append('%' + tag.lower() + '%')
            with sqlite3.connect(self.name) as db:
                return            db.execute(''.join(query),
parameters).fetchall()

        def find_active_courses(self, telegram_id):
            query_courses = 'SELECT name FROM courses WHERE
course_id IN (SELECT course_id FROM student_course WHERE
student_id=(SELECT    student_id    FROM    students    WHERE
telegram_id=?))'
            with sqlite3.connect(self.name) as db:
                courses     =     db.execute(query_courses,
(telegram_id,))
                return courses.fetchall()

        def find_pending_homework(self, telegram_id):
```

```
            query_courses = 'SELECT course_id FROM
student_course WHERE student_id=(SELECT student_id FROM
students WHERE telegram_id=?)'
            query_homework = 'SELECT courses.name,
description, due FROM (SELECT * FROM homework WHERE course=?
AND due >= date(\'now\') ORDER BY due ASC LIMIT 1) INNER
JOIN courses ON course=courses.course_id'
            with sqlite3.connect(self.name) as db:
                courses = db.execute(query_courses,
(telegram_id,)).fetchall()
                homework = []
                if len(courses) > 0:
                    for course in courses:
                        course_homework =
db.execute(query_homework, (course[0],)).fetchall()
                        if len(course_homework) > 0:

homework.extend(course_homework)
                return homework

        def has_notify_flag(self, telegram_id):
            query = 'SELECT notify FROM students WHERE
telegram_id=?'
            with sqlite3.connect(self.name) as db:
                r = db.execute(query,
(telegram_id,)).fetchall()
                if r:
                    return bool(r[0][0])
```

```
                return False

        def set_notify_flag(self, telegram_id, value):
            query = 'UPDATE students SET notify=? WHERE
telegram_id=?'
            with sqlite3.connect(self.name) as db:
                return bool(db.execute(query, (int(value),
telegram_id)).rowcount)

        def get_notifees(self):
            query = 'SELECT telegram_id FROM students WHERE
notify=1'
            with sqlite3.connect(self.name) as db:
                return db.execute(query).fetchall()
```

3. Листинг файлу «server.py»

```
    class BotServer(Bot):
        states = {'main_unauth': 1, 'main_auth': 2,
'search': 3}

        def __init__(self, token, database, language):
            super().__init__(token)
            self.updater = Updater(token, use_context=True)
            self.dispatcher = self.updater.dispatcher
            self.db = database
            self.strings = self.db.load_strings(language)
            self.error_receiver = None
```

```
            self.generate_keyboards()
            self.load_handlers()

        def l10n(self, string_id):
            try:
                return self.strings[string_id]
            except KeyError as e:
                print('l10n:    id    \'{}\'    does    not
exist.'.format(string_id))
                raise e

        def generate_keyboards(self):
            self.kbd_authorized = ReplyKeyboardMarkup([
                [self.l10n('kbd_info')],
                [self.l10n('kbd_search')],
                [self.l10n('kbd_courses')],
                [self.l10n('kbd_pending')]],
                one_time_keyboard=False)
            self.kbd_unauthorized = ReplyKeyboardMarkup([
                [self.l10n('kbd_info')],
                [self.l10n('kbd_search')],
                [KeyboardButton(self.l10n('kbd_auth'),
                request_contact=True)]],
                one_time_keyboard=False)

        def load_handlers(self):
            states = {
```

```
                self.states['main_unauth']: [

MessageHandler(Filters.regex('^({})$'.format(
                        self.l10n('kbd_info'))),
self.handle_about),

MessageHandler(Filters.regex('^({})$'.format(
                        self.l10n('kbd_search'))),
self.handle_search),
                    MessageHandler(Filters.contact,
self.handle_auth)
                ],
                self.states['main_auth']: [

MessageHandler(Filters.regex('^({})$'.format(
                        self.l10n('kbd_info'))),
self.handle_about),

MessageHandler(Filters.regex('^({})$'.format(
                        self.l10n('kbd_search'))),
self.handle_search),

MessageHandler(Filters.regex('^({})$'.format(
                        self.l10n('kbd_courses'))),
self.handle_courses),

MessageHandler(Filters.regex('^({})$'.format(
```

```
                        self.l10n('kbd_pending'))),
self.handle_pending),
                    CommandHandler('notifications',
self.handle_notifications)
                ],
                self.states['search']: [
                    MessageHandler(Filters.text,
self.handle_search_query),
                    MessageHandler(Filters.all,
self.handle_search_cancel)
                ]
            }
            main_handler = ConversationHandler(
                entry_points=[CommandHandler('start',
self.handle_greeting)],
                states=states,
                fallbacks=[MessageHandler(Filters.all,
self.handle_unknown)],
                conversation_timeout=10.*60)
            self.dispatcher.add_handler(main_handler)

self.dispatcher.add_error_handler(self.handle_error)

        def get_keyboard(self, user_id):
            if self.db.is_user_registered(user_id):
                return self.kbd_authorized
            return self.kbd_unauthorized
```

```
        def get_return_state(self, user_id):
            if self.db.is_user_registered(user_id):
                return self.states['main_auth']
            return self.states['main_unauth']

        def handle_greeting(self, update, context):
            update.message.reply_text(self.l10n('greet'),

reply_markup=self.get_keyboard(update.message.from_user.id))
            return
self.get_return_state(update.message.from_user.id)

        def handle_about(self, update, context):
            update.message.reply_text(self.l10n('info'))
            return
self.get_return_state(update.message.from_user.id)

        def handle_auth(self, update, context):
            if
self.db.is_phone_in_database(update.message.contact.phone_nu
mber):

self.db.update_telegram_id(update.message.contact.phone_numb
er,
                           update.message.from_user.id)

update.message.reply_text(self.l10n('auth_success'),
```

```
reply_markup=self.get_keyboard(update.message.from_user.id))
            else:

update.message.reply_text(self.l10n('auth_failure'))
            return
self.get_return_state(update.message.from_user.id)

        def handle_search(self, update, context):

update.message.reply_text(self.l10n('initiate_search'))
            return self.states['search']

        def handle_search_query(self, update, context):
            try:
                results                                          =
self.db.search_courses_by_tags(update.message.text.split())
            except ValueError:

update.message.reply_text(self.l10n('query_too_short'))
                return
self.get_return_state(update.message.from_user.id)
            if not results:

update.message.reply_text(self.l10n('search_empty'))
            else:
                result = [self.l10n('search_result'), '\n']
                for i, r in enumerate(results):
```

```
                    result.extend([str(i + 1), ') ', r[0],
' — ', r[1], '\n'])
                update.message.reply_text(''.join(result))
            return
self.get_return_state(update.message.from_user.id)

        def handle_search_cancel(self, update, context):

update.message.reply_text(self.l10n('cancel_search'))
            return
self.get_return_state(update.message.from_user.id)

        def build_course_list(self, courses):
            result = []
            for i, r in enumerate(courses):
                result.extend([str(i  +  1),  ')  ',  r[0],
'\n'])
            return result

        def handle_courses(self, update, context):
            courses                                          =
self.db.find_active_courses(update.message.from_user.id)
            if not courses:

update.message.reply_text(self.l10n('courses_empty'))
            else:
                result    =    [self.l10n('courses_result'),
'\n']
```

```
result.extend(self.build_course_list(courses))
                update.message.reply_text(''.join(result))
            return self.states['main_auth']

        def build_homework_list(self, homework):
            result = []
            for i, r in enumerate(homework):
                result.extend([str(i + 1), ') ', r[0], ' —
']) 
                if r[1] is not None:
                    result.extend([r[1], ' — '])
                result.extend([r[2], '\n'])
            return result

        def handle_pending(self, update, context):
            homework                                        =
self.db.find_pending_homework(update.message.from_user.id)
            if not homework:

update.message.reply_text(self.l10n('pending_empty'))
            else:
                result    =    [self.l10n('pending_result'),
'\n']

result.extend(self.build_homework_list(homework))
                update.message.reply_text(''.join(result))
            return self.states['main_auth']
```

```
        def handle_notifications(self, update, context):
            if len(context.args) > 0:
                if context.args[0].lower() == 'on':
                    value = True
                elif context.args[0].lower() == 'off':
                    value = False
                else:

update.message.reply_text(self.l10n('notif_invalid'))
                    return self.states['main_auth']

self.db.set_notify_flag(update.message.from_user.id, value)

            result = [self.l10n('notif_status')]
            if
self.db.has_notify_flag(update.message.from_user.id):

result.append(self.l10n('notif_status_true'))
            else:

result.append(self.l10n('notif_status_false'))
            update.message.reply_text(''.join(result))
            return self.states['main_auth']

        def handle_unknown(self, update, context):

update.message.reply_text(self.l10n('invalid_input'))
```

```
            return
self.get_return_state(update.message.from_user.id)

        def handle_error(self, update, context):
            if self.error_receiver is not None:
                self.error_receiver(update, context)

        def send_notifications(self, context):
            notifees = self.db.get_notifees()
            prefix = self.l10n('notif_prefix')
            for n in notifees:
                homework                                    =
self.db.find_pending_homework(n[0])
                if homework:
                    message = [prefix, '\n']

message.extend(self.build_homework_list(homework))
                    context.bot.send_message(n[0],
''.join(message))

        def enable_notifications(self, utc_time):

self.updater.job_queue.run_daily(self.send_notifications,
utc_time,
                    name='notify')

        def disable_notifications(self):
```

```
            jobs                                                    =
self.updater.job_queue.get_jobs_by_name('notify')
            for job in jobs:
                job.schedule_removal()

        def mass_send(self, message):
            receivers = self.db.get_notifees()
            for r in receivers:
                self.send_message(r[0], message)

        def start(self):
            self.updater.start_polling()

        def stop(self):
            self.updater.stop()
```

4. Лістинг файлу «gui.py»

```
import datetime
import tkinter as tk
from tkinter import ttk
from tkinter import messagebox

class BotGUI(tk.Tk):
    def __init__(self, server):
        self.server = server
        self.server.error_receiver = self.receive_error
        self.running = False
```

```
            # Initialize GUI
            super().__init__()
            self.title('Чат-бот НКНМЦ ІТП')
            self.resizable(False, False)
            self.iconbitmap('itp.ico')
            self.protocol("WM_DELETE_WINDOW",
self.on_close)

            # Create the tab container
            self.tabs = ttk.Notebook(self)

            # Create the first tab
            tab1 = ttk.Frame(self.tabs)
            logs_label = ttk.Label(tab1, text='Журнал:')
            logs_label.pack(anchor='w', padx=8, pady=4)
            logs_frame = ttk.Frame(tab1)
            logs_scrollbar = ttk.Scrollbar(logs_frame)
            logs_scrollbar.pack(side=tk.RIGHT, fill=tk.Y)
            self.logs_text = tk.Text(logs_frame, height=12,
width=48,
                    wrap=tk.WORD, state=tk.DISABLED,
                    yscrollcommand=logs_scrollbar.set)
            self.logs_text.tag_config('TIME',
foreground='red')
            self.logs_text.pack(side=tk.LEFT)

logs_scrollbar.config(command=self.logs_text.yview)
```

```
            logs_frame.pack(padx=8)
            status_frame = ttk.Frame(tab1)
            self.start_button   =   ttk.Button(status_frame,
text='Запустити',
                    width=24, command=self.pressed_start)
            self.start_button.pack(side=tk.LEFT, padx=8)
            self.stop_button   =   ttk.Button(status_frame,
text='Зупинити',
                    width=24,             state=tk.DISABLED,
command=self.pressed_stop)
            self.stop_button.pack(side=tk.LEFT, padx=8)
            status_frame.pack(padx=8, pady=4)
            self.tabs.add(tab1, text='Стан роботи')

            # Create the second tab
            tab2 = ttk.Frame(self.tabs)
            self.notify = tk.IntVar(0)
            notify_checkbutton    =    ttk.Checkbutton(tab2,
command=self.changed_notify,
                    variable=self.notify,  text='Сповіщувати
користувачів')
            notify_checkbutton.pack(anchor='w',      padx=8,
pady=4)
            self.time_frame = ttk.Frame(tab2)
            notify_label    =    ttk.Label(self.time_frame,
text='Час (UTC):',
                    state=tk.DISABLED)
            notify_label.pack(side=tk.LEFT, padx=8)
```

```
            vcmd                                                =
(self.time_frame.register(self.changed_time), '%P', '%W')
            self.h_spinbox   =   ttk.Spinbox(self.time_frame,
from_=0, to=23, width=4,
                    state=tk.DISABLED,       validate='key',
validatecommand=vcmd)
            self.h_spinbox.pack(side=tk.LEFT, padx=8)
            self.m_spinbox   =   ttk.Spinbox(self.time_frame,
from_=0, to=59, width=4,
                    state=tk.DISABLED,       validate='key',
validatecommand=vcmd)
            self.m_spinbox.pack(side=tk.LEFT)
            self.update_button                                  =
ttk.Button(self.time_frame, text='Задати',
                    command=self.pressed_update,
state=tk.DISABLED)
            self.update_button.pack(padx=8)
            self.time_frame.pack(anchor='w')
            self.tabs.add(tab2,            text='Автоматичне
сповіщення', state=tk.DISABLED)

            # Create the third tab
            tab3 = ttk.Frame(self.tabs)
            bulk_label           =           ttk.Label(tab3,
text='Повідомлення:')
            bulk_label.pack(anchor='w', padx=8, pady=4)
            bulk_frame = ttk.Frame(tab3)
            bulk_scrollbar = ttk.Scrollbar(bulk_frame)
```

```
            bulk_scrollbar.pack(side=tk.RIGHT, fill=tk.Y)
            self.bulk_text = tk.Text(bulk_frame, height=12,
width=48, wrap=tk.WORD,
                    yscrollcommand=bulk_scrollbar.set)
            self.bulk_text.pack(side=tk.LEFT)

bulk_scrollbar.config(command=self.bulk_text.yview)
            bulk_frame.pack(padx=8)
            self.bulk_button       =       ttk.Button(tab3,
text='Відправити',
                    command=self.pressed_bulk)
            self.bulk_button.pack(anchor='e',        padx=8,
pady=4)
            self.tabs.add(tab3,   text='Масова   розсилка',
state=tk.DISABLED)

            # Pack the tabs
            self.tabs.pack(fill='both', expand='yes')

            # Set the initial state
            self.h_spinbox.set(12)
            self.m_spinbox.set(30)

        def update_log(self, message):
            now  =  datetime.datetime.now().isoformat('  ',
'seconds')
            self.logs_text.config(state=tk.NORMAL)
```

```
            self.logs_text.insert(tk.END,                '[{}]
'.format(now), 'TIME')
            self.logs_text.insert(tk.END,
'{}\n'.format(message))
            self.logs_text.see(tk.END)
            self.logs_text.config(state=tk.DISABLED)

        def receive_error(self, update, context):
            self.update_log('Помилка                          —
{}'.format(context.error))

        def pressed_start(self):
            self.start_button.config(state=tk.DISABLED)
            self.server.start()
            self.running = True
            self.stop_button.config(state=tk.NORMAL)
            self.tabs.tab(1, state=tk.NORMAL)
            self.tabs.tab(2, state=tk.NORMAL)
            self.update_log('Сервер запущено')

        def pressed_stop(self):
            self.stop_button.config(state=tk.DISABLED)
            self.server.stop()
            self.running = False
            self.start_button.config(state=tk.NORMAL)
            self.tabs.tab(1, state=tk.DISABLED)
            self.tabs.tab(2, state=tk.DISABLED)
            self.update_log('Сервер зупинено')
```

```
        def changed_notify(self):
            if bool(self.notify.get()):
                state = tk.NORMAL
                self.update_log('Автоматичні      сповіщення
увімкнені')
                self.pressed_update()
            else:
                state = tk.DISABLED
                self.server.disable_notifications()
                self.update_log('Автоматичні      сповіщення
вимкнені')
            for child in self.time_frame.winfo_children():
                child.config(state=state)

        def changed_time(self, value, widget):
            return (value.isdigit() and len(value) < 4) or
not value

        def get_time(self):
            iso                                              =
'{:02}:{:02}'.format(int(self.h_spinbox.get()),
                     int(self.m_spinbox.get()))
            return datetime.time.fromisoformat(iso)

        def pressed_update(self):
            try:
                time = self.get_time()
```

```
                self.server.disable_notifications()
                self.server.enable_notifications(time)
                self.update_log('Встановлено новий час
сповіщення — {}'.format(
                        time.isoformat()))
            except ValueError:
                messagebox.showerror('Помилка', 'Невірний
формат часу.')

        def pressed_bulk(self):
            self.bulk_button.config(state=tk.DISABLED)
            message = self.bulk_text.get('1.0', 'end-1c')
            if message:
                self.server.mass_send(message)
                self.bulk_text.delete('1.0', 'end')
                self.update_log('Відправлено масове
повідомлення — \"{}\"'.format(
                        message))
            else:
                messagebox.showerror('Помилка',
                        'Повідомлення не може бути
порожнім.')
            self.bulk_button.config(state=tk.NORMAL)

        def on_close(self):
            if self.running:
                self.server.stop()
            self.destroy()
```