

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Комп'ютерних наук і технологій
(повне найменування факультету)

Комп'ютерні системи та мережі
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)
бакалаврський
(ступінь вищої освіти)

на тему: РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ СТУДЕНТІВ ВНЗ

Виконав: студент 4 курсу,
групи КНТ-511

Спеціальності
123 Комп'ютерна інженерія
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Комп'ютерна інженерія
(назва освітньої програми (спеціалізації))

КОВАЛЬЧУК А.О.
(ПРИЗВИЩЕ та ініціали)

Керівник ТЯГУНОВА М.Ю.
(ПРИЗВИЩЕ та ініціали)

Рецензент ЄФИМЕНКО М.В.
(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет Комп'ютерних наук і технологій
Кафедра комп'ютерних систем та мереж
Ступінь вищої освіти бакалаврський
Спеціальність 123 Комп'ютерна інженерія
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерна інженерія
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри КУДЕРМЕТОВ Р.К.

«14» квітня 2025 року

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

КОВАЛЬЧУКА Артема Олеговича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка інформаційної системи для студентів ВНЗ

керівник проєкту (роботи) к.т.н., доцент, ТЯГУНОВА М.Ю.

(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від « 08 » квітня 2025 року № 151

2. Строк подання студентом проєкту (роботи) 01.06.2025 р.

3. Вихідні дані до проєкту (роботи) перелік студентів академічних груп та їх успішності, необхідно розробити Телеграм-бота із зручним доступом до інформації про успішність навчання студентів.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1) Аналіз інструментів та бібліотек для створення Телеграм-бота;

2) Проєктування системи;

3) Реалізація системи;

4) Результати роботи.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4	ТЯГУНОВА М.Ю.		
Нормоконтроль	ПОЛЬСЬКА О.В.		

7. Дата видачі завдання «11» березня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	до 18.04.2025	
2	Аналіз предметної області	до 22.04.2025	
3	Аналіз інструментів та бібліотек для роботи з Телеграм-ботом	до 01.05.2025	
4	Проектування системи	до 07.05.2025	
5	Реалізація системи	до 15.05.2025	
6	Перевірка працездатності	до 21.05.2025	
7	Оформлення пояснювальної записки	до 25.05.2025	
8	Проходження нормоконтролю	до 01.06.2025	
9	Перевірка на наявність академічного плагіату	до 03.06.2025	
10	Проходження рецензування	до 10.06.2025	

Студент(ка)

(підпис)

Артем КОВАЛЬЧУК

(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис)

Марія ТЯГУНОВА

(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
51 с., 3 табл., 28 рис., 15 джерел.

ІНФОРМАЦІЙНА СИСТЕМА, TELEGRAM-БОТ, АВТОМАТИЗАЦІЯ ОБЛІКУ ОЦІНОК, БАЗА ДАНИХ, PYTHON, AIOGRAM

Об'єкт розробки – інформаційна система для студентів ВНЗ.

Мета роботи – підвищення ефективності навчального процесу та забезпечення зручного, швидкого й безпечного доступу до навчальних результатів студентів шляхом розробки Telegram-бота з автоматизованим наданням інформації студентам.

У першій частині дипломної роботи було проведено аналіз технічного завдання, а також здійснено огляд і порівняння інструментів та бібліотек, необхідних для розробки Telegram-бота.

У другій частині виконано проєктування архітектури програмної системи: сформовано структуру бази даних, визначено основні алгоритми взаємодії користувачів із системою та описано сценарії використання бота для студентів і викладачів.

У третій частині роботи розглянуто практичну реалізацію системи: описано вибір і встановлення програмного забезпечення, процес реєстрації Telegram-бота, модульну структуру коду та реалізацію серверної частини за допомогою Python, бібліотеки Aiogram, SQLAlchemy та aiosqlite. Описано процес розробки CRUD-операцій, впровадження FSM-логіки та забезпечення безпеки й розмежування доступу для різних категорій користувачів.

У четвертій частині продемонстровано результати роботи Telegram-бота у вигляді скріншотів ключових етапів взаємодії користувача з системою.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 51 pages, 3 tables, 28 figures, 15 references.

INFORMATION SYSTEM, TELEGRAM BOT, AUTOMATION OF GRADE RECORDING, DATABASE, PYTHON, AIOGRAM

Object of development – an information system for university students.

Purpose of the work – to increase the efficiency of the educational process and provide convenient, fast, and secure access to students' academic results by developing a Telegram bot with automated information delivery.

In the first part of the thesis, the technical requirements were analyzed, and an overview and comparison of the tools and libraries necessary for Telegram bot development were performed.

In the second part, the architecture of the software system was designed: the database structure was formed, the main user interaction algorithms were defined, and usage scenarios for both students and teachers were described.

The third part considers the practical implementation of the system: the choice and installation of software, the process of Telegram bot registration, the modular structure of the code, and the implementation of the server part using Python, the Aiogram library, SQLAlchemy, and aiosqlite are described. The process of developing CRUD operations, implementing FSM logic, and ensuring security and differentiated access for various user categories is covered.

The fourth part demonstrates the results of the Telegram bot's operation in the form of screenshots of the key stages of user interaction with the system.

ЗМІСТ

Вступ.....	8
1 Аналіз інструментів та бібліотек для роботи з Телеграм-ботом.....	10
1.1 Aiogram та порівняння його з іншими бібліотеками.....	10
1.2 Порівняння SQLite з іншими СУБД.....	13
1.3 SQLAlchemy.....	16
1.4 Aiosqlite	17
2 Проєктування системи.....	18
2.1 Призначення системи та висунуті вимоги до неї.....	18
2.2 Загальна структура системи та її складові.....	20
2.3 Опис функціонування системи	21
3 Реалізація системи.....	23
3.1 Встановлення необхідного ПЗ та реєстрація бота	23
3.1.1 Завантаження Python та необхідних бібліотек.....	23
3.1.2 Реєстрація у BotFather.....	24
3.2 Опис структури коду, взаємодії та модульності.....	28
3.3 Опис структури бази даних	29
3.4 Реалізація серверної системи	31
3.4.1 Створення моделей даних	31
3.4.2 Ініціалізація бази даних	32
3.4.3 Реалізація CRUD-операцій	32
3.5 Реалізація логіки Telegram-бота	34
3.5.1 FSM-логіка покрокової взаємодії	34
3.5.2 Основні хендлери та їх функціонал	35
3.5.3 Динамічні inline-клавіатури	35
3.6 Реалізація безпеки та розмежування доступу до інформації.....	36
4 Результати роботи	38
Висновки	48
Перелік джерел посилання	50

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД	– База даних;
СУБД	– Система управління базами даних;
API	– Application Programming Interface, Прикладний програмний інтерфейс;
ID	– Identifier, Ідентифікатор;
CRUD	– Create, Read, Update, Delete, Створення, читання, оновлення, видалення;
FSM	– Finite State Machine, Скінченний автомат станів.

ВСТУП

Цифровізація освіти в Україні сьогодні є необхідністю в умовах переходу на онлайн навчання. Навчальні установи активно впроваджують сучасні ІТ-рішення, щоб оптимізувати навчальний процес, зробити його більш прозорим, доступним та зручним як для викладачів, так і для студентів. Однією з важливих складових є автоматизація системи обміну інформацією між навчальним закладом і здобувачами освіти.

Отримання оцінок – базовий, але водночас критично важливий елемент зворотного зв'язку у процесі навчання. Традиційні способи інформування про результати (оголошення на парах, списки в електронних таблицях, запит в деканат) часто виявляються незручними, повільними або навіть неможливими в умовах онлайн-навчання. Крім того, вони потребують зайвих дій з боку викладача або персоналу деканату, що вимагає часу і може викликати затримки.

При цьому, більшість студентів активно користуються мобільними пристроями та месенджерами. Один з найпопулярніших – Telegram – уже став звичним каналом комунікації щодо навчання та у повсякденному житті молоді. Саме тому використання Telegram-бота для отримання оцінок є логічним та зручним рішенням.

Telegram-бот – це програмний інструмент, що працює у месенджері та дозволяє автоматично відповідати на запити користувача, взаємодіяти з базою даних та забезпечувати швидкий доступ до інформації. Такий бот може ідентифікувати студента, отримувати дані про його навчальні результати з бази даних та повертати їх у зручному форматі – майже миттєво та без зайвих зусиль.

Метою цієї дипломної роботи є підвищення ефективності навчального процесу та забезпечення зручного, швидкого й безпечного доступу до навчальних результатів студентів шляхом розробки Telegram-бота з автоматизованим наданням інформації студентам.

Для досягнення поставленої мети необхідно виконати такі завдання:

- проаналізувати сучасні інструменти створення ботів та обрати необхідні для розробки;
- визначити вимоги до розроблюваної системи;
- спроектувати архітектуру боту;
- спроектувати інтерфейс системи;
- реалізувати систему;
- проаналізувати отримані результати.

У процесі реалізації буде спроектована архітектура системи, створено інтерфейс взаємодії з користувачем, створення бази даних придатної для редагування та подальшого використання на практиці та забезпечено базовий рівень захисту даних.

Розробка такого рішення є не лише технічно цікавою задачею, але й має практичну цінність для навчального закладу, який прагне бути сучасним, цифровим і орієнтованим на потреби студентів.

1 АНАЛІЗ БІБЛІОТЕК ТА ІНСТРУМЕНТІВ РОЗРОБКИ ТЕЛЕГРАМ-БОТА

1.1 Aiogram та порівняння його з іншими бібліотеками

У процесі створення Telegram-бота для взаємодії студентів з електронною базою оцінювання ключовим технічним рішенням є вибір інструменту для роботи з Telegram Bot API. На сьогодні існує декілька поширених бібліотек, які дозволяють ефективно реалізувати таку систему, зокрема `python-telegram-bot`, `Telethon`, `Pyrogram` і `Aiogram`. Кожна з них має свої особливості та сфери застосування, однак у рамках даного проєкту було обрано саме `Aiogram` – сучасну, асинхронну бібліотеку на мові Python [1, 2] (таблиця 1.1).

`python-telegram-bot` – одна з найстаріших і найпопулярніших бібліотек, побудована на синхронній моделі. Її перевага – простота для новачків і розвинене ком'юніті. Проте при роботі з великою кількістю одночасних запитів або в асинхронному середовищі її використання може спричинити зниження ефективності або потребу в додаткових механізмах, що ускладнює реалізацію [3].

`Telethon` [4] та `Pyrogram` [5] – орієнтовані більше на створення клієнтів, а не ботів. Їхнє використання виправдане у випадках, коли потрібно емулювати роботу звичайного користувача, а не бота. Для цілей цього проєкту – офіційного Telegram-бота для студентів не є оптимальним підходом.

`Aiogram` – сучасна Python-бібліотека для створення Telegram-ботів, побудована на асинхронній архітектурі з використанням модуля `asyncio`. Завдяки цьому вона чудово підходить для розробки високопродуктивних застосунків, здатних обробляти велику кількість запитів одночасно без затримок у роботі [6].

На відміну від інших рішень, ця бібліотека з самого початку проєктувалася з урахуванням принципів неблокуючого програмування. Весь її інтерфейс повністю адаптований для роботи з `async/await`, що значно полегшує взаємодію з такими інструментами, як `aiosqlite` або асинхронні HTTP-клієнти. Завдяки цьому

розробник може реалізувати складну логіку без використання сторонніх обгортки чи додаткових засобів, що спрощує підтримку і розширення системи.

Асинхронність дозволяє боту залишатися активним навіть у момент очікування відповіді від зовнішнього джерела. Наприклад, під час звернення до бази даних або обробки запитів через API інші користувачі не стикаються з затримками у відповідях. Це особливо важливо в умовах, коли бот має працювати із великою кількістю одночасних сесій або активно взаємодіє з базою даних.

Aiogram також вирізняється зручною та гнучкою структурою. Вона дозволяє організувати код у вигляді окремих модулів – для обробки повідомлень, перевірки прав доступу, генерації клавіатур, взаємодії з базою тощо. Такий підхід позитивно впливає як на швидкість розробки, так і на подальшу підтримку: у разі потреби розширення функціональності додати нові компоненти (наприклад, розклад занять та екзаменів) можна без серйозних змін у вже реалізованій логіці.

Окрему увагу варто приділити можливостям контролю доступу. Незважаючи на вбудований захист Telegram, Aiogram дозволяє легко реалізувати перевірку ID користувача або валідацію введених даних, що критично важливо в освітніх проєктах, де обробляються персональні дані, зокрема інформація про успішність студентів.

Ще однією сильною стороною є активна спільнота. Aiogram має добре структуровану документацію, численні приклади та відкриті репозиторії з готовими рішеннями. Це дає змогу швидко знаходити відповіді на питання, що виникають у процесі розробки, і значно скорочує час на пошук технічних рішень.

У межах цього проєкту використання Aiogram забезпечило створення надійного, функціонального й зручного Telegram-бота, який відповідає всім поставленим вимогам. Вибір цієї бібліотеки став не лише технічно вдалим, а й практично обґрунтованим – вона дозволила реалізувати рішення, орієнтоване на зручність кінцевого користувача, стабільність роботи та легкість розширення в майбутньому.

З урахуванням вимог до асинхронності, масштабованості та простоти інтеграції з іншими Python-інструментами, Aiogram є оптимальним вибором для реалізації Telegram-бота в рамках цієї дипломної роботи.

Таблиця 1.1 – порівняння бібліотек

Характеристика	Aiogram	python-telegram-bot	Telethon	Pyrogram
Модель роботи	Асинхронна (asyncio)	Синхронна (частково асинхронна)	Асинхронна	Асинхронна / синхронна
API	Telegram Bot API	Telegram Bot API	Telegram MTPROTO API	Telegram MTPROTO API
Призначення	Боти	Боти	Клієнти (емуляція користувача)	Клієнти (емуляція користувача)
Простота інтеграції	Висока (чітка структура)	Висока для новачків	Складніша конфігурація	Середній рівень складності
Підтримка асинхронності	Повна	Частково (через додаткові модулі)	Повна	Часткова/повна (залежить від режиму)
Використання FSM	Є вбудована підтримка	Немає	Не передбачено	Не передбачено
Оновлення API Telegram	Швидке реагування на зміни	Швидке	Затримка можлива	Затримка можлива
Масштабованість	Висока	Обмежена без додаткових потоків	Висока	Висока
Підтримка inline-елементів	Повна	Повна	Непряма	Непряма
Сумісність з SQLite / SQLAlchemy	Відмінна (через aiomysql, databases, asyncpg тощо)	Обмежена (через синхронність)	Можлива, але не типова	Можлива, але не типова

1.2 Порівняння SQLite з іншими СУБД

У контексті розробки невеликих застосунків, особливо тих, що не потребують складної інфраструктури, база даних SQLite є одним з найбільш практичних рішень. Вона представляє собою вбудовану реляційну СУБД, яка зберігає всі дані в одному файлі на диску і не потребує розгортання окремого серверного середовища чи додаткових служб. Це робить SQLite ідеальним вибором для мобільних застосунків, локальних програм, тестових середовищ, а також для невеликих серверних проєктів, зокрема Telegram-ботів [7].

Одна з ключових переваг SQLite полягає в її простоті. Усе, що потрібно для запуску – це сам файл бази даних і бібліотека доступу до нього, яка вже вбудована в стандартну бібліотеку Python. Цей фактор значно знижує поріг входу для розробників і дозволяє швидко розпочати роботу з даними без додаткових витрат часу на конфігурацію або адміністрування.

Попри простоту, SQLite підтримує майже повну специфікацію мови SQL: тут доступні таблиці, індекси, обмеження, транзакції, агрегатні функції, зовнішні ключі тощо. Це дає змогу будувати повноцінні моделі даних і реалізовувати складні запити, не відмовляючись від принципів реляційної моделі. Також система досить стабільна й протестована на мільйонах реальних пристроїв – SQLite використовується, наприклад, у браузерях (зокрема Firefox), операційних системах, Android-додатках, і навіть у деяких компонентах сучасних баз даних.

Проте SQLite не є універсальним рішенням і має свої обмеження. Найважливішим з них є обробка одночасного доступу: хоча система дозволяє багатьом процесам читати з бази одночасно, вона підтримує лише один записуючий процес у конкретний момент часу. Це означає, що при високому навантаженні або великій кількості конкурентних операцій запису можуть виникати конфлікти чи затримки. Саме тому SQLite найкраще підходить для застосунків з невисоким рівнем паралелізму, або таких, де переважають операції читання. Наприклад, як у створеному боті, де основні користувачі – студенти

мають доступ лише до функції читання, а заповнюванням на практиці буде займатись, скоріш за все, одна людина.

У рамках даного проєкту SQLite обрано свідомо: Telegram-бот не обробляє великої кількості одночасних запитів і працює у межах передбачуваного сценарію – отримання, зберігання та виведення оцінок. У таких умовах переваги SQLite – легкість, швидкість, відсутність потреби в сервері – переважають її обмеження. А у поєднанні з бібліотекою `aiosqlite`, яка дозволяє використовувати асинхронний доступ до бази, рішення стає ще більш гнучким та ефективним у контексті асинхронної архітектури Telegram-бота. Проте для повноти картини доцільно розглянути й альтернативні варіанти, які могли б бути застосовані в аналогічних проєктах – насамперед PostgreSQL, MySQL та MongoDB (таблиця 1.2).

PostgreSQL – це потужна серверна реляційна СУБД, яка підтримує всі сучасні можливості роботи з даними: складні SQL-запити, транзакції, роботу з JSON, CTE (загальні табличні вирази), розширення, зберігані процедури тощо. Вона ідеально підходить для великих систем, де важлива надійність, масштабованість і висока паралельність обробки запитів. Однак використання PostgreSQL потребує налаштування окремого сервера, що ускладнює розгортання проєкту, особливо якщо мова йде про невеликий бот, який запускається на звичайному VPS або локальному хості [8].

MySQL, як ще одна популярна серверна реляційна база даних, має трохи меншу функціональну глибину у порівнянні з PostgreSQL, проте теж забезпечує стабільну роботу. Вона часто обирається у веброзробці через широку підтримку хостингами та знайомість розробників. Але, як і PostgreSQL, MySQL потребує окремого інсталяційного середовища й більше ресурсів для обслуговування [9].

Окремо варто згадати MongoDB, яка представляє клас NoSQL-баз даних і оперує документами у форматі BSON (аналог JSON). Цей підхід дає гнучкість у структурі даних, що зручно для застосунків, де дані мають змінний формат або не підпорядковуються суворій схемі. Проте в проєктах, де чітко визначена структура, як-от список студентів, оцінки, дисципліни, реляційна модель виявляється логічнішою та зручнішою [10].

У підсумку можна сказати, що хоча всі згадані бази даних мають свої переваги, SQLite у поєднанні з aiomysql є найоптимальнішим вибором для проєктів на кшталт цього Telegram-бота. Вона проста в інтеграції, не вимагає окремого серверного середовища, добре працює з асинхронною логікою та забезпечує достатню продуктивність для задач невеликого масштабу. У випадку, якщо проєкт розширюватиметься або зростатиме навантаження, його архітектура дозволяє відносно безболісно перейти на більш потужну СУБД без кардинального переписування логіки.

Таблиця 1.2 – порівняння СУБД

Критерій	SQLite	PostgreSQL	MySQL	MongoDB (NoSQL)
Тип СУБД	Вбудована, файлова	Повнофункціональна, серверна	Серверна реляційна	Документо-орієнтована
Встановлення / обслуговування	Не потребує сервера, легко підключити	Потрібен сервер і налаштування	Також потребує сервера	Необхідна окрема інсталяція сервера
Продуктивність на малих проєктах	Висока	Висока	Висока	Середня при невеликих об'ємах
Паралельна обробка запитів	Обмежена (особливо запис)	Висока	Висока	Висока
Підтримка асинхронного доступу	Через aiomysql	Через asyncpg, databases	Через aiomysql	Через motor або подібні клієнти
Масштабованість	Низька	Висока	Висока	Висока
Надійність у великих системах	Обмежена	Висока	Висока	Висока
Підтримка складних запитів	Є, але без розвинених механізмів	Повна (CTE, JSON, геодані, тощо)	Обмежена порівняно з PostgreSQL	Відсутня SQL, використовується інший підхід
Ідеальне застосування	Невеликі, локальні або прості проєкти	Складні вебзастосунки, API, великі бази	Проєкти середньої складності	Гнучкі сховища без жорсткої схеми

1.3 SQLAlchemy

У сучасній розробці програмного забезпечення практично неможливо обійтися без збереження та обробки даних. Незалежно від масштабу проєкту – чи то вебзастосунок, десктопна програма, чи Telegram-бот – дані потребують надійного зберігання, структурованого доступу та можливості обробки. У мові Python одним із найбільш гнучких і потужних рішень для цього є бібліотека SQLAlchemy.

SQLAlchemy – це бібліотека для роботи з базами даних, яка підтримує два рівні доступу: низькорівневий (SQL Expression Language) і високорівневий (ORM – Object-Relational Mapping). Саме завдяки ORM SQLAlchemy здобула широку популярність: вона дозволяє розглядати структуру бази даних як систему Python-класів, а роботу з даними – як взаємодію з об'єктами цих класів [11].

Інакше кажучи, замість того, щоб оперувати SQL-запитами вручну, розробник працює з Python-об'єктами. Кожен клас відповідає окремій таблиці в базі даних, а кожен об'єкт – окремому рядку в цій таблиці. Такий підхід значно підвищує читабельність і підтримуваність коду, а також полегшує взаємодію з базою даних, особливо в складних або динамічних проєктах.

Окрім зручної моделі опису структури бази, SQLAlchemy надає інструменти для: створення сесій (контекстів роботи з даними), виконання запитів будь-якої складності (включно з фільтрацією, сортуванням, агрегацією тощо), побудови зв'язків між таблицями (один-до-одного, один-до-багатьох, багато-до-багатьох) та автоматичного контролю цілісності даних.

З технічної точки зору, SQLAlchemy підтримує роботу з різними типами баз даних: SQLite, PostgreSQL, MySQL, Oracle та іншими. Завдяки цьому, розробник може легко переносити проєкт між середовищами без суттєвих змін у коді. Наприклад, під час розробки Telegram-бота цілком зручно використовувати SQLite як легку базу, а після завершення – перейти на щось інше для стабільної роботи на сервері.

Таким чином, SQLAlchemy – це не просто бібліотека для зручного доступу до бази даних, а повноцінний інструмент архітектурного рівня, який дозволяє грамотно організувати взаємодію між логікою застосунку та його сховищем даних. У контексті дипломного проєкту, де важливо забезпечити чітку структуру, надійність та масштабованість, використання SQLAlchemy є обґрунтованим і технічно доцільним рішенням.

1.4 Aiosqlite

У сучасних застосунках на Python, які працюють у режимі реального часу або обслуговують багато користувачів одночасно – як, наприклад, Telegram-боти – дедалі більшого значення набуває асинхронність. Вона дозволяє ефективно використовувати ресурси програми, не блокуючи виконання на час очікування зовнішніх операцій, зокрема доступу до бази даних. Класичні драйвери для SQLite працюють синхронно, що створює проблему у проєктах, де загальна логіка побудована на `asyncio`. Для її вирішення була створена бібліотека `aiosqlite`.

`aiosqlite` – це обгортка над стандартною бібліотекою `sqlite3`, яка реалізує підтримку асинхронного доступу до бази даних через корутини (`async def / await`) [12, 13]. За своєю суттю, вона забезпечує неблокуюче виконання SQL-запитів, що дає змогу зберігати послідовність виконання програми навіть під час звернення до бази. Це особливо корисно в тих випадках, коли застосунок одночасно обробляє кілька подій, наприклад, запити від різних користувачів.

З технічного боку, `aiosqlite` не створює повноцінний асинхронний драйвер з нуля – вона працює шляхом делегування запитів у окремий потік, завдяки чому інші частини програми можуть продовжувати виконання, не чекаючи завершення операції запису чи читання. Такий підхід має свої обмеження, однак у більшості невеликих проєктів він дає змогу досягти прийняттого рівня продуктивності.

Робота з `aiosqlite` дуже схожа на класичну модель взаємодії з `sqlite3`, але відбувається в асинхронному контексті. Для відкриття з'єднання, виконання запиту, читання результатів або завершення транзакції використовуються ключові слова `async with` та `await`. Такий синтаксис добре інтегрується з асинхронними фреймворками, зокрема `aiogram`, і забезпечує логічну цілісність коду: всі частини проєкту працюють в одному стилі без необхідності перемикатися між синхронним та асинхронним підходами.

Незважаючи на простоту, `aiosqlite` має усе необхідне для реалізації базових операцій: створення таблиць, вставка даних, оновлення, видалення, вибірка з фільтрацією тощо. Завдяки підтримці транзакцій, контекстних менеджерів і захищених параметризованих запитів вона також дозволяє дотримуватись високих стандартів безпеки та надійності.

У контексті дипломного проєкту використання `aiosqlite` є доцільним вибором. По-перше, через легкість налаштування: не потрібна складна інфраструктура або зовнішній сервер бази даних. По-друге, `aiosqlite` забезпечує повну сумісність із асинхронною логікою Telegram-бота, що гарантує стабільну й передбачувану роботу навіть за умови зростання кількості користувачів.

2 ПРОЄКТУВАННЯ СИСТЕМИ

2.1 Призначення системи і вимоги до неї

Розроблений Telegram-бот має на меті автоматизувати та спростити процес отримання студентами інформації про свої оцінки. Традиційна модель доступу до навчальних результатів зазвичай передбачає звернення до викладача або деканату. Telegram-бот, як зручний, швидкий та доступний інтерфейс, дає змогу організувати цей процес значно ефективніше, використовуючи звичний для більшості студентів месенджер.

Основною функціональністю бота є можливість студенту, після проходження базової авторизації, отримати перелік своїх оцінок за предметами, а також інші пов'язані з навчанням дані. З боку адміністрації (наприклад, викладачів або відповідального за базу даних користувача) передбачена можливість додавання, редагування або видалення інформації через окремий набір команд. Уся взаємодія з ботом здійснюється через інтуїтивно зрозумілий інтерфейс, побудований на базі бібліотеки Aiogram, яка забезпечує асинхронну обробку повідомлень Telegram API.

Для зберігання даних використовується SQLite, що є зручним рішенням для проєктів малого масштабу, оскільки не вимагає окремого серверного середовища. У зв'язку з тим, що бот побудований за асинхронною архітектурою, було обрано бібліотеку Aiosqlite, яка дозволяє працювати з базою даних у неблокуючому режимі, забезпечуючи плавну та стабільну роботу всієї системи. При цьому для побудови логіки моделей, збереження та обробки інформації застосовується ORM-інструмент SQLAlchemy, що полегшує взаємодію з базою, підвищує безпеку та дозволяє в майбутньому легко змінити СУБД без значних змін у коді.

Розглянемо вимоги до бота. Бот має бути інтуїтивно зрозумілим для студентів та викладачів, без потреби у додаткових інструкціях. Завдяки використанню асинхронного програмування, бот повинен залишатися швидким і чутливим до запитів навіть за одночасного використання кількох користувачами. Доступ до інформації має бути обмежений авторизованими користувачами, а вся передача даних – захищеною від випадкового витоку або стороннього доступу. Архітектура рішення повинна дозволяти подальше масштабування – зокрема, міграцію на іншу СУБД або додавання нових функцій (наприклад, повідомлення про нові оцінки, статистика, графіки успішності тощо).

Telegram-бот має працювати на звичайному сервері з Python-оточенням без потреби у складних залежностях чи додатковому адмініструванні.

Загалом, реалізація цього бота є відповіддю на реальну потребу в покращенні цифрової комунікації між студентами та освітніми установами. Використання сучасних інструментів розробки забезпечує надійну, легку в підтримці та ефективну систему, яка може стати основою для подальших удосконалень.

2.2 Загальна структура системи та її складові

Telegram-бот для отримання студентами оцінок є програмною системою, що складається з кількох взаємопов'язаних компонентів, кожен з яких виконує чітко визначену функцію. Завдяки модульній структурі забезпечується логічна організація коду, спрощується розробка, тестування та подальша підтримка системи. Функціонально можна виокремити такі модулі системи:

- інтерфейс користувача (Telegram-інтерфейс);
- обробник логіки;
- база даних;
- модулі конфігурації та ініціалізації.

Основною точкою взаємодії користувача з системою є Telegram-інтерфейс – бот, який працює у месенджері Telegram. Через нього студенти та адміністратори (наприклад, викладачі) надсилають запити у вигляді повідомлень або команд, на які бот відповідає відповідно до своєї логіки. Взаємодія з Telegram API реалізована за допомогою бібліотеки Aiogram, що дозволяє асинхронно обробляти запити, використовуючи структуру диспетчерів, хендлерів та станів[14].

Обробник логіки відповідає за реалізацію усіх правил, що визначають поведінку бота. Сюди входять: Обробка команд користувачів (/start, /додати_оцінку, /переглянути_результати тощо), перевірка прав доступу (розрізнення студентів і адміністраторів), вивід інформації у зручному для користувача форматі.

База даних відповідає за зберігання всієї інформації – списків студентів, оцінок, предметів тощо – здійснюється у базі даних SQLite. Доступ до бази реалізовано через бібліотеку aioredis, яка забезпечує неблокуючі запити до бази в асинхронному середовищі. Для зручного та безпечного опису структури даних, а також для роботи з таблицями використовується ORM SQLAlchemy. Завдяки ORM доступ до інформації відбувається не через "сирі" SQL-запити, а через об'єкти та моделі, що підвищує читаємість і безпечність коду.

Модулі конфігурації та ініціалізації відповідають за зберігання токена бота, налаштувань бази даних, імен таблиць, списків доступних команд тощо. Це дозволяє централізовано змінювати параметри системи без потреби втручатися в основну логіку програми. Також у структурі проєкту передбачений стартовий скрипт (`main.py`), який відповідає за запуск бота, ініціалізацію диспетчера, реєстрацію хендлерів, підключення до бази даних і створення таблиць.

2.3 Опис функціонування системи

Система Telegram-бота реалізована як асинхронний застосунок, що працює на основі взаємодії з Telegram API, обробки запитів користувачів, роботи з базою даних та формування відповідей. Її робота організована послідовно, з поділом на окремі етапи, що забезпечують цілісне та надійне функціонування бота. Функціонально можна виділити такі етапи функціонування системи:

- ініціалізація системи;
- взаємодія з користувачем;
- ідентифікація користувача;
- обробка запитів і доступ до бази даних;
- формування і відправка відповіді;
- підтримка декількох сесій.

Ініціалізація основних компонентів відбувається після запуску бота: підключення до бази даних SQLite за допомогою бібліотеки `aiosqlite`, завантаження конфігураційних налаштувань, підключення до Telegram API через бібліотеку `aiogram`, а також реєстрація обробників команд і повідомлень.

Взаємодія з користувачем відбувається коли він розпочинаючи роботу з ботом, надає доступ до свого мобільного телефону до якого прив'язаний його аккаунт Telegram. У відповідь бот виводить меню дій, якщо номер телефону до цього був доданий у базу даних. Подальші запити здійснюються через введення

команд або кнопок, що відповідають передбаченим сценаріям: перегляд оцінок, додавання нових записів тощо.

Ідентифікація користувача відбувається, коли при першій взаємодії бот запитує доступ до номера телефону, і якщо запис з таким номером уже є в базі даних, то до цього запису прив'язується Telegram ID користувача, який здійснив цей запит. Після успішної ідентифікації для кожної категорії користувачів стає доступним відповідний набір функцій.

Обробка запитів і доступ до бази даних відбувається в залежності від введеної команди. Після її введення бот викликає відповідну функцію, яка формує запит до бази даних. Дані зберігаються у SQLite, а доступ до них реалізується через SQLAlchemy у поєднанні з aiomysqlite, що забезпечує асинхронну взаємодію з БД без блокування основного потоку виконання.

Формування та відправка відповіді відбувається, коли результати обробки запиту, незалежно від його типу, повертаються користувачу у вигляді повідомлення в Telegram. У випадку успішної операції це може бути перелік оцінок, підтвердження додавання чи редагування даних. У разі помилки бот інформує про це користувача.

Підтримка декількох сесій проявляється в тому, що бот здатен обробляти запити від кількох користувачів одночасно завдяки асинхронній архітектурі. Aiogram забезпечує обробку подій без блокування, а aiomysqlite дозволяє виконувати запити до бази даних у фоновому режимі, що робить систему ефективною навіть при зростанні кількості активних користувачів.

У підсумку, Telegram-бот функціонує як посередник між користувачем і базою даних, забезпечуючи простий та швидкий доступ до навчальної інформації, мінімізуючи навантаження на адміністративний персонал та оптимізуючи комунікацію в освітньому процесі (рис. 2.1).

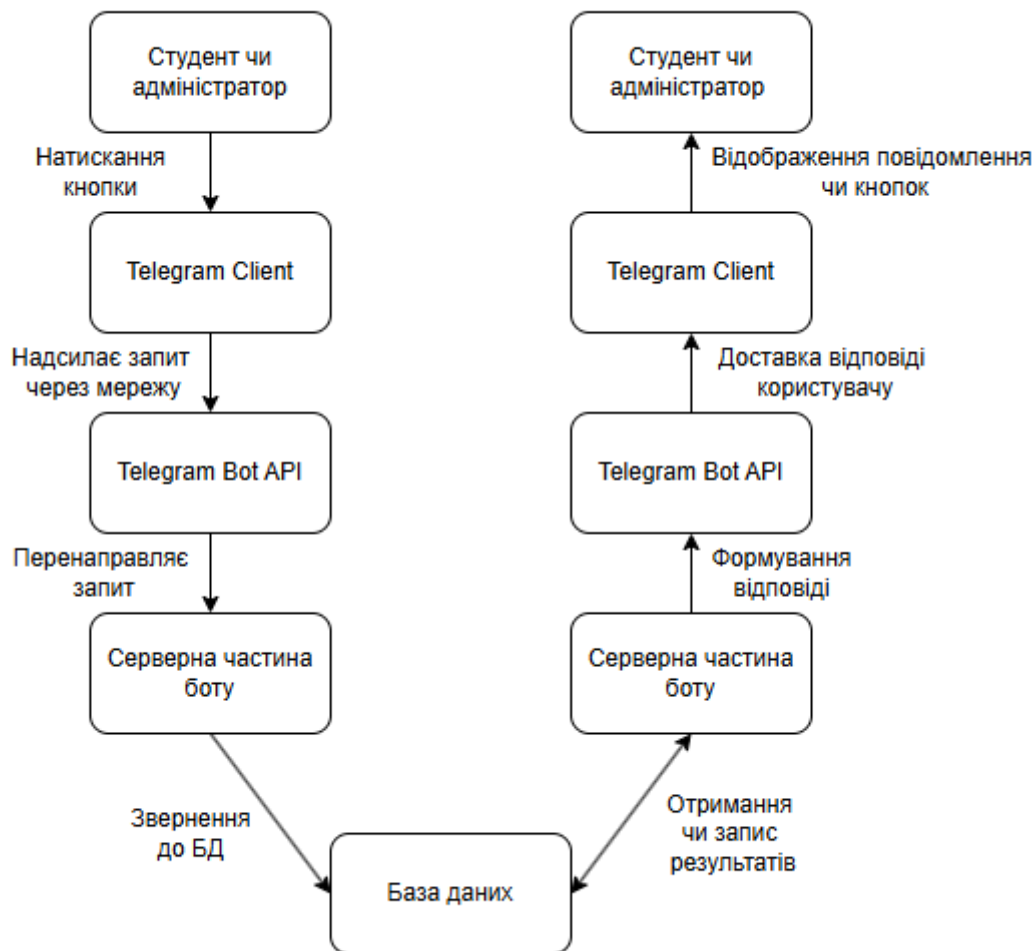


Рисунок 2.1 – Діаграма взаємодії користувача з ботом

3 РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Встановлення необхідного ПЗ та реєстрація бота

3.1.1 Завантаження Python та необхідних бібліотек

Для коректної роботи Telegram-бота та його інтеграції з базою даних і Telegram API необхідно виконати ряд кроків із встановлення й налаштування програмного забезпечення.

Спочатку треба встановити мову розробки Python. Для цього завантажується дистрибутив з офіційного сайту [15]. Після встановлення можна перевірити коректність інсталяції командою в консолі – «python --version»

Після цього треба встановити бібліотеки, які були обрані у розділах вище (aiogram, SQLAlchemy, aiomysqlite). Встановлення бібліотек здійснюється командою – «pip install aiogram sqlalchemy aiomysqlite».

3.1.2 Реєстрація у BotFather

Після вибору та встановлення всіх інструментів та бібліотек потрібно переходити до розробки. Першочергово для цього потрібно зареєструвати бота і отримати його унікальний токен. Для цього в месенджері Telegram передбачено зручний інструмент, який сам і є ботом, він має назву «BotFather». Щоб почати з ним роботу потрібно знайти його в пошуковій панелі, туди потрібно ввести @botfather (рис. 3.1).

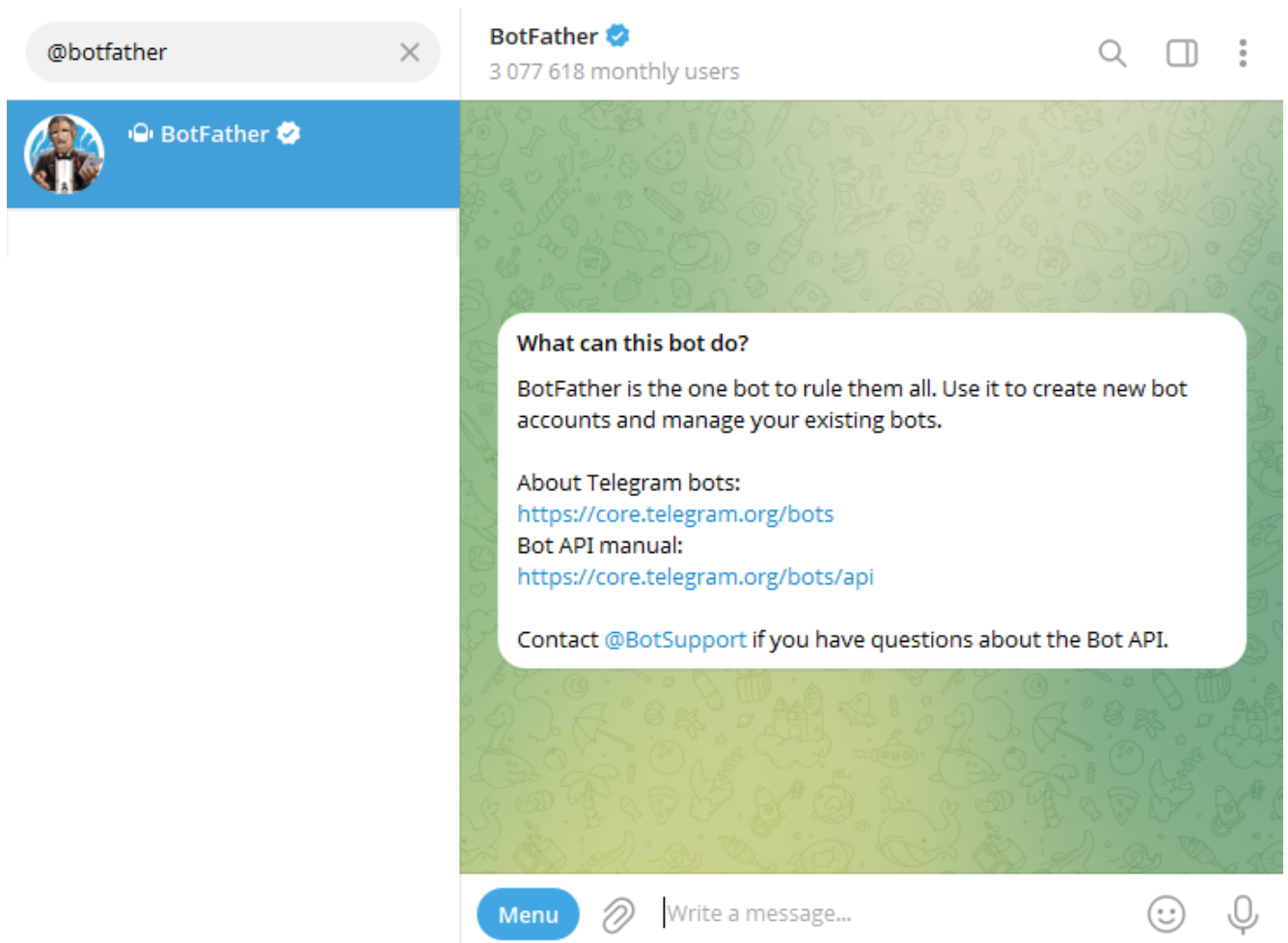


Рисунок 3.1 – BotFather, бот для створення і управління особистими ботами.

Після цього пишемо команду «/start» для отримання початкових даних для роботи (рис. 3.2).

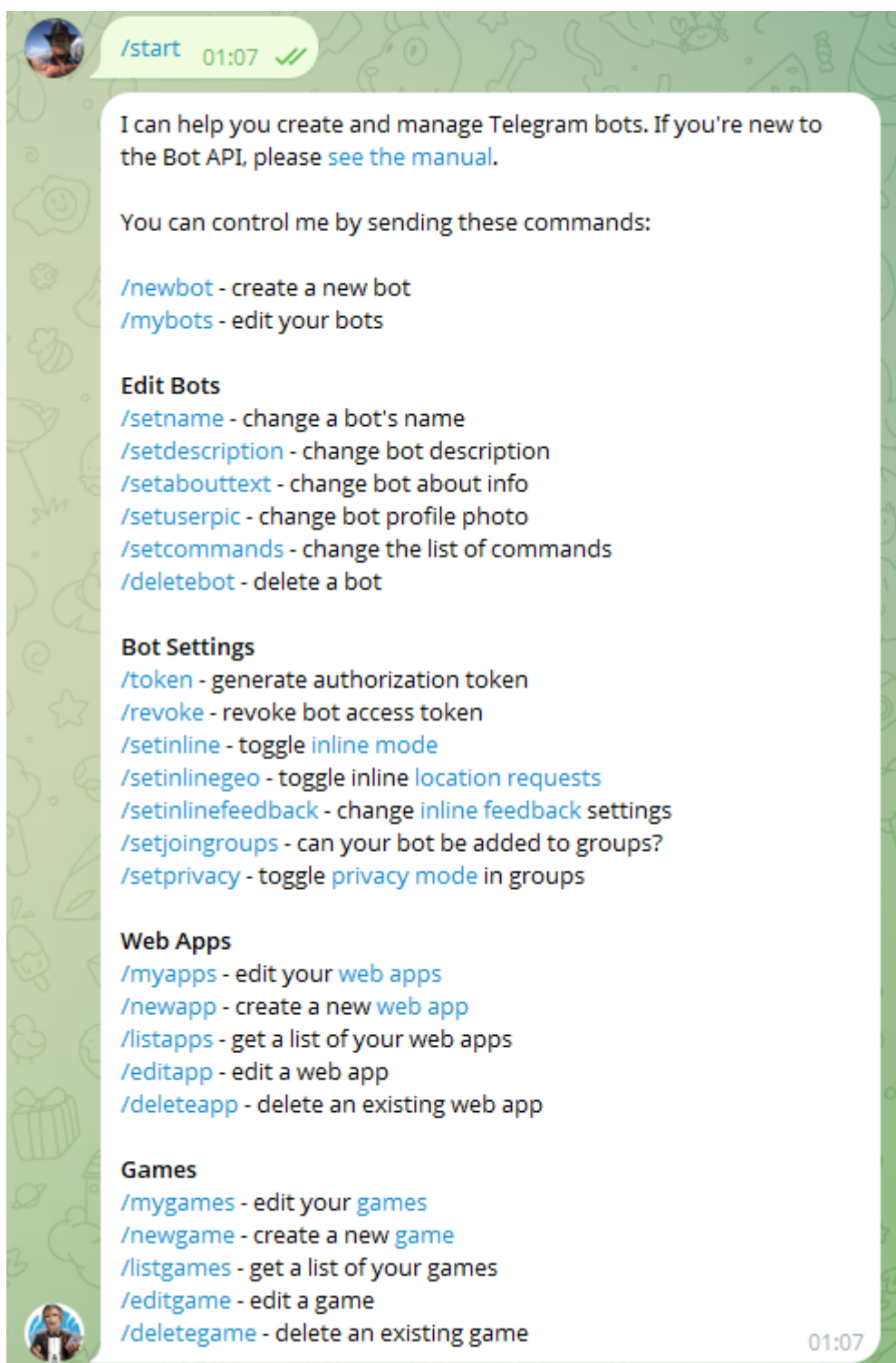


Рисунок 3.2 – Отриманні команди після команди «/start»

Серед отриманих команд від бота є «/newbot», вона потрібна для створення власного унікального бота. Після її введення BotFather просить ввести ім'я та

телеграм юзернейм. Вводимо їх та отримуємо згенерований токен, який потрібен для підключення серверної частини до Telegram Bot API (рис. 3.3).

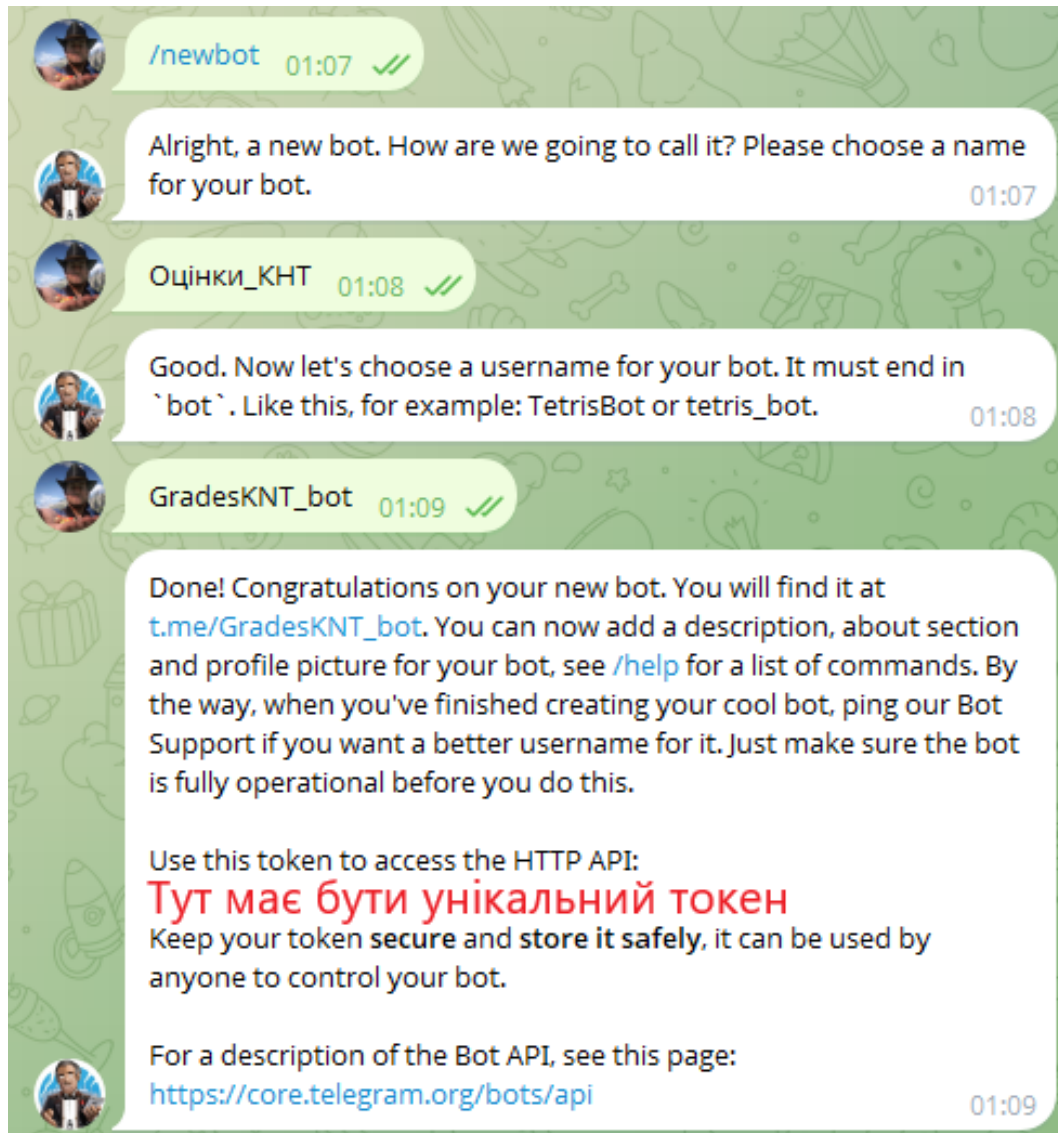
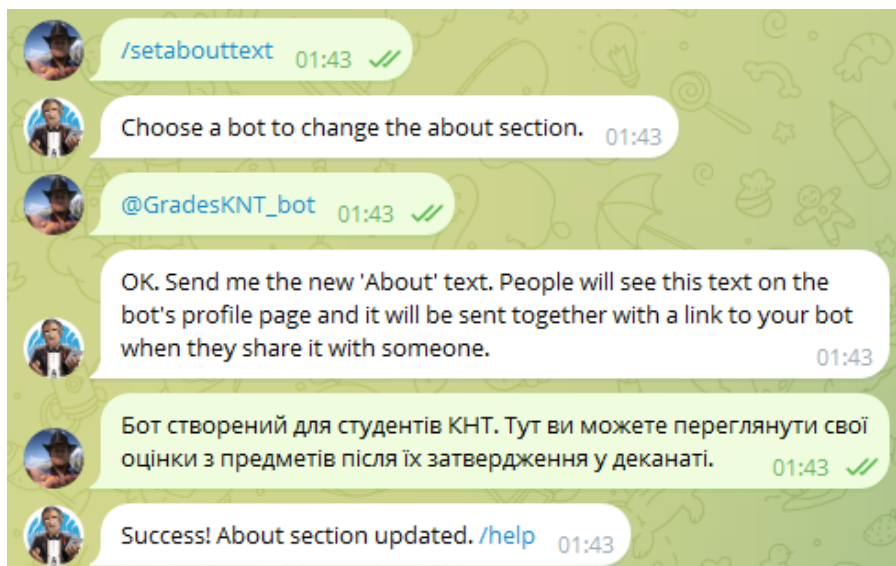


Рисунок 3.3 – Результати виконання команди «/newbot»

Після цього треба зробити мінімальне оформлення. Надати опис та зображення профіля боту, а також опис при першому відкритті боту. Відповідно до цього використовуємо команди «/setabouttext» (рис.3.4), «/setuserpic» (рис.3.5) та «/setdescription» (рис.3.6).

Рисунок 3.4 – Виконання команди «`/setabouttext`»Рисунок 3.5 – Виконання команди «`/setuserpic`»

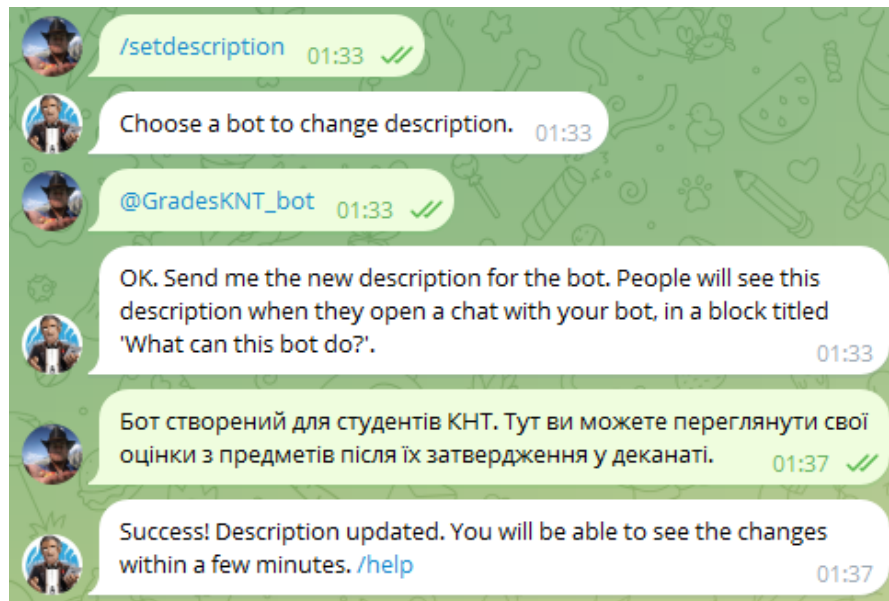


Рисунок 3.6 – Виконання команди «/setdescription»

Тепер все готово для того, щоб підключити до створеного бота серверну частину та наповнити його функціоналом. Але варто ще звернути увагу на команду «/revoke», яка скасовує токен бота та надає новий, вона може знадобитися якщо є підозра на компрометацію токена.

3.2 Опис структури коду, взаємодії та модульності

Для забезпечення підтримуваності, масштабованості й зручності розвитку проєкту, програмна система була розділена на окремі логічні модулі. Такий підхід дає змогу легко орієнтуватися у коді, швидко знаходити й змінювати необхідні частини програми, а також розширювати функціонал у майбутньому. Уся логіка роботи Telegram-бота поділена на такі основні модулі:

- models.py;
- requests.py;
- keyboards.py;
- handlers.py.

`models.py` – є основою для всієї структури даних, описує структуру бази даних за допомогою SQLAlchemy, містить класи основних сутностей та їх взаємозв'язки.

`requests.py` – містить функції для виконання основних CRUD-операцій з базою даних (створення, читання, оновлення, видалення записів), а також логіку валідації та перевірки даних. Використовує моделі з `models.py` для безпосередньої взаємодії з базою даних.

`keyboards.py` – реалізує генерацію динамічних inline-клавіатур для Telegram-бота: меню, списки груп, студентів, предметів, семестрів тощо. Викликає функції з `requests.py` для отримання актуальних даних, на основі яких будує клавіатури.

`handlers.py` – виступає центральним координатором, що отримує події від користувача, взаємодіє із клавіатурами через `keyboards.py` та працює з базою даних через `requests.py`. Містить обробники (handler-и) для роботи з Telegram API, організовані відповідно до сценаріїв роботи користувача (реєстрація, вибір предметів, додавання оцінок, перегляд і т.п.).

3.3 Опис структури бази даних

Розроблена система для автоматизації обліку оцінок студентів базується на реляційній базі даних, реалізованій із використанням ORM-бібліотеки SQLAlchemy. Для зберігання даних застосовується легка та проста у налаштуванні СУБД SQLite, яка ідеально підходить для невеликих проєктів. Структура бази даних містить такі ключові сутності (рис. 3.7):

- користувач (`users`);
- група (`groups`);
- предмет (`subjects`);
- оцінка (`grades`);
- таблиця студент-предмет (`student_subject`).

Таблиця users зберігає інформацію про студентів та викладачів. Кожен користувач має унікальний ідентифікатор, Telegram User ID, номер телефону, ПІБ, а також позначку, чи є він викладачем. Для студентів додатково зберігається посилання на групу. Основні поля : «id», «tg_user_id», «phone_number», «full_name», «is_teacher», «group_id».

Таблиця groups зберігає інформацію про навчальні групи. Група має унікальний ідентифікатор та назву. Основні поля: «id», «title».

Таблиця subjects містить інформацію про навчальні дисципліни. Кожен предмет має унікальний ідентифікатор, назву та номер семестру, у якому він викладається. Основні поля: «id», «title», «semester».

Таблиця grades зберігає інформацію про виставлені оцінки. Кожен запис містить посилання на користувача, предмет, числове значення оцінки, коментар та дату створення. Основні поля: «id», «score», «comment», «created_at», «user_id», «subject_id».

Проміжна таблиця student_subject реалізує зв'язок "багато до багатьох" між студентами та предметами. Це дозволяє гнучко призначати предмети окремим студентам, а не лише цілим групам. Основні поля: «student_id», «subject_id».

Взаємозв'язки реалізовані таким чином, що один студент може належати лише до однієї групи, а група – до багатьох студентів, студент може мати багато предметів, та само як і один предмет може бути призначений багатьом студентів, один студент може мати багато оцінок з різних предметів, але лише одну з кожного предмета.

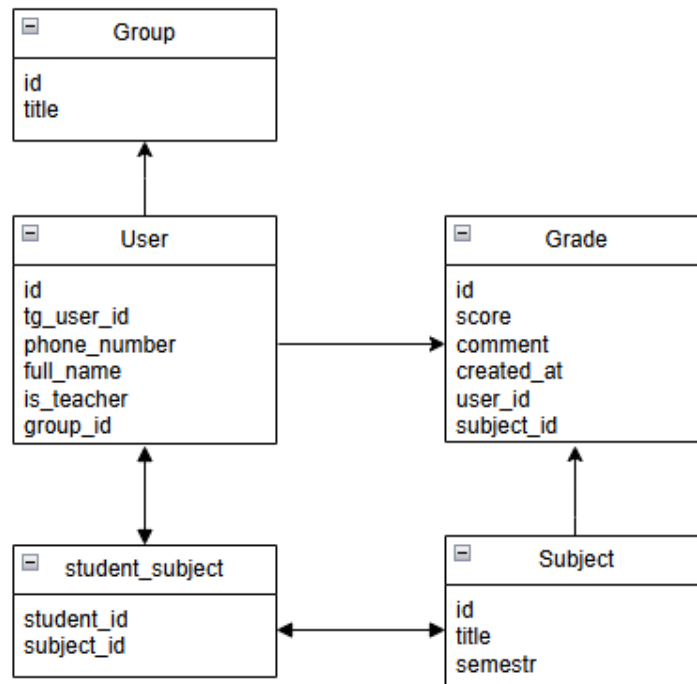


Рисунок 3.7 – UML-діаграма структури БД

3.4 Реалізація серверної частини

Реалізація серверної частини системи передбачає розробку програмної логіки, яка забезпечує взаємодію між Telegram-ботом, базою даних та зовнішніми користувачами (студентами та викладачами). Основними компонентами серверної частини виступають моделі даних, асинхронний доступ до бази даних, а також реалізація основних функціональних сценаріїв (CRUD-операцій) для роботи з основними об'єктами системи.

3.4.1 Створення моделей даних

Моделі створені так, що кожна містить визначення відповідних атрибутів, типів даних та взаємозв'язків із іншими моделями. Наприклад, для моделі user визначено відношення до групи (group), предметів (subjects) та оцінок (grades) (лістинг 3.1). Для кожної моделі також передбачено валідацію полів (наприклад, діапазон номеру семестру).

Лістинг 3.1 – Реалізація моделі user

```
class User(Base):
    __tablename__ = 'users'

    id: Mapped[int] = mapped_column(primary_key=True)
    tg_user_id: Mapped[int] = mapped_column(BigInteger,
    nullable=True)
    phone_number: Mapped[str] = mapped_column(String, unique=True)
    full_name: Mapped[str] = mapped_column(String(150))
    is_teacher: Mapped[bool] = mapped_column(default=False)

    subjects: Mapped[list["Subject"]] = relationship(
        secondary=student_subject, back_populates="students",
        passive_deletes=True
    )
    group_id: Mapped[int | None] = mapped_column(
        ForeignKey("groups.id"), nullable=True
    )
    group: Mapped["Group"] = relationship(
        back_populates="students"
    )
    grades: Mapped[list["Grade"]] = relationship(
        back_populates='user', cascade='all, delete-orphan'
    )
```

3.4.2 Ініціалізація бази даних

Для ініціалізації бази даних використовується асинхронна функція, яка створює всі необхідні таблиці згідно зі схемою, визначеною у моделях (лістинг 3.2). У результаті виконання цієї функції у СУБД SQLite створюються всі таблиці для подальшої роботи.

Лістинг 3.2 – Функція, що виконує ініціалізацію бази даних

```
async def async_main():
    async with engine.begin() as conn:
        await conn.run_sync(Base.metadata.create_all)
```

3.4.3 Реалізація CRUD-операцій

Важливим етапом у практичній реалізації програмної системи є забезпечення операцій створення, читання, оновлення та видалення даних. У розробленій системі CRUD-операції дозволяють ефективно керувати основними сутностями:

користувачами, групами, предметами, а також оцінками, які виставляються студентам. Для забезпечення зручності розробки, спрощення підтримки коду та підвищення його читабельності, вся логіка взаємодії з базою даних була винесена у окремий програмний модуль – requests.py. Цей модуль містить набір функцій, що реалізують основні операції роботи з базою даних, використовуючи асинхронні запити SQLAlchemy. Для прикладу розглянемо детальніше реалізацію сутності group, у якій присутні операції читання (лістинг 3.3), створення (лістинг 3.4) та видалення (лістинг 3.5). Робота з групами забезпечує додавання нових навчальних груп та їхнє видалення за потреби. Функції забезпечують перевірку унікальності назв груп, а також обробку винятків, якщо групу не знайдено або виникає спроба додати вже наявну групу.

Лістинг 3.3 – Отримання груп

```
async def get_groups():
    async with async_session() as session:
        return await session.scalars(select(Group))
```

Лістинг 3.4 – Створення групи

```
async def set_group(active_user_tg_id: int, group: str):
    async with async_session() as session:
        group_in_db = await session.scalar(select(Group).where(
            Group.title==group)
        )
        user = await get_user_with_session(
            active_user_tg_id, session
        )

        if not group_in_db and user.is_teacher:
            session.add(Group(title=group))
            await session.commit()

        else:
            raise PermissionError('Ви не є викладачем або запис вже існує')
```

Лістинг 3.5 – Видалення групи

```
async def del_group(
    active_user_tg_id: int, group_id
):
```

```

async with async_session() as session:
    user = await get_user_with_session(active_user_tg_id, session)
    group = await session.scalar(
        select(Group).where(Group.id==group_id)
    )

    if group is None:
        raise ValueError('Групу не знайдено')
    if user.is_teacher:
        await session.delete(group)

        await session.commit()
    else:
        raise PermissionError('Користувач не має статус викладача')

```

3.5 Реалізація логіки Telegram-бота

Telegram-бот є основним інтерфейсом взаємодії кінцевого користувача із системою. Завдяки використанню бібліотеки aiogram, було реалізовано покрокову FSM-логіку для різних сценаріїв, а також інтегровано динамічні клавіатури для зручної роботи з об'єктами бази даних (групи, студенти, предмети, оцінки). Бот підтримує різні сценарії взаємодії, орієнтовані на два типи користувачів: студента та викладача. Основні функціональні можливості:

- авторизація користувачів;
- внесення авторизаційних даних студента адміністратором;
- перегляд студентом своїх оцінок за семестрами;
- перегляд оцінок студентів для викладача ;
- додавання, зміна та видалення оцінок;
- керування групами та предметами;
- призначення студентам предметів.

3.5.1 FSM-логіка покрокової взаємодії

FSM – це математична модель, що дозволяє описати послідовність кроків (станів), через які проходить система або користувач. У Telegram-ботах FSM дає змогу запам'ятати, на якому етапі взаємодії перебуває кожен користувач, та

відповідно реагувати на його подальші дії. У проєкті для складних сценаріїв реалізовано окремий клас станів (StatesGroup), що дозволяє покроково опрацьовувати введені користувачем дані, гарантуючи правильний порядок виконання операцій. Наприклад, при авторизації, коли користувач надає доступ до свого телефону, спочатку бот перевіряє, чи існує користувач у базі, якщо так, то надає доступ до інформації, якщо ні, то виводиться помилка.

3.5.2 Основні хендлери та їх функціонал

Усі сценарії реалізовано через обробники (handler-и), які реагують на певні дії користувача – команди, повідомлення, натискання кнопок (таблиця 3.1).

Таблиця 3.1 – Основні хендлери

Назва handler-a	Призначення
handle_start_command	Запуск бота та перевірка реєстрації користувача
handle_register_name, handle_register_group	Покрокова реєстрація
handle_all_student_grade	Перегляд студентом власних оцінок
handle_teacher_subject	Керування предметами
handler_choice_semester	Вибір семестру
handle_save_new_subject	Додавання нового предмета
handle_del_subject	Видалення предмета
handle_teacher_group	Керування групами
handle_save_new_group	Додавання нової групи
handle_del_group	Видалення групи
select_grade_group, select_grade_student, select_grade_subject	Вибір групи, студента, предмета для роботи з оцінками
input_grade_value, handle_grade_input, handler_comment_input	Додавання, редагування та коментування оцінок
handle_choose_group, handle_choose_subject, handle_show_student_grades	Перегляд викладачем оцінок студентів
handle_add_student_subject, handle_select_student, handle_select_subject, handle_save_data	Призначення студенту нових предметів або видалення предметів у студента

3.5.3 Динамічні inline-клавіатури

Для кожного етапу сценарію (наприклад, вибір групи, студента, предмета, семестру) бот генерує inline-клавіатури, що значно спрощує навігацію для користувача та мінімізує кількість ручного введення. Клавіатури формуються

динамічно на основі даних із бази. Наприклад, при виборі групи бот отримує актуальний список груп із бази даних і будує клавіатуру саме з тих груп, які є на поточний момент (лістинг 3.6). Якщо якась група буде видалена, кнопка для неї автоматично зникне з клавіатури. Якщо додана нова, вона з'явиться для вибору.

Лістинг 3.6 – Приклад динамічного формування клавіатури

```
async def groups(include_skip_option=True):
    all_groups = await get_groups()
    keyboard = InlineKeyboardBuilder()
    for group in all_groups:
        keyboard.add(InlineKeyboardButton(
            text=group.title,
            callback_data=f'group:{group.id}')
        )
    if include_skip_option:
        keyboard.add(InlineKeyboardButton(
            text='Не вказувати групу',
            callback_data='group:None')
        )
    else:
        keyboard.add(InlineKeyboardButton(
            text='Повернутись до меню',
            callback_data='go_back')
        )
    return keyboard.adjust(2).as_markup()
```

3.6 Реалізація безпеки та розмежування доступу до інформації

Одним із найважливіших завдань при створенні інформаційних систем, особливо тих, що працюють з персональними даними (наприклад, оцінками студентів), є забезпечення надійного захисту інформації. Розроблений Telegram-бот включає в себе набір заходів, які гарантують безпечність і коректність доступу до інформації для різних категорій користувачів.

Telegram-бот використовує авторизацію за номером, а також вбудований механізм ідентифікації Telegram, а саме унікальний Telegram User ID, який гарантовано є унікальним для кожного користувача і недоступний стороннім особам. Під час першого запуску бота користувач надає Telegram доступ до свого

номеру телефону, і якщо запис з таким номером є в базі даних, то в цей запис вже додається його Telegram ID, завдяки цьому вдалось реалізувати захист від доступу студентів до інформації, яка їм не належить. Кожен запит до бази даних, пов'язаний з отриманням оцінок студента, містить перевірку відповідності Telegram User ID, з якого надійшов запит та ідентифікатору студента в базі даних (лістинг 3.6). Це забезпечує гарантію, що студент отримує доступ лише до своїх даних, а також створює деяку автоматичність та економить час, адже після початкової авторизації користувач більше не потребує додаткових дій, наприклад введення паролів і т.п.

Ще в проєкті, звісно, є розмежування ролей користувачів. Реалізовано два стани – студент та адміністратор-викладач. Студент має доступ виключно до перегляду власних оцінок, адміністратор ж може здійснювати всі операції з базою даних, які вже були описані у розділах вище. Розмежування реалізовано через окремий атрибут у базі даних – поле `is_teacher` у таблиці користувачів. Якщо значення цього поля встановлено як «1», користувач отримує права викладача. Для звичайних студентів це поле автоматично встановлюється як «0».

З міркувань безпеки та з метою контролю доступу адміністративні ролі не призначаються автоматично. Поле `is_teacher` у базі даних не може бути змінено користувачем самостійно через інтерфейс бота, а встановлюється виключно адміністратором системи шляхом прямого втручання у базу даних. Це забезпечує захист від випадкового або зловмисного привласнення розширених прав, або від витоку паролів, якщо б вони використовувались. В умовах практичного використання, коли адміністратор буде лише один, або невелика кількість таке рішення є цілком виправданим.

Також у проєкті увагу було приділено захисту від SQL-ін'єкцій – одного з найпоширеніших типів атак на інформаційні системи, що працюють із базами даних. SQL-ін'єкція виникає у випадках, коли користувач отримує змогу вставити довільний SQL-код у запит до бази даних через незахищене введення даних, що може призвести до витоку чи знищення інформації. Для протидії цій загрозі у розробленому Telegram-боті використовується ORM-бібліотека SQLAlchemy. Взаємодія з базою даних здійснюється через об'єктно-реляційне відображення, де

всі параметри, отримані від користувача, передаються як окремі аргументи функцій, а не підставляються напряму у текст SQL-запиту. Усі ці операції обробляються SQLAlchemy із використанням параметризованих запитів. Таким чином, навіть якщо користувач спробує ввести некоректні або небезпечні дані, вони не вплинуть на структуру запиту та не спричинять виконання шкідливого коду. Крім того, у системі відсутні «сирі» SQL-запити із ручним форматуванням рядків, які є головною причиною виникнення SQL-ін'єкцій. Асинхронна взаємодія з базою через SQLAlchemy та aiosqlite також передбачає використання безпечних механізмів передачі параметрів. Таким чином, застосування ORM та виключення ручних SQL-запитів у розробленій системі повністю захищає її від SQL-ін'єкцій, що відповідає сучасним вимогам до інформаційної безпеки програмних продуктів.

Лістинг 3.6 – Перевірка ID при виклику оцінок

```
async def get_grade(tg_user_id: int, subject_id) -> Grade:
    async with async_session() as session:
        user = await get_user(tg_user_id)
        grade = await session.scalar(select(Grade).where(and_(
            Grade.user_id==user.id, Grade.subject_id==subject_id
        )))
    return grade
```

4 РЕЗУЛЬТАТИ РОБОТИ

Для ілюстрації роботи розробленого Telegram-бота наводяться приклади основних екранів та сценаріїв взаємодії користувача з системою:

- авторизація користувача (рис. 4.1, рис 4.2, рис. 4.3);
- перегляд студентом власних оцінок (рис. 4.4, рис 4.5);
- надання прав викладачу та його можливості (рис. 4.6, рис 4.7);
- присвоєння оцінки та її перегляд (рис. 4.8, рис. 4.9);
- додання предмету (рис. 4.10, рис. 4.11);
- додання групи (рис. 4.12, рис. 4.13);

- присвоєння предметів студентам (рис. 4.14, рис 4.15, рис 4.16);
- видалення користувача (рис. 4.17, рис. 4.18);
- створення користувача (рис. 4.19, рис 4.20).

id	tg_user_id	phone_number	full_name	is_teacher	group_id
Фи...	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр
1	1133213	+380733333333	Авдеев Нікіта	0	1
3	213344	+380731111111	Бірюков Денис	0	1
4	1234553	+380732222222	Бугров Аріель	0	1
5	2146212	+380734444444	Восус Ілля	0	1
6	2135657682	+380735555555	Гетман Сергій	0	1
7	1231235	+380736666666	Гречка Нікіта	0	1
8	213578732	+380737777777	Дроздов ...	0	1
9	2579325565	+380738888888	Залозний ...	0	1
10	6344533321	+380739999999	Кодак Ірина	0	1
11	NULL	+380731145585	Ковальчук Артем	0	1

Рисунок 4.1 – Вигляд таблиці користувачів до моєї авторизації

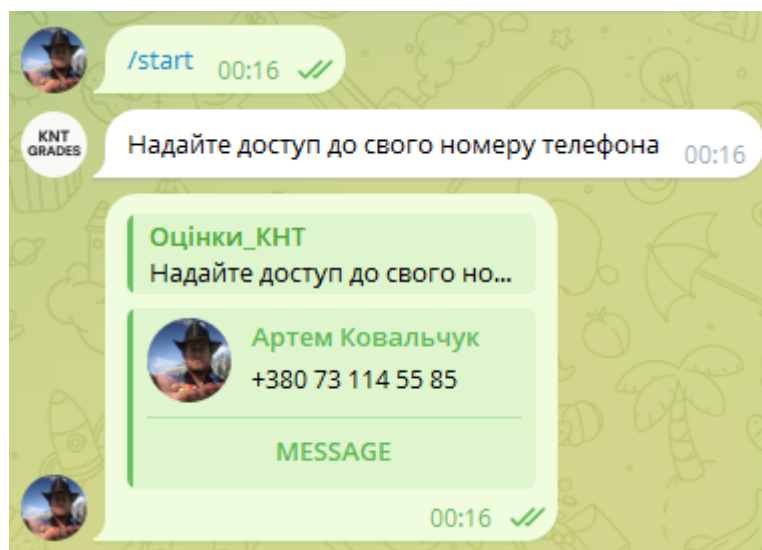


Рисунок 4.2 – Авторизація користувача за номером телефону

id	tg_user_id	phone_number	full_name	is_teacher	group_id
Фи...	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр
1	1133213	+380733333333	Авдеев Нікіта	0	1
3	213344	+380731111111	Бірюков Денис	0	1
4	1234553	+380732222222	Бугров Аріель	0	1
5	2146212	+380734444444	Восус Ілля	0	1
6	2135657682	+380735555555	Гетман Сергій	0	1
7	1231235	+380736666666	Гречка Нікіта	0	1
8	213578732	+380737777777	Дроздов Святослав	0	1
9	2579325565	+380738888888	Залозний Олександр	0	1
10	6344533321	+380739999999	Кодак Ірина	0	1
11	704614466	+380731145585	Ковальчук Артем	0	1

Рисунок 4.3 – Поява ID в таблиці після авторизації

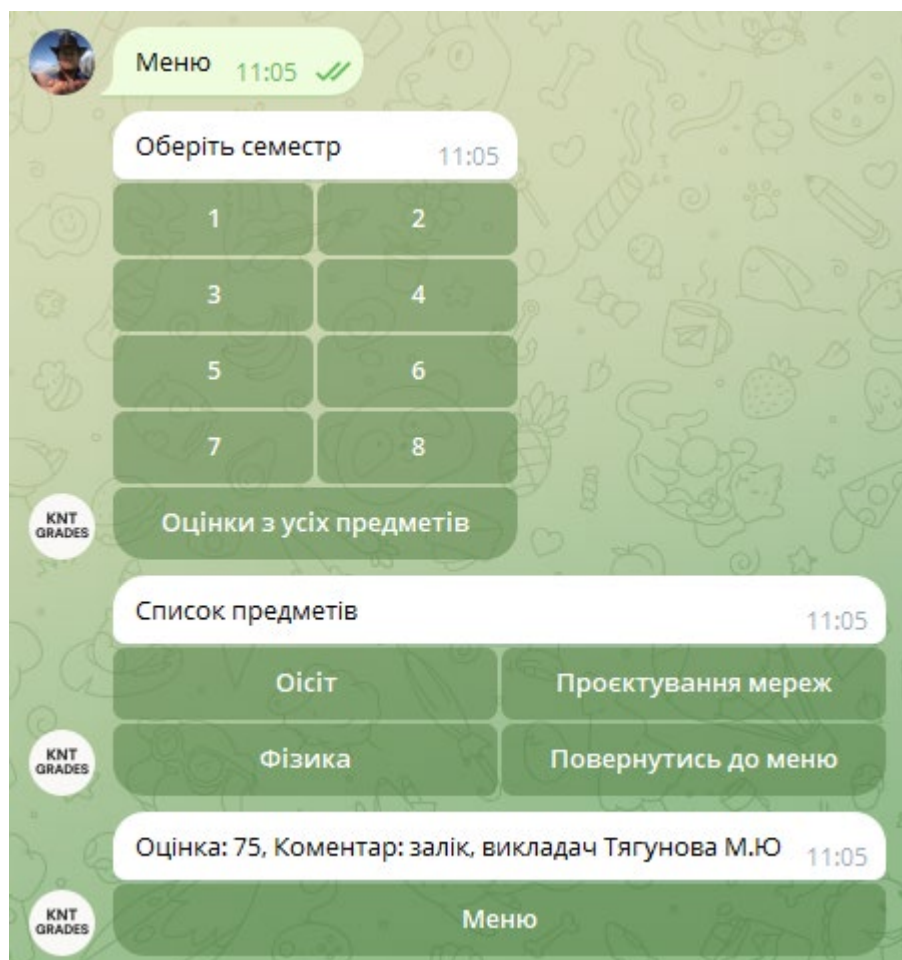


Рисунок 4.4 – Отримання студентом оцінки з одного предмету з вибором семестру

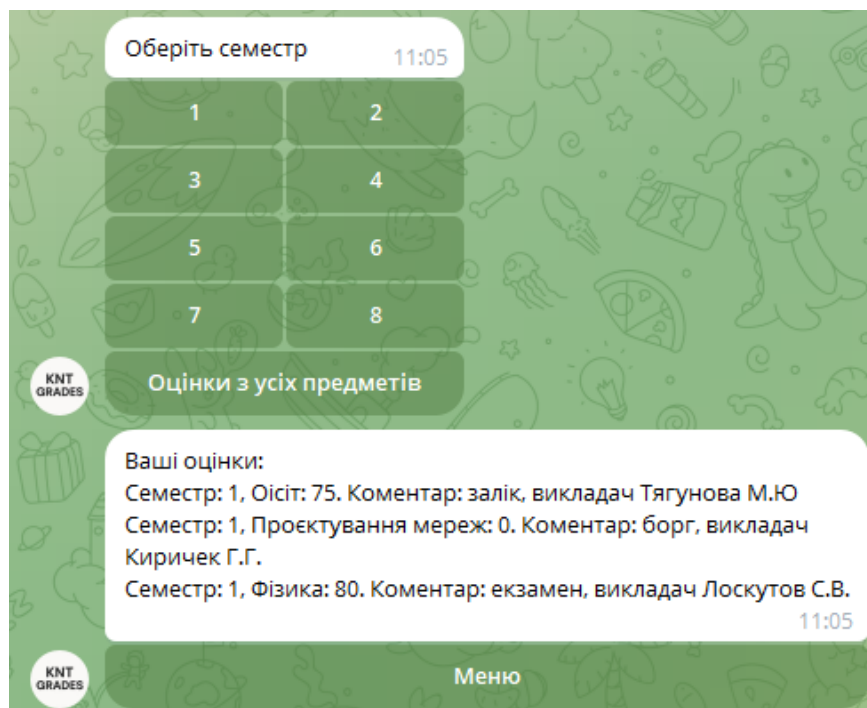


Рисунок 4.5 – Отримання студентом оцінок списком з усіх предметів

id	tg_user_id	phone_number	full_name	is_teacher	group_id
Фи...	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр
1	1133213	+380733333333	Авдеев Нікіта	0	1
3	213344	+380731111111	Бірюков Денис	0	1
4	1234553	+380732222222	Бугров Аріель	0	1
5	2146212	+380734444444	Восус Ілля	0	1
6	2135657682	+380735555555	Гетман Сергій	0	1
7	1231235	+380736666666	Гречка Нікіта	0	1
8	213578732	+380737777777	Дроздов Святослав	0	1
9	2579325565	+380738888888	Залозний Олександр	0	1
10	6344533321	+380739999999	Кодак Ірина	0	1
11	704614466	+380731145585	Ковальчук Артем	1	NULL

Рисунок 4.6 – Надання прав адміністратора вручну



Рисунок 4.7 – Меню адміністратора-викладача

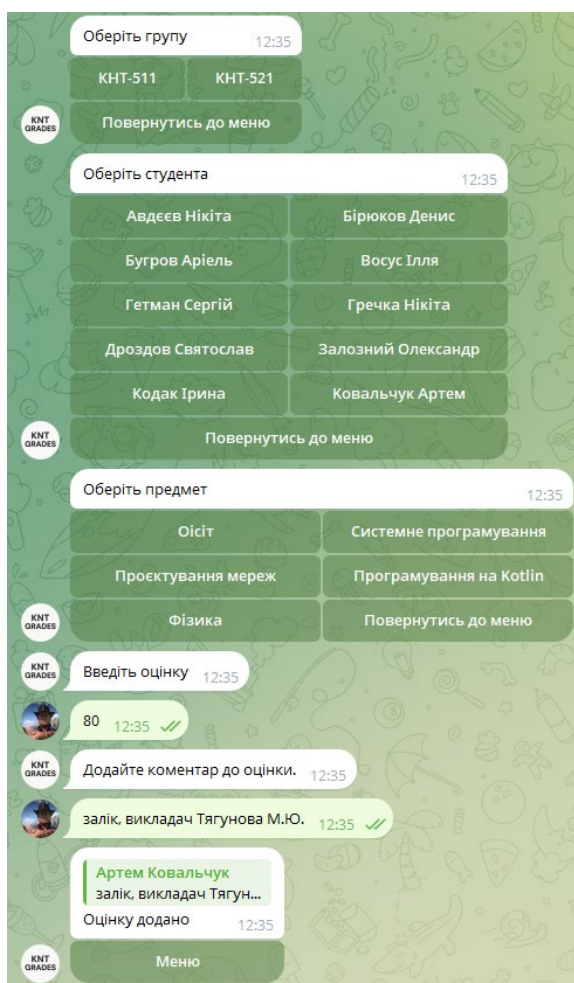


Рисунок 4.8 – Процес додання оцінки студенту Бірюков Денис

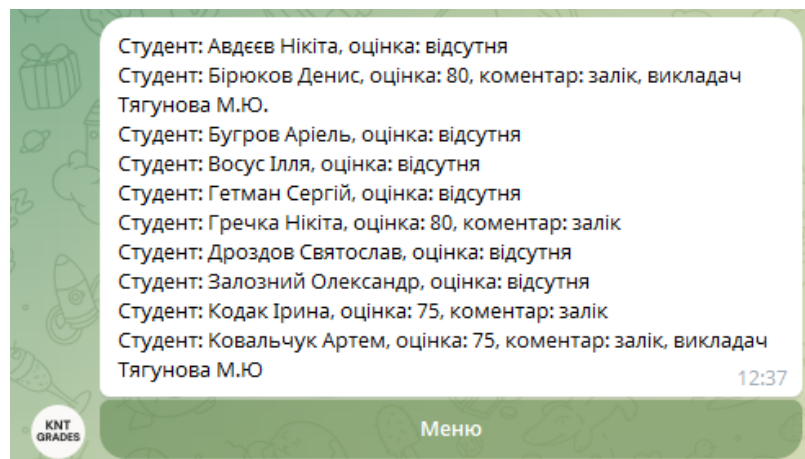


Рисунок 4.9 – Результат виконання при перегляді

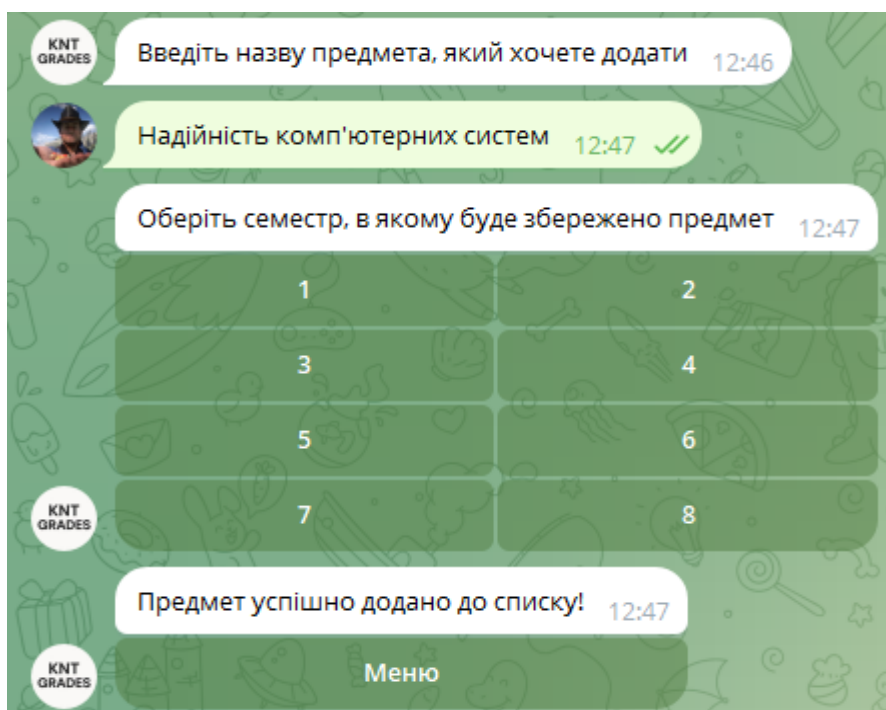


Рисунок 4.10 – Додання предмету у восьмий семестр

	id	title	semester
	Фи...	Фильтр	Фильтр
1	4	Оісіт	1
2	5	Системне програмування	1
3	7	Проектування мереж	1
4	8	Програмування на Kotlin	3
5	9	Фізика	1
6	10	Надійність комп'ютерних систем	8

Рисунок 4.11 – Поява предмета в базі даних

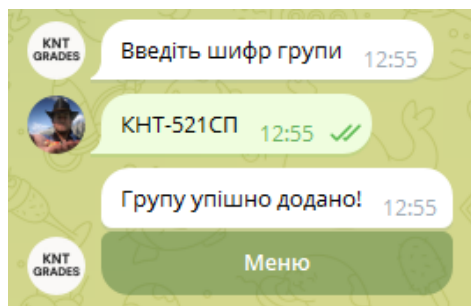


Рисунок 4.12 – Створення групи

	<u>id</u>	title
	Фи...	Фильтр
1	1	КНТ-511
2	2	КНТ-521
3	3	КНТ-521СП

Рисунок 4.13 – Результат створення групи

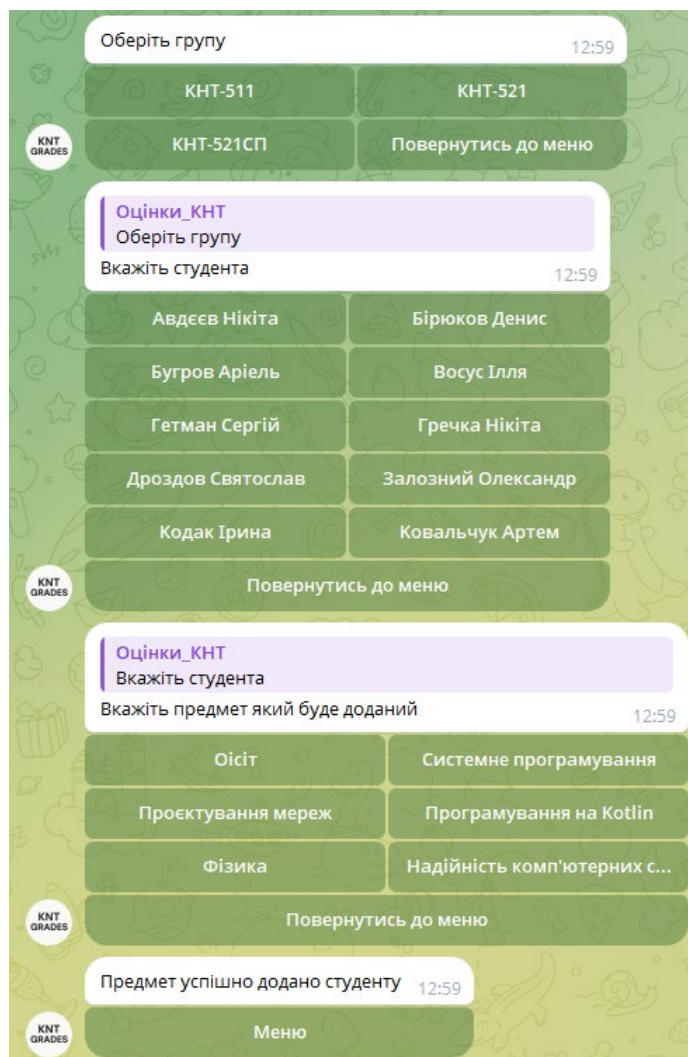


Рисунок 4.14 – Додання предмету окремому студенту

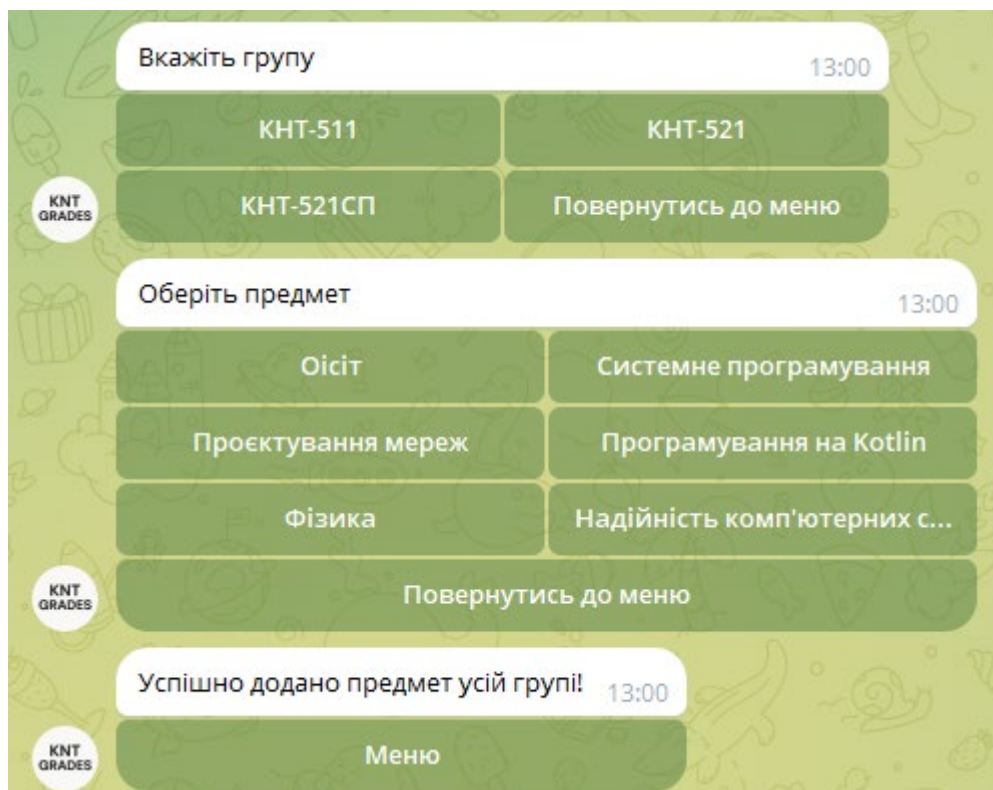


Рисунок 4.15 – Додання предмету всій групі

15	3	4
16	1	10
17	3	10
18	4	10
19	5	10
20	6	10
21	7	10
22	8	10
23	9	10
24	10	10
25	11	10

Рисунок 4.16 – Результат додавання предмету студенту та групі. Перший стовбець – student_id, другий – subject_id

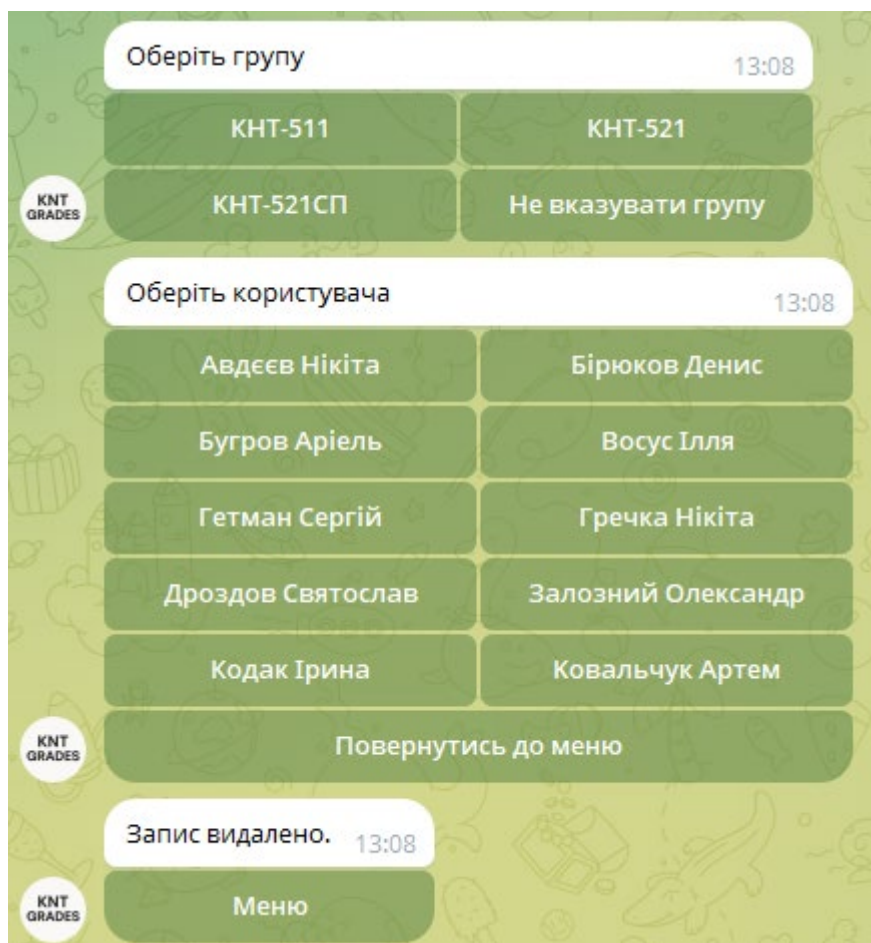


Рисунок 4.17 – Видалення студента Бугров Аріель

id	tg_user_id	phone_number	full_name	is_teacher	group_id
Фи...	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр
1	1133213	+380733333333	Авдєєв Нікіта	0	1
3	213344	+380731111111	Бірюков Денис	0	1
5	2146212	+380734444444	Восус Ілля	0	1
6	2135657682	+380735555555	Гетман Сергій	0	1
7	1231235	+380736666666	Гречка Нікіта	0	1
8	213578732	+380737777777	Дроздов Святослав	0	1
9	2579325565	+380738888888	Залозний Олександр	0	1
10	6344533321	+380739999999	Кодак Ірина	0	1
11	704614466	+380731145585	Ковальчук Артем	1	NULL

Рисунок 4.18 – Результат видалення

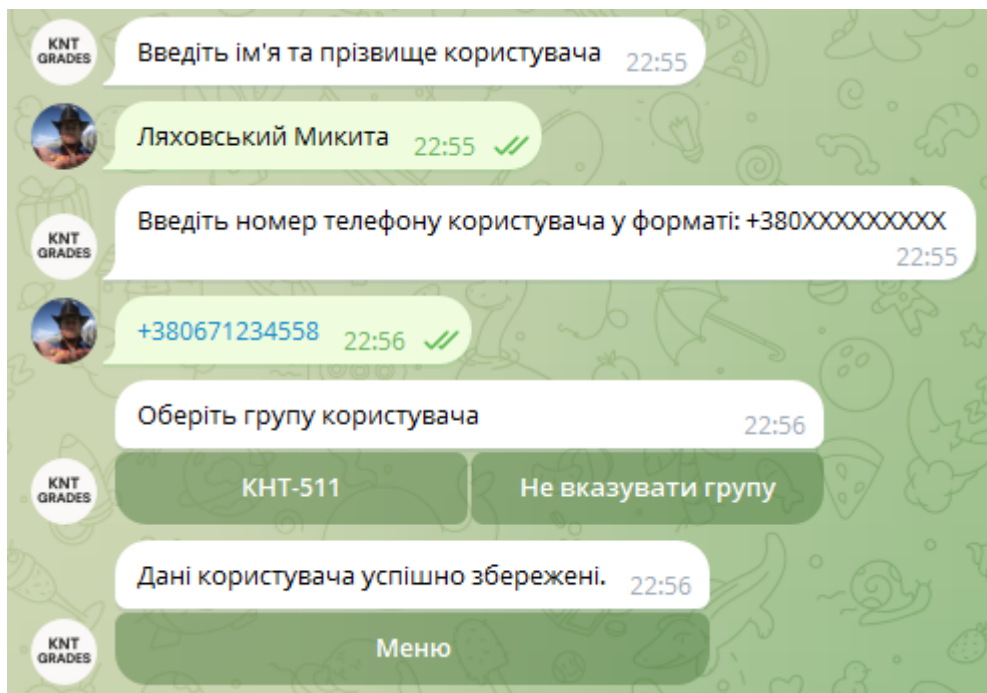


Рисунок 4.19 – Процес створення нового користувача у боті

id	tg_user_id	phone_number	full_name	is_teacher	group_id
Фи...	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр
1	1133213	+380733333333	Авдеев Нікіта	0	1
3	213344	+380731111111	Віржков Денис	0	1
5	2146212	+380734444444	Восус Ілля	0	1
6	2135657682	+380735555555	Гетман Сергій	0	1
7	1231235	+380736666666	Гречка Нікіта	0	1
8	213578732	+380737777777	Дроздов Святослав	0	1
9	2579325565	+380738888888	Залозний Олександр	0	1
10	6344533321	+380739999999	Кодак Ірина	0	1
11	704614466	+380731145585	Ковальчук Артем	1	NULL
12	NULL	+380671234558	Ляховський Микита	0	1

Рисунок 4.20 – Результат створення користувача

ВИСНОВКИ

У результаті виконання дипломної роботи було успішно реалізовано Telegram-бота, призначеного для зручного, швидкого та безпечного отримання студентами оцінок. Реалізація такого інструменту є актуальною відповіддю на потребу в цифровізації освітніх процесів, зокрема в частині комунікації між навчальним закладом та студентами.

Розроблений бот дозволяє студентам у будь-який момент отримати доступ до своєї академічної інформації без необхідності фізичної присутності в навчальному закладі або постійного звертання до адміністративного персоналу. Це значно спрощує доступ до освітніх послуг, зменшує навантаження на деканат та сприяє загальній інформатизації освітнього середовища.

Технічно, реалізація проєкту базувалась на використанні бібліотеки Aiogram, яка, завдяки повністю асинхронній архітектурі, забезпечує високу продуктивність, швидку обробку запитів і зручну структуру для масштабування та підтримки коду. Використання SQLite у поєднанні з бібліотекоюaiosqlite стало вдалим рішенням з огляду на простоту інтеграції, відсутність потреби у розгортанні окремого сервера та задовільну продуктивність для поставлених задач.

У процесі розробки також було враховано базові вимоги до захисту даних: обмеження доступу за номером телефону та ID користувача, перевірка введених даних, контроль помилок. Це дозволяє підтримувати базовий рівень безпеки при роботі з персональною інформацією студентів, такою як оцінки чи академічна заборгованість.

Також у ході роботи проведено порівняльний аналіз інших популярних рішень для зберігання даних. З'ясовано, що хоча для великих систем доцільніше використовувати PostgreSQL або MySQL, у рамках цього проєкту SQLite є цілком прийнятним вибором, який забезпечує оптимальний баланс між простотою, продуктивністю й обсягом ресурсу.

Отже, дипломна робота досягла своєї мети: було створено ефективний, простий у використанні та надійний Telegram-бот, який вирішує реальну задачу – полегшує студентам доступ до оцінок та заборгованостей. Результати розробки можна розглядати як основу для подальшого вдосконалення та розширення систем електронної взаємодії у сфері вищої освіти.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Python Software Foundation. Python Language Reference, version 3.11. URL: <https://docs.python.org/3/> (Дата звернення 25.04.2025)
2. Real Python. Практичні матеріали для розробників на Python. URL: <https://realpython.com/> (Дата звернення 25.04.2025)
3. Python Telegram Bot Documentation. URL: <https://python-telegram-bot.org/> (Дата звернення 27.04.2025)
4. Telethon Documentation. URL: <https://docs.telethon.dev/> (Дата звернення 27.05.2025)
5. Pyrogram Documentation. URL: <https://docs.pyrogram.org/> (Дата звернення 27.04.2025)
6. Aiogram Team. Aiogram Documentation (v3 beta). URL: <https://docs.aiogram.dev/> (Дата звернення 27.04.2025)
7. SQLite. SQLite Documentation. URL: <https://www.sqlite.org/docs.html> (Дата звернення 29.04.2025)
8. PostgreSQL Global Development Group. PostgreSQL Official Documentation. URL: <https://www.postgresql.org/docs/> (Дата звернення 29.04.2025)
9. Oracle. MySQL Documentation. URL: <https://dev.mysql.com/doc/> (Дата звернення 29.04.2025)
10. MongoDB Inc. MongoDB Manual. URL: <https://www.mongodb.com/docs/manual/> (Дата звернення 29.04.2025)
11. SQLAlchemy. SQLAlchemy Documentation. URL: <https://www.sqlalchemy.org/> (Дата звернення 30.04.2025)
12. GitHub. aiosqlite – Async SQLite 3 driver for Python 3. URL: <https://github.com/omnilib/aiosqlite> (Дата звернення 30.04.2025)
13. Python Enhancement Proposals. PEP 492 – Coroutines with async and await syntax. URL: <https://peps.python.org/pep-0492/> (Дата звернення 30.04.2025)

14. Telegram Messenger LLP. Telegram Bot API. URL: <https://core.telegram.org/bots/api> (Дата звернення 01.05.2025)

15. Дистрибутив мови Python. URL: <https://www.python.org/downloads/> (Дата звернення 15.05.2025)

ДОДАТОК А
Слайди презентації

Розробка інформаційної системи для студентів ВНЗ

Виконав:
студент групи КНТ-511
Ковальчук Артем

Керівник:
к.т.н., доцент
Тягунова Марія

Рисунок А.1 – Слайд 1

Мета та завдання роботи

Мета: Підвищення ефективності навчального процесу та забезпечення студентам зручного, швидкого й безпечного доступу до оцінок через Telegram-бот.

Завдання:

- Проаналізувати сучасні інструменти створення ботів;
- Сформулювати вимоги до системи;
- Спроектувати архітектуру та інтерфейс бота;
- Реалізувати Telegram-бот;
- Перевірити працездатність і протестувати систему.

Рисунок А. 2 – Слайд 2

Актуальність проекту

Актуальність проекту полягає у необхідності оптимізувати процес інформування студентів про їхні навчальні результати у сучасних закладах освіти. Традиційні способи — через деканат, паперові або електронні відомості — часто незручні та повільні. З огляду на популярність месенджера Telegram серед молоді, саме ця платформа є доцільним вибором для автоматизації процесу отримання студентами оцінок. Telegram-бот забезпечує швидкий та захищений доступ до персональних навчальних результатів, зменшує навантаження на адміністрацію та дає змогу студенту переглянути свої оцінки у будь-який момент.

Рисунок А.3 – Слайд 3

Обрані інструменти

Основною мовою програмування обрано Python, яка завдяки своїй гнучкості та великій кількості бібліотек ідеально підходить для створення ботів і роботи з базами даних. Для реалізації взаємодії із Telegram API застосовано бібліотеку **Aiogram**, що дозволяє будувати асинхронних Telegram-ботів з підтримкою складної логіки, FSM та інтерактивних клавіатур.

У якості системи керування базою даних використано легку, але функціональну SQLite, що у поєднанні з ORM-бібліотекою **SQLAlchemy** забезпечує зручне описання структури таблиць на рівні Python-коду й гарантує захист від SQL-ін'єкцій. Для асинхронної роботи з базою даних використовується драйвер **aiosqlite**, що підвищує продуктивність бота при роботі з великою кількістю запитів.

Також проект організовано за принципами модульності, що спрощує його підтримку та розширення. Реєстрація і управління Telegram-ботом здійснюється через офіційний сервіс BotFather, що забезпечує безпеку й унікальність доступу до API. Такий вибір інструментів дозволяє гарантувати надійну, ефективну й масштабовану роботу всієї системи.

Рисунок А.4 – Слайд 4

Архітектура системи

Розроблена система має три основні компоненти:

- Telegram-бот, що слугує інтерфейсом для студентів і викладачів;
- серверну частину, яка відповідає за обробку запитів користувачів та взаємодію з базою даних;
- реляційну базу даних, у якій зберігається структурована інформація про студентів, групи, предмети й оцінки.

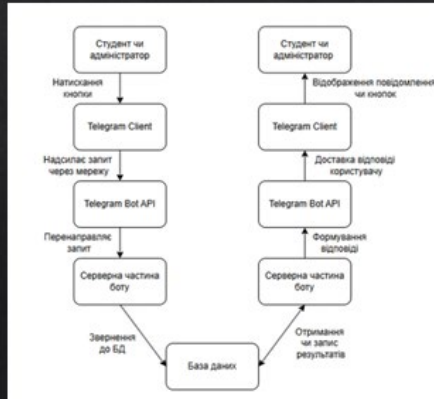


Рисунок А.5 – Слайд 5

Функціональні можливості бота

Студенту доступно:

- Авторизація через Telegram ID та номер телефону;
- Перегляд власних оцінок за предметами та семестрами.

Адміністратору доступно:

- Реєстрація нових користувачів;
- Призначення, редагування, видалення оцінок;
- Керування предметами і групами;
- Призначення предметів окремим студентам або групам.

Для цього реалізовано динамічні клавіатури для вибору груп, предметів, студентів і т.п., що мінімізує помилки та прискорює роботу.

Рисунок А.6 – Слайд 6

Захист та конфіденціальність

Авторизація за Telegram User ID і номером телефону;

Студент бачить тільки власні оцінки;

Розмежування прав доступу: студент / адміністратор;

Адміністративні ролі призначаються вручну через БД;

Захист від SQL-ін'єкцій — використання ORM SQLAlchemy та параметризованих запитів.

Рисунок А.7 – Слайд 7

Побудова бази даних

Для зберігання інформації використовується реляційна база даних SQLite, яка містить ключові сутності: студенти, групи, предмети та оцінки.

Структура бази даних забезпечує зручне зберігання, швидкий пошук та обробку даних, а також дозволяє гнучко призначати предмети окремим студентам або цілим групам.

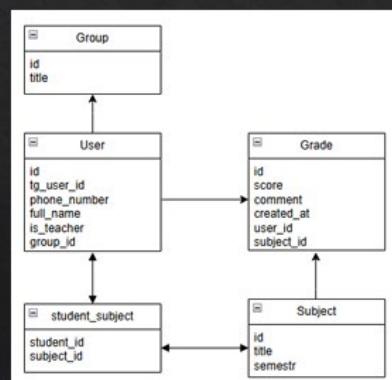


Рисунок А.8 – Слайд 8

Можливості масштабування та розвитку системи

Реалізована система побудована з урахуванням можливості подальшого розширення її функціоналу. Наприклад, в подальшому можливий її перехід на іншу СУБД, реалізація більшої кількості ролей, розширений функціонал з розкладами, посиланнями на пари, тощо. Для зручності в подальшому можна реалізувати внесення даних в базу через таблиці Excel або інші.

Рисунок А.9 – Слайд 9

Висновки

У результаті виконання роботи було розроблено сучасну інформаційну систему у вигляді Telegram-бота для автоматизації обліку оцінок студентів. Проєкт пройшов етапи від аналізу технічного завдання і вибору інструментів до впровадження архітектури, розробки коду та демонстрації готового продукту.

Запропоноване рішення відповідає сучасним тенденціям цифровізації освіти. Система забезпечує зручний, швидкий і захищений доступ студентів до власних навчальних результатів, а також спрощує роботу викладачів з оцінками, групами і предметами. Модульна та асинхронна архітектура дозволяє легко масштабувати та розширювати функціонал у майбутньому.

Рисунок А.10 – Слайд 10