ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «**TELEGRAM-БОТ ДЛЯ ОСВІТНІХ ПРОСТОРІВ НА PYTHON**»

на здобуття освітнього ступеня бакалавра
зі спеціальності 122 Комп'ютерні науки
(код, найменування спеціальності)
освітньо-професійної програми Комп'ютерні науки
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Іванна РИЩИКОВЕЦЬ
(підпис) *(Ім'я, ПРІЗВИЩЕ здобувача)*

Виконала: здобувачка вищої освіти гр. КНД-42, Іванна РИЩИКОВЕЦЬ

Керівник: Максим ФЕСЕНКО
Науковий ступінь вчене звання: д.ф.-м.н., професор

Рецензент:
Науковий ступінь, вчене звання
_____
(Ім'я, ПРІЗВИЩЕ)

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Комп'ютерних наук

Ступінь вищої освіти Бакалавр

Спеціальність 122 Комп'ютерні науки

Освітньо-професійна програма «Комп'ютерні науки»

ЗАТВЕРДЖУЮ
Завідувач кафедрою
Комп'ютерних наук
_____ Віктор ВИШНІВСЬКИЙ
«_____» __ _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Рищиковець Іванні Олекандрівні
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: «Telegram-бот для освітніх просторів на Python»

Керівник кваліфікаційної роботи Олена ШИКУЛА д.ф.-м.н., професор,
(Ім'я, ПРІЗВИЩЕ науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» лютого 2025р. № 56

2. Строк подання кваліфікаційної роботи «30» травня 2025р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література, середовище Python, параметри для створення інформаційної системи.

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

1. Аналіз інформаційних систем, їх класифікація, архітектура та створення.
2. Розробка інформаційної системи контролю записів учнів на гуртки освітнього простору.

3. Розробка Telegram-бота на мові Python для автоматизації процесу запису учнів, організації розкладу, та комунікації адміністрації щколи з учнями.

5. Перелік графічного матеріалу: *презентація*
   1. Схема меню інформаційної системи.
   2. Створення таблиць
   3. Екранні форми інформаційної системи.
   4. Фрагменти програмного коду.

6. Дата видачі завдання «24» лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	23.02-09.03.25	виконано
2	Оцінка та аналіз сучасних методів менеджменту в освітніх просторах	09.03-16.03.25	виконано
3	Формування вимог до Telegram-бота для освітніх просторів	17.03-23.03.25	виконано
4	Проєктування архітектури бота	24.03-29.03.25	виконано
5	Вибір інструментів розробки	01.04-07.04.25	виконано
6	Розробка Telegram-бота	08.04-24.04.25	виконано
7	Оформлення роботи: вступ, висновки, реферат	25.04-02.05.25	виконано
8	Розробка демонстраційних матеріалів	03.05-08.05.25	виконано

Здобувач вищої освіти _____ Іванна РИЩИКОВЕЦЬ
(підпис) (Ім'я, ПРІЗВИЩЕ)

Керівник кваліфікаційної роботи _____ Максим ФЕСЕНКО
(підпис) (Ім'я, ПРІЗВИЩЕ)

ЗМІСТ

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 37 стор., 7 табл., 9 рис., 40 джерел.

Мета роботи:

Метою даної роботи є дослідження процесу взаємодії користувачів у сфері позашкільної освіти з цифровими технологіями, а також розробка Telegram бота для спрощення взаємодії користувачів (батьків) із адміністрацією дитячого простору та автоматизації й полегшення адміністративних процесів закладу.

Робота спрямована на аналіз сучасних інструментів і методів, які сприяють покращенню комунікації та оптимізації управління в освітніх закладах.

Об’єкт дослідження:

Об’єктом дослідження є процес взаємодії користувача з цифровими системами у сфері освіти, зокрема у контексті позашкільних освітніх закладів, де технології відіграють ключову роль у спрощенні доступу до інформації та управлінні адміністративними завданнями.

Предмет дослідження:

Предметом дослідження є методи та засоби впровадження Telegram-ботів для обробки запитів, реєстрації користувачів і управління даними в контексті освітніх послуг. Особлива увага приділяється можливостям автоматизації рутинних процесів і забезпечення зручного доступу до інформації для всіх учасників освітнього процесу.

Короткий зміст роботи:

У теоретичній частині дипломної роботи розглянуто основи створення та реалізації Telegram-ботів, їхню архітектуру, принципи роботи та особливості використання API Telegram. Досліджено переваги використання ботів у сфері освіти, зокрема для автоматизації комунікації та управління даними. Окремо проаналізовано технічні аспекти, пов’язані з розробкою ботів, включаючи інтеграцію з платформою Telegram через API та принципи обробки запитів користувачів.

Практична частина роботи присвячена створенню Telegram-бота на мові програмування Python із використанням бібліотеки Aiogram. Розроблений бот забезпечує зручний інтерфейс для користувачів, дозволяючи переглядати актуальний перелік гуртків, обирати заняття та проходити реєстрацію на обраний гурток. Для адміністраторів закладу передбачено спеціальний функціонал: після введення пароля через опцію "🔐 Адмін-панель" відкривається доступ до редагування розкладу занять та перегляду списку зареєстрованих користувачів. Дані користувачів і розклад зберігаються у файлах users.json і schedule.txt, що забезпечує базову функціональність системи.

Бот дозволяє користувачам (батькам) швидко отримувати інформацію про розклад, обирати гуртки та реєструватися на заняття, а адміністраторам — ефективно управляти розкладом і списком учасників, що значно полегшує організаційні процеси в дитячому просторі. У роботі також проведено тестування бота, яке підтвердило його працездатність і відповідність поставленим вимогам.

Ключові слова: Telegram-бот, Aiogram, Python, реєстрація, освітній центр, гуртки, автоматизація, чат-бот, адміністрування, розклад, бот для освітніх просторів.

ВСТУП

Інформатизація суспільства стала однією з ключових ознак сучасного етапу розвитку цивілізації. Усі сфери життя активно впроваджують цифрові рішення для оптимізації процесів, зменшення впливу людського чинника та підвищення ефективності. Особливо важливою ця трансформація є для галузі освіти, де потреба у швидкій, надійній та доступній комунікації між усіма учасниками навчального процесу зростає з кожним роком.

Останні роки довели, що освітні заклади мають бути готові до гнучкої взаємодії з учнями, батьками та викладачами як в онлайн-, так і в офлайн-середовищі. Розвиток дистанційного навчання, необхідність керування позаурочними активностями (гуртки, факультативи, консультації) та важливість оперативного інформування вимагають впровадження сучасних інструментів менеджменту.

Одним із найбільш доступних і водночас потужних інструментів є Telegram — месенджер із відкритим API, що дозволяє створювати повноцінних ботів для автоматизації комунікації та обслуговування користувачів. Завдяки своїй популярності серед молоді, Telegram є зручним каналом для впровадження освітніх рішень, особливо коли йдеться про школярів та їхніх батьків, які вже активно користуються цією платформою.

Актуальність теми цієї дипломної роботи полягає в необхідності створення простої, ефективної та доступної системи управління освітніми просторами, зокрема — процесом запису на позашкільні курси, обробки запитів користувачів, надання розкладу занять та адміністративного моніторингу. Сучасні школи та гурткові центри часто використовують ручні методи обліку (Google-форми, Excel, паперові журнали), які не тільки є неефективними, але й створюють додаткове навантаження на працівників.

Метою кваліфікаційної роботи є розробка Telegram-бота для автоматизованого управління освітніми послугами, з урахуванням зручності для

кінцевого користувача, простоти адміністрування та можливості масштабування.

У межах реалізації передбачається використання мови програмування Python, фреймворку aiogram, а також базових інструментів збереження даних у форматі JSON або вбудованої бази даних SQLite.

Предметом дослідження є процес автоматизації взаємодії користувачів з освітнім середовищем за допомогою чат-бота. Об'єктом дослідження — цифрові інструменти менеджменту освітніх активностей у позашкільній сфері.

Основні завдання дослідження включають:

- аналіз потреб навчальних закладів у сфері менеджменту роботи;
- дослідження можливостей Telegram API;
- проектування архітектури Telegram-бота;
- реалізацію основного функціоналу;
- тестування бота на цільовій аудиторії;
- формулювання висновків і пропозицій щодо майбутнього розвитку.

Результатом реалізації стане працюючий Telegram-бот, здатний приймати запити на запис до гуртків, відображати розклад занять, здійснювати обробку запитів користувачів і надавати обмежений доступ до адміністративної панелі. Рішення буде придатне до впровадження в реальних умовах і забезпечуватиме ефективну комунікацію між закладом освіти та учасниками освітнього процесу.

Таким чином, ця дипломна робота є практично орієнтованою розробкою, яка відповідає сучасним вимогам до цифрових рішень в освіті, має прикладне значення й демонструє потенціал для подальшого розширення функціоналу.

1 АНАЛІЗ РОЗВИТКУ ТЕХНОЛОГІЙ

1.1 Історичний огляд розвитку інформаційних технологій в освіті

Еволюція освітніх технологій є яскравим прикладом трансформації від простих допоміжних засобів до інтегрованих цифрових екосистем, що є основою сучасного навчального процесу. Наприкінці 20-го століття, період, коли інформаційні технології тільки починали свій шлях у масовому сегменті, впровадження перших персональних комп'ютерів у шкільні навчальні заклади стало знаковою подією. Ці машини, які нині здаються архаїчними, переважно використовувалися для вивчення основ програмування — від простих алгоритмів до створення базових застосунків. Це був своєрідний вступ до цифрового світу, що відкривав нові горизонти для викладання та навчання. Тоді ніхто й не уявляв, наскільки глибоко комп'ютери змінять освіту.

З настанням 1990-х років і стрімким поширенням Інтернету, освіта пережила справжню революцію. Глобальна мережа не лише надала доступ до невичерпних обсягів інформації, але й сприяла появі перших онлайн-курсів. Ці ранні спроби дистанційного навчання були новаторськими, дозволяючи студентам отримувати знання незалежно від їхнього географічного розташування. Викладачі, зі свого боку, активно створювали електронні бази знань та цифрові бібліотеки, що уможливило доступ до навчальних матеріалів 24/7. Цей етап заклав фундамент для концепції відкритої освіти та постійного доступу до інформації.

Початок 2000-х років характеризувався інтенсивним розвитком систем управління навчанням (LMS), серед яких Moodle стала одним із флагманів. Ці платформи пропонували комплексний підхід до організації навчального процесу: від розміщення навчальних матеріалів до проведення онлайн-тестувань, ведення журналів оцінок, управління домашніми завданнями та організації інтерактивних форумів. LMS перетворилися на віртуальні класи, які

дозволяли викладачам ефективно керувати курсами, а учням — систематизовано отримувати доступ до всіх необхідних ресурсів у будь-який зручний час, що значно підвищило гнучкість навчання та автономність студентів. Також з’явилися можливості для персоналізації навчання, адже викладачі могли відстежувати прогрес кожного учня та адаптувати матеріали.

У 2010-х роках, з масовим поширенням смартфонів та повсюдною інтеграцією месенджерів у повсякденне життя, освітній простір почав активно адаптуватися до мобільних технологій. З'явилися мобільні додатки, спеціально розроблені для освітніх цілей, що дозволяли оперативно обмінюватися інформацією. Наприклад, додатки для перегляду розкладу, отримання сповіщень про оцінки, нагадування про дедлайни стали нормою. Це не лише підвищило зручність для учнів, але й прискорило комунікацію між усіма учасниками освітнього процесу, роблячи її практично миттєвою. Мобільність стала ключовим фактором у забезпеченні безперервного доступу до освіти.

На сучасному етапі ми спостерігаємо глибоку інтеграцію технологій штучного інтелекту (ШІ) та чат-ботів в освітнє середовище. Ці інструменти вже не просто допомагають, а автоматизують значну частину рутинних адміністративних завдань. Це включає автоматичний запис на заняття, інформування про зміни в розкладі, надання швидких відповідей на типові питання, а також персоналізовані рекомендації щодо навчання. Розвиток цих технологій підкреслює загальну тенденцію до спрощення доступу до інформації та автоматизації освітніх процесів.

Однак, попри значні досягнення, багато існуючих систем досі характеризуються певною громіздкістю та низькою адаптивністю, що створює об'єктивну потребу у розробці більш простих, інтуїтивно зрозумілих та гнучких рішень, які зможуть ефективно відповідати потребам сучасного користувача.

1.2 Сучасні технології для освітнього менеджменту

Сучасний ландшафт освітнього менеджменту характеризується широким спектром інструментів, що забезпечують ефективне управління навчальним процесом. Провідні платформи, такі як Google Classroom, надають викладачам можливість створювати та поширювати завдання, а учням — зручно їх надсилати в онлайн-форматі, забезпечуючи централізований обмін інформацією та файлами. Microsoft Teams пропонує інтеграцію з пакетом офісних програм Microsoft, що робить його ідеальним інструментом для організації відеоконференцій, спільної роботи над документами в режимі реального часу та ефективної командної взаємодії. Moodle, як одна з найстаріших і найдосвідченіших систем управління навчанням, залишається популярною завдяки своїй здатності підтримувати створення повноцінних навчальних курсів, інтегрувати різноманітні формати контенту, проводити тести, організовувати форуми для обговорень та вести детальну статистику успішності.

Однак, останні роки виявили особливу перспективу у використанні месенджерів, зокрема Telegram, для освітніх цілей. Чат-боти в Telegram здобувають популярність завдяки своїй винятковій простоті використання, широкій доступності та високій швидкості комунікації. Приклади їх успішного застосування включають ботів для оперативного нагадування про уроки, автоматичної реєстрації на додаткові курси та інформування про зміни в розкладі. Це дозволяє миттєво доносити важливу інформацію до учнів, а викладачам — значно економити час на виконанні рутинних адміністративних завдань, звільняючи їх для більш важливої педагогічної роботи. Інтуїтивно зрозумілий інтерфейс месенджерів дозволяє швидко адаптуватися до використання таких ботів, що особливо цінно для користувачів з різним рівнем цифрової грамотності.

Окрім цього, значний прогрес спостерігається у сфері застосування штучного інтелекту (ШІ) в освітньому менеджменті. ШІ вже демонструє здатність до глибокого аналізу індивідуальних потреб учнів на основі їхнього

прогресу, стилю навчання та інтересів. Це дозволяє ШІ не просто пропонувати, а й формувати індивідуальні освітні траєкторії та персоналізовані плани навчання, адаптуючи контент і темп викладання до унікальних особливостей кожного учня. Також активно розвиваються системи, що інтегруються з Інтернетом речей (IoT). Наприклад, використання сенсорів у класах для автоматичного контролю відвідуваності або моніторингу кліматичних умов для створення оптимального навчального середовища. Проте, важливо зазначити, що впровадження цих передових технологій часто пов'язане зі значними фінансовими витратами та технічною складністю, що обмежує їх доступність для невеликих освітніх просторів, таких як мовні школи, приватні гуртки чи творчі студії.

На підставі детального аналізу, можна констатувати, що месенджери, зокрема Telegram, представляють собою надзвичайно перспективний напрямок для розробки простих, ефективних та доступних рішень у сфері освітнього менеджменту. Їхня здатність забезпечувати швидку комунікацію та автоматизацію процесів робить їх ідеальними для оптимізації освітніх послуг у широкому спектрі навчальних закладів, особливо там, де обмежені ресурси не дозволяють впровадження складних і дорогих LMS-систем.

1.3 Аналіз існуючих рішень та їх недоліків

На ринку інформаційних технологій представлено велику кількість рішень, призначених для управління освітніми процесами, проте їх функціональність, доступність та зручність використання часто є неоднорідними та мають значні обмеження. Розглянемо, наприклад, бот EduBot. Хоча він ефективно надає учням актуальний розклад та нагадування про заняття, його функціонал не поширюється на більш складні адміністративні завдання, такі як запис на додаткові заняття, управління групами чи відстеження відвідуваності. Це обмежує його застосовність до простих інформаційних потреб. Аналогічно,

боти, створені за допомогою BotFather, часто мають лише базові функції. Вони дозволяють налаштувати прості команди та відповіді, але не надають гнучкості для розробки складного інтерфейсу, інтеграції з базами даних або додавання розширених функцій, що є критично важливими для повноцінного управління навчальним процесом, наприклад, комплексне управління розкладом з можливістю редагування для викладачів.

Потужні та широко відомі платформи, такі як Google Classroom і Moodle, безперечно, пропонують широкий спектр можливостей: від створення завдань та інтерактивних тестів до керування оцінками та організації спільної роботи. Однак, їхня багатофункціональність часто обертається складністю освоєння. Викладачам та адміністраторам, особливо без глибоких технічних знань, може знадобитися значний час та додаткові курси для повноцінного опанування всіх можливостей цих систем. Для великих університетів чи шкіл, де є штатні IT-спеціалісти, це може бути прийнятно, але для невеликих освітніх просторів – приватних репетиторських центрів, мовних курсів, творчих студій, гуртків – ці платформи виявляються надмірно громіздкими та невиправдано складними. Вони схожі на багатофункціональний комбайн, який не потрібен для обробки невеликої грядки. Крім того, значною проблемою є недостатня або неповна локалізація українською мовою в багатьох міжнародних освітніх системах, що створює бар'єри у використанні для українськомовних користувачів та знижує загальну зручність.

Ще одним критичним недоліком, який часто спостерігається в існуючих рішеннях, є відсутність гармонійної інтеграції з популярними месенджерами. У сучасному світі Telegram став де-факто стандартом для швидкої та ефективної комунікації між учнями, викладачами та батьками. Проте, більшість освітніх платформ не мають прямого зв'язку з месенджерами. Це призводить до фрагментації інформаційного потоку: учням доводиться перевіряти розклад в одній системі, спілкуватися з викладачем у Telegram, а здавати домашні завдання на іншій платформі. Таке розпорошення інформації не лише незручне, але й веде

до втрати часу, потенційних помилок у комунікації та загального зниження ефективності освітнього менеджменту.

Отже, аналіз існуючих рішень виявляє чіткі обмеження: надмірна складність для специфічних потреб, наявність мовних бар'єрів та критична відсутність інтеграції з основними комунікаційними каналами. Ці недоліки переконливо підтверджують об'єктивну потребу у розробці власного, спеціалізованого Telegram-бота. Таке рішення має бути не лише простим та інтуїтивно зрозумілим, але й максимально адаптованим до унікальних потреб невеликих освітніх просторів, забезпечуючи централізоване та ефективне управління навчальним процесом через звичний для користувачів інтерфейс месенджера.

2 ОПИС ОБРАНОЇ ТЕХНОЛОГІЇ

2.1 Огляд платформи Telegram і API

Платформа Telegram, що з'явилася на світ у 2013 році завдяки зусиллям Павла Дурова та його команди, стрімко здобула глобальне визнання та значну популярність, утвердившись як один із провідних месенджерів у світі. Її непересічний успіх базується на кількох фундаментальних принципах, що відрізняють її від конкурентів: висока швидкість передачі повідомлень, що забезпечує миттєву комунікацію; надійна система шифрування, яка гарантує конфіденційність та безпеку даних користувачів; та винятковій гнучкості у функціоналі, що дозволяє адаптувати платформу під різноманітні потреби. Однією з найважливіших переваг Telegram, яку я, ціную особливо, є його кросплатформність. Він доступний на широкому спектрі пристроїв, включаючи смартфони (iOS, Android), настільні комп'ютери (Windows, macOS, Linux) та навіть через веб-інтерфейс у будь-якому сучасному браузері. Ця універсальність робить його надзвичайно зручним для всіх категорій користувачів в освітньому середовищі — від учнів, які можуть використовувати його на своїх мобільних пристроях, до викладачів та адміністраторів, що працюють з комп'ютерами. Кожен учасник освітнього процесу може взаємодіяти з платформою, використовуючи той пристрій, який є для нього найзручнішим, що значно знижує бар'єри для впровадження нових технологій та забезпечує максимальну інклюзивність.

Ключовою особливістю, що виділяє Telegram серед інших месенджерів та робить його ідеальною платформою для автоматизації освітніх процесів, є підтримка ботів через Bot API (Application Programming Interface). Bot API — це потужний, добре документований набір інструментів, що дозволяє розробникам створювати інтерактивні програми, які можуть спілкуватися з користувачами безпосередньо через чат. Це не просто відправлення текстових повідомлень; бот

може виконувати широкий спектр дій: надсилати текстові та мультимедійні повідомлення (фото, відео, аудіо, файли будь-якого типу), створювати кастомні клавіатури з кнопками (як звичайними, що замінюють стандартну клавіатуру, так і вбудованими в повідомлення — так звані inline-клавіатури), обробляти різноманітні команди (наприклад, /start, /help, /schedule), взаємодіяти з користувачем у складних діалогових сценаріях, а також інтегруватися з каналами та групами. У контексті мого проєкту, Bot API є фундаментальною основою для розробки бота, який буде виконувати такі критично важливі функції, як оперативне відображення актуального розкладу занять, забезпечення зручної можливості запису на додаткові гуртки та секції, а також надання адміністративних функцій для ефективного управління освітнім процесом.

Процес створення бота в Telegram є напрочуд інтуїтивно зрозумілим та доступним навіть для початківців у програмуванні. Для цього необхідно звернутися до спеціального, офіційного бота @BotFather у Telegram. Після надсилання команди /newbot користувачеві послідовно пропонується обрати назву для свого бота (наприклад, "SmartChildBot" – назва, яку я обрала для свого проєкту, що відображає його освітню спрямованість) та унікальне ім'я користувача (username), яке має закінчуватися на "bot" (наприклад, @ChildTutorBot). Після успішного завершення цих кроків, @BotFather генерує унікальний API-токен, який є довгим рядком символів, наприклад: 7599198811:AAGQ_rurdlArgcNWK1vLbIMyNkuUMPT-0XA. Цей токен є своєрідним "ключем" до вашого бота і є критично важливим для підключення програмного коду бота до серверів Telegram. Він дозволяє застосунку відправляти запити до Telegram API та отримувати оновлення від користувачів. Зберігання цього токена в безпеці є першочерговим завданням, оскільки він надає повний контроль над функціоналом бота.

Переваги використання Telegram для реалізації мого проєкту є численними та вагомими, що стало вирішальним фактором при виборі платформи:

- Бюджетність: Створення та базове використання функціоналу бота не потребує жодних фінансових витрат. Це є надзвичайно важливим аспектом для

освітніх проєктів, які часто мають обмежені бюджети, дозволяючи зосередити ресурси на розробці функціоналу та покращенні користувацького досвіду, а не на ліцензійних платежах чи оплаті інфраструктури. Це забезпечує сталість та доступність рішення.

- Підтримка української мови: Інтерфейс Telegram повністю локалізований українською мовою, а Bot API дозволяє надсилати повідомлення та формувати клавіатури рідною мовою. Це забезпечує комфортне та інтуїтивно зрозуміле використання для українських користувачів, що є критично важливим для широкого впровадження в освітній системі України та сприяє інклюзивності.

- Висока популярність: Значна частина учнів, викладачів та батьків вже активно використовують Telegram у повсякденному житті для особистого та професійного спілкування. Це значно знижує бар'єр для входу, оскільки користувачам не потрібно встановлювати нові додатки або освоювати незнайомі інтерфейси. Вони можуть взаємодіяти з ботом у звичному для себе середовищі, що прискорює адаптацію та прийняття нового інструменту.

- Простота та інтуїтивність: Використання інтерактивних клавіатур, вбудованих кнопок та простих текстових команд дозволяє користувачам швидко та ефективно отримувати потрібну інформацію або виконувати дії, мінімізуючи необхідність запам'ятовування складних інструкцій. Це робить бота доступним для користувачів з різним рівнем цифрової грамотності, від наймолодших учнів до дорослих адміністраторів.

- Надійність та швидкість: Telegram відомий своєю стабільною роботою, високою швидкістю доставки повідомлень та стійкістю до навантажень. Це є важливим для оперативного інформування та взаємодії в освітньому процесі, особливо у випадках, коли потрібне швидке поширення інформації (наприклад, зміни в розкладі, оголошення про надзвичайні ситуації).

Однак, існують певні обмеження, які я врахувала при розробці бота та які необхідно мати на увазі при його експлуатації:

Бот не може ініціювати розмову першим: Задля уникнення спаму та забезпечення конфіденційності користувачів, бот не може самостійно надсилати

повідомлення користувачеві, доки той не надішле перше повідомлення (наприклад, команду /start) або не додасть бота до групи. Це вимагає від користувача першого кроку для початку взаємодії, що може бути вирішено через інструкції на сайті або в інших комунікаційних каналах.

- Обмеження на розмір повідомлень та файлів: Існує ліміт на розмір текстового повідомлення (до 4096 символів) та розмір файлів, які можна надсилати через бот (до 50 МБ для безкоштовних ботів). Хоча для більшості освітніх потреб (розклад, короткі оголошення, невеликі документи) цього достатньо, для передачі великих відео-лекцій, об'ємних навчальних матеріалів або високоякісних зображень можуть знадобитися альтернативні рішення, такі як посилання на хмарні сховища або компресія даних.

Таким чином, платформа Telegram є ідеальним вибором для реалізації мого освітнього бота завдяки своїй широкій доступності, надійній інфраструктурі та потужним можливостям Bot API, що дозволяють створити функціональний, зручний та масштабований інструмент для управління навчальним процесом.

2.2 Огляд бібліотеки aiogram для розробки ботів

Для розробки функціонального та ефективного Telegram-бота я обрала бібліотеку aiogram. Ця бібліотека, написана на мові програмування Python, є спеціалізованим інструментом для взаємодії з Telegram Bot API. Мій вибір aiogram був зумовлений її ключовою перевагою – архітектурою, заснованою на асинхронному програмуванні з використанням вбудованої в Python бібліотеки asyncio. Асинхронність дозволяє боту обробляти велику кількість запитів одночасно, не блокуючи виконання інших операцій. Це має вирішальне значення для мого освітнього бота, оскільки він повинен оперативно реагувати на повідомлення від кількох учнів або викладачів одночасно. Наприклад, коли один користувач переглядає розклад, інший може в цей же момент записуватися на гурток, а третій — надсилати запит адміністратору, і всі ці дії обробляються

паралельно, забезпечуючи високу швидкість відгуку та безперебійну роботу системи. Це дозволяє уникнути "вузьких місць" та забезпечити плавну взаємодію навіть при зростанні кількості користувачів.

Розглянемо основні компоненти бібліотеки aiogram, які є фундаментом для побудови логіки мого бота:

- Bot: Це центральний об'єкт, який встановлює з'єднання з серверами Telegram за допомогою наданого API-токену. Він є основним інтерфейсом для всіх операцій, пов'язаних з Telegram API, відповідаючи за відправлення повідомлень, файлів, обробку команд та отримання оновлень від користувачів. Bot інкапсулює всі низькорівневі взаємодії з API, дозволяючи мені зосередитися на бізнес-логіці, а не на деталях мережевих протоколів.

- Dispatcher: Цей компонент є "мозком" бота, відповідальним за обробку всіх вхідних оновлень (повідомлень, команд, натискань кнопок тощо), що надходять від Telegram. Dispatcher аналізує тип оновлення та передає його відповідній функції-обробнику (handler). Наприклад, якщо користувач надсилає команду /start, Dispatcher ідентифікує цю команду і викликає спеціально визначену асинхронну функцію, яка відповідає за обробку цієї команди, забезпечуючи логічне розгалуження поведінки бота. Це дозволяє мені структурувати код та легко додавати нові функції.

- Message: Це клас, який представляє собою об'єкт вхідного повідомлення від користувача. Він містить всю інформацію про повідомлення: текст, ідентифікатор користувача, дані про відправника, час відправлення, тип повідомлення (текст, фото, документ) та інші метадані. Message є лише одним з типів оновлень; aiogram також обробляє CallbackQuery (для натискань inline-кнопок), InlineQuery (для взаємодії з ботом в інших чатах) та інші, що дозволяє мені реалізовувати різноманітні сценарії взаємодії.

Приклад ініціалізації бота в aiogram демонструє простоту підключення до Telegram API та готовність до обробки подій:

```
from aiogram import Bot, Dispatcher, types
```

from aiogram.contrib.fsm.storage.memory import MemoryStorage # Для управління станами діалогів

API_TOKEN отримується від @BotFather. У реальному проєкті його слід зберігати в змінних середовища.

API_TOKEN = '7599198811:AAGQ_rurdlArgcNWK1vLbIMyNkuUMPT-0XA' #

Ініціалізація об'єкта Bot за допомогою отриманого токена

bot = Bot(token=API_TOKEN)=

Ініціалізація сховища для Finite State Machine (FSM).

MemoryStorage підходить для невеликих проєктів та тестування.

Для продакшену варто використовувати RedisStorage або інші рішення.

storage = MemoryStorage()

Ініціалізація Dispatcher, який буде обробляти всі вхідні оновлення від Telegram.

Передаємо йому об'єкт Bot та сховище для FSM.

dp = Dispatcher(bot, storage=storage)

Приклад реєстрації простого обробника для команди /start:

@dp.message_handler(commands=['start'])

async def cmd_start(message: types.Message):

Обробник команди /start.

Відправляє привітання користувачеві.

Для запуску бота в асинхронному режимі використовується asyncio.run(dp.start_polling())

Цей код зазвичай розміщується в основному файлі бота.

У порівнянні з іншою популярною бібліотекою для розробки Telegram-ботів на Python, python-telegram-bot, бібліотека aiogram має кілька суттєвих переваг, що стали вирішальними для мого вибору. По-перше, як вже зазначалося, асинхронність aiogram забезпечує значно швидшу та ефективнішу обробку запитів. Це означає, що якщо, наприклад, 10 учнів одночасно надсилають різні команди, aiogram може обробити їх практично паралельно, не створюючи "черги" та затримок. Це критично важливо для підтримки високої

продуктивності та користувацького досвіду в умовах інтенсивного використання освітнього бота. По-друге, aiogram пропонує зручні та потужні інструменти для створення складних користувацьких інтерфейсів, зокрема гнучкі механізми для створення клавіатур (як звичайних, що з'являються під полем введення, так і inline-клавіатур, вбудованих безпосередньо в повідомлення) та ефективну підтримку складних діалогових сценаріїв за допомогою Finite State Machine (FSM). FSM дозволяє легко реалізовувати багатоетапні діалоги, наприклад, процес реєстрації нового учня, де бот послідовно запитує ім'я, групу, контактні дані, зберігаючи стан користувача між запитами. Це спрощує розробку інтерактивних та багатофункціональних ботів, роблячи їх більш "людяними" та зрозумілими. Хоча асинхронне програмування вимагає розуміння специфічних концепцій, таких як ключові слова async та await, я вважаю, що це є виправданим інвестицією в ефективність та масштабованість проєкту.

У моєму проєкті бібліотека aiogram використовується для реалізації всіх ключових функцій бота: від обробки базових команд, таких як /start та /help, до створення та управління динамічними клавіатурами з кнопками для навігації по функціоналу. Вона є основою для реалізації складних сценаріїв, таких як запис на гуртки, де користувач обирає з доступних варіантів, а також для розробки адмін-панелі, яка дозволяє адміністраторам керувати розкладом, додавати нових користувачів або надсилати оголошення. Наприклад, коли користувач натискає кнопку "🖩 Переглянути розклад", aiogram обробляє цю подію (CallbackQuery) і викликає відповідну асинхронну функцію, яка зчитує дані з файлу schedule.txt та повертає їх користувачеві у вигляді форматованого повідомлення. Завдяки своїй швидкості, гнучкості та багатому функціоналу, aiogram стала надійною та ефективною основою для розробки мого освітнього бота, дозволивши мені реалізувати всі заплановані можливості.

2.3 Інструменти та технологічний стек

Для розробки мого освітнього бота я обрала мову програмування Python версії 3.11. Цей вибір обґрунтований тим, що Python є однією з найпопулярніших та найдинамічніших мов програмування у світі, що зумовлено її простотою синтаксису, високою читабельністю коду та надзвичайно великою кількістю доступних бібліотек для різноманітних завдань. Простота Python дозволяє розробникам, швидко писати функціональний код, що є критично важливим для студентського проєкту з обмеженими часовими рамками. Його синтаксис, що нагадує природну мову, значно спрощує процес розробки та налагодження, а також сприяє легшому розумінню коду іншими розробниками, що важливо для потенційного розвитку проєкту в майбутньому. Крім того, Python має потужну спільноту підтримки та велику кількість ресурсів для навчання, що було корисним під час самостійної розробки.

Однією з ключових переваг Python, яку я активно використовувала у своєму проєкті, є його ефективність у роботі з файлами та даними. Наприклад, операції зчитування та запису структурованих даних у файли займають лише кілька рядків коду, що демонструє лаконічність та потужність мови:
import json - Імпортуємо стандартну бібліотеку для роботи з JSON-даними
import os - Імпортуємо бібліотеку для взаємодії з операційною системою (для перевірки існування файлу)

Функція для завантаження даних з JSON-файлу має такий вигляд

```
def load_data(filename):
    if not os.path.exists(filename):
        # Якщо файл не існує, повертаємо порожній словник, щоб уникнути помилок
        return {}
    with open(filename, "r", encoding="utf-8") as f:
        return json.load(f)
# Функція для збереження даних у JSON-файл
```

```
def save_data(filename, data):
    with open(filename, "w", encoding="utf-8") as f:
        # json.dump серіалізує Python-об'єкт в JSON-формат
        # ensure_ascii=False дозволяє коректно зберігати українські символи
        # indent=2 робить JSON-файл більш читабельним, додаючи відступи
        json.dump(data, f, ensure_ascii=False, indent=2)
# Приклад використання:
users_data = load_data("users.json") # Завантажуємо дані користувачів
print("Завантажені дані користувачів:", users_data)
# Додамо нового користувача або оновимо існуючого
{
  "899240710": {
    "name": "Іванка",
    "group": ":art: Малювання",
    "contact": "@water_in_your_brain"
  },
  "899240710": {
    "name": "Іванка",
    "group": null,
    "contact": "@water_in_your_brain"
  },
  "413977352": {
    "name": "Даня",
    "group": ":notes: Музика",
    "contact": "@zipyliaD"
  },
  "590048178": {
    "name": "Sesha",
    "group": null,
    "contact": "@yedrog"
```

```
  }
}
save_data("users.json", users_data) # Зберігаємо оновлені дані
print("Дані користувачів збережено.")
# Приклад роботи з текстовим файлом для розкладу
def load_schedule(filename):
    if not os.path.exists(filename):
        return "Розклад ще не завантажено."
    with open(filename, "r", encoding="utf-8") as f:
        return f.read()
def save_schedule(filename, schedule_text):
    with open(filename, "w", encoding="utf-8") as f:
        f.write(schedule_text)
schedule_content = load_schedule("schedule.txt")
print("\nЗавантажений розклад:\n", schedule_content)
new_schedule = """Понеділок:
  10:00 - Малювання (група А)
  14:00 - Англійська мова (група Б)
Вівторок:
  11:00 - Робототехніка (група А)
  15:00 - Музика (група Б)
Середа:
  10:00 - Малювання (група Б)
  16:00 - Шахи (група А)
save_schedule("schedule.txt", new_schedule)
print("Розклад оновлено.")
```

Для зберігання даних у моєму проєкті на поточному етапі використовуються локальні файли. Зокрема, інформація про користувачів (їхні імена, обрані гуртки, контактні дані, а також статус адміністратора) зберігається у файлі users.json. Формат JSON (JavaScript Object Notation) був обраний через

його простоту, високу читабельність для людини та легкість парсингу та серіалізації у Python за допомогою вбудованої бібліотеки json. Приклад структури даних у users.json виглядає наступним чином, демонструючи, як я організувала інформацію про кожного користувача за його унікальним Telegram ID:


```
{
 "899240710": {
  "name": "Іванка",
  "group": ":art: Малювання",
  "contact": "@water_in_your_brain"
 },
 "899240710": {
  "name": "Іванка",
  "group": null,
  "contact": "@water_in_your_brain"
 },
 "413977352": {
  "name": "Даня",
  "group": ":notes: Музика",
  "contact": "@zipyliaD"
 },
 "590048178": {
  "name": "Sesha",
  "group": null,
  "contact": "@yedrog"
 },
 "7848346857": {
  "name": "Elizabet",
  "group": ":notes: Музика",
  "contact": "@lizyunau"  }  }
```

Для зберігання розкладу занять використовується простий текстовий файл schedule.txt. Цей формат є оптимальним для лінійних, неструктурованих даних, які легко читаються та редагуються вручну адміністратором. Я обрала цей формат для простоти реалізації на початковому етапі, оскільки він не вимагає складних парсерів і дозволяє швидко оновлювати розклад. Приклад вмісту schedule.txt:

Понеділок:

10:00 - Малювання (група А, викладач: Олена Петрівна)

14:00 - Англійська мова (група Б, викладач: Анна Іванівна)

Вівторок:

11:00 - Робототехніка (група А, викладач: Сергій Миколайович)

15:00 - Музика (група Б, викладач: Ірина Василівна)

Середа:

10:00 - Малювання (група Б, викладач: Олена Петрівна)

16:00 - Шахи (група А, викладач: Дмитро Олександрович)

Хоча для великих та масштабованих проєктів використання повноцінних систем управління базами даних (СУБД), таких як SQLite (для вбудованих або невеликих застосунків), PostgreSQL (для потужних, багатокористувацьких веб-додатків) або MongoDB (для NoSQL рішень), було б значно ефективнішим з точки зору цілісності даних, швидкості доступу, підтримки конкурентного доступу та можливостей масштабування, на початковому етапі розробки та для цілей студентського проєкту було прийнято рішення використовувати файлову систему. Це дозволило спростити реалізацію, уникнути необхідності налаштування та адміністрування бази даних, а також значно прискорити процес прототипування, що було важливим для мене, як для розробниці, що працює самостійно. Однак, я чітко усвідомлюю, що для подальшого розвитку та комерційного використання проєкту, перехід на СУБД є обов'язковим. Це забезпечить кращу масштабованість, надійність зберігання даних, підтримку багатокористувацького доступу, складних запитів та ефективне резервне копіювання.

Для запуску та тестування бота на поточному етапі я використовую локальний комп'ютер під управлінням операційної системи Windows 11. Після встановлення Python та бібліотеки aiogram, бот запускається за допомогою простої команди в терміналі: python my_school_bot.py. Після цього бот встановлює з'єднання з Telegram API та починає обробляти вхідні запити від користувачів. Для забезпечення цілодобової та безперебійної роботи бота в реальних умовах, у майбутньому планується його розгортання на хмарному сервісі. Одним з найбільш підходящих варіантів є Heroku — платформа як послуга (PaaS), яка дозволяє легко розгортати та масштабувати веб-додатки та ботів без необхідності глибокого занурення в налаштування серверної інфраструктури. Процес розгортання на Heroku включає завантаження коду проєкту до репозиторію Git, налаштування файлу Procfile для визначення команди запуску бота, а також встановлення змінних середовища (наприклад, для зберігання API-токену бота) для забезпечення безпеки та гнучкості конфігурації. Це дозволить моєму боту бути доступним 24/7 для всіх користувачів.

Додаткові інструменти, що використовуються мною в процесі розробки, включають інтегроване середовище розробки VS Code (Visual Studio Code). Це потужний та гнучкий редактор коду, який надає широкі можливості для написання, налагодження та тестування коду завдяки підтримці розширень (наприклад, для Python, GitLens), вбудованому терміналу та інструментам для контролю версій (інтеграція з Git). Сам Telegram-клієнт також є невід'ємною частиною процесу, оскільки він використовується для безпосереднього тестування функціоналу бота та взаємодії з ним з точки зору кінцевого користувача, дозволяючи мені швидко перевіряти зміни та виправляти помилки.

Цей технологічний стек є оптимальним для поточного етапу проєкту, оскільки він дозволяє швидко створити функціональний прототип бота з мінімальними витратами ресурсів та часовими затратами, забезпечуючи при цьому достатню гнучкість для подальшого розвитку.

Детальний аналіз та обґрунтування вибору технологій, представлені у цьому розділі, переконливо демонструють, що обраний мною стек повністю відповідає поставленим завданням та потребам проєкту зі створення освітнього бота. Я, як розробниця, впевнена, що платформа Telegram виступає як ідеальне середовище для взаємодії з користувачами. Її інтуїтивно зрозумілий інтерфейс, широка популярність серед цільової аудиторії (учнів, викладачів, адміністраторів) та кросплатформність значно знижують поріг входу та сприяють швидкому та безболісному впровадженню бота в повсякденну освітню практику. Можливості Telegram Bot API дозволяють створювати високоінтерактивні та функціональні боти з кастомними клавіатурами та підтримкою різноманітних команд, що є ідеальним для реалізації таких ключових функцій, як відображення актуального розкладу занять, організація процесу запису на додаткові гуртки та надання адміністративного функціоналу.

Бібліотека aiogram, написана на Python, стала фундаментальною основою для розробки програмної логіки мого бота. Її архітектура, заснована на асинхронному програмуванні з використанням asyncio, забезпечує виняткову продуктивність та здатність обробляти велику кількість одночасних запитів від користувачів без затримок. Це гарантує високу швидкість відгуку бота навіть в умовах інтенсивного використання, що є критично важливим для забезпечення позитивного користувацького досвіду в освітньому середовищі. Гнучкість aiogram у створенні складних інтерфейсів та реалізації багатоетапних діалогів за допомогою Finite State Machine (FSM) значно спрощує розробку інтерактивних функцій, таких як реєстрація користувачів або послідовний запис на гуртки, дозволяючи мені створювати складну логіку відносно простими засобами.

Мій вибір Python як основної мови програмування є стратегічно виправданим завдяки його високій читабельності, простоті синтаксису та багатій екосистемі бібліотек. Це дозволяє здійснювати швидке прототипування та ефективну розробку, що є особливо цінним для студентського проєкту, який я виконувала самостійно.

Робота з файлами у Python, зокрема форматами JSON для структурованих даних (інформація про користувачів) та простими текстовими файлами для лінійних даних (розклад), виявилася зручним та швидким рішенням на початковому етапі розробки. Це дозволило мені зосередитися на логіці бота, мінімізуючи час на налаштування інфраструктури.

Однак, я усвідомлюю, що для забезпечення довгострокової масштабованості, надійності та цілісності даних у майбутньому, мій проєкт передбачає перехід від файлового зберігання до повноцінної системи управління базами даних, наприклад, SQLite для локального використання або PostgreSQL для хмарного розгортання. Це дозволить ефективно керувати великими обсягами даних, забезпечити конкурентний доступ та реалізувати складніші запити, що буде наступним етапом розвитку моєї роботи.

Поточний локальний запуск бота на моєму комп'ютері з Windows 11 є оптимальним для фази розробки та тестування, дозволяючи оперативно вносити зміни та налагоджувати функціонал. Проте, для забезпечення стабільної та цілодобової роботи бота в реальних умовах експлуатації, необхідним кроком є його розгортання на хмарному сервісі, такому як Heroku. Хмарна платформа гарантує високу доступність, автоматичне масштабування та надійну інфраструктуру, звільняючи мене від необхідності постійно підтримувати локальний сервер.

Таким чином, обраний мною технологічний стек, що включає Telegram як платформу взаємодії, aiogram як фреймворк для розробки бота, Python як мову програмування та стратегію поетапного розгортання та масштабування даних, є оптимальним для створення ефективного, зручного та перспективного бота для управління освітніми просторами.

3 ОПИС ПРОГРАМНОГО ПРОДУКТУ

3.1 Головне меню та логіка взаємодії користувача з ботом

Розроблений Telegram-бот є інноваційним інструментом, спрямованим на автоматизацію управління освітніми просторами, зокрема позашкільними дитячими гуртками. Його логіка взаємодії базується на інтуїтивно зрозумілому головному меню, яке об’єднує чітко визначені функціональні модулі, кожен із яких відповідає за виконання конкретних завдань. Такий підхід забезпечує простоту використання для користувачів із різним рівнем технічної підготовки, включаючи батьків, учнів та адміністраторів.

Бот реалізовано на мові програмування Python із використанням бібліотеки Aiogram, що дозволяє ефективно обробляти асинхронні запити та забезпечувати швидку реакцію навіть при високому навантаженні. Дані зберігаються у файлах users.json (для інформації про користувачів) та schedule.txt (для розкладу), що є простим, але ефективним рішенням для невеликих освітніх закладів.

Головне меню бота розроблено з урахуванням принципів UX-дизайну, що забезпечує логічну навігацію та мінімізує когнітивне навантаження на користувача.

Кожна опція меню чітко позначена та супроводжується емодзі для візуальної диференціації, що підвищує зручність сприйняття.

Особливості логіки взаємодії: Система передбачає розгалужену логіку діалогів залежно від дій користувача. Після вибору бот зберігає інформацію у словнику users і повертається до головного меню (Таблиця 3.1). У випадку з опцією "📝 Зареєструватися" бот перевіряє наявність обраного гуртка; якщо вибір відсутній, користувачу надсилається відповідне повідомлення з проханням обрати гурток. Реалізація асинхронного оброблення запитів за допомогою

бібліотеки aiogram забезпечує швидке реагування (менше 2 секунд) і можливість одночасної роботи з кількома користувачами.

Таблиця 3.1 - Логіка взаємодії Telegram-бота

Пункт меню	Призначення	Запити/дії користувача
Обрати гурток	Дозволяє користувачу обрати гурток	Вибір із доступних гуртків (наприклад, "Малювання")
Переглянути розклад	Надання доступу до актуального розкладу	Натиснення кнопки "Переглянути розклад"
Зареєструватися	Завершення процесу реєстрації на гурток	Натиснення кнопки після вибору гуртка
? Допомога	Надання інструкцій для користувачів	Натиснення кнопки "? Допомога"
Зв'язатися з адміністратором	Забезпечення зв’язку з адміністратором	Натиснення кнопки
Адмін-панель	Надання доступу до адміністративних функцій	Введення пароля "Пароль: mysecretpass"

Збереження стану та цілісність даних: Кожна дія користувача супроводжується оновленням стану у файловій системі (файли users.json та schedule.txt), що дозволяє підтримувати цілісність даних і забезпечувати доступ до історії взаємодії. Наприклад, при реєстрації дані користувача, включаючи user_id, ім’я, обраний гурток і контакт, зберігаються у форматі JSON. Адміністративний доступ до повного розкладу та списку користувачів реалізовано через модуль admin_panel.py, що гарантує централізоване управління системою.

Переваги моделі: Застосування такої логіки взаємодії дозволяє автоматизувати рутинні процеси, зменшити навантаження на адміністраторів і

підвищити рівень комфорту для користувачів. Гнучкість системи забезпечується можливістю скасування дій на будь-якому етапі діалогу та повернення до головного меню, що є важливим для клієнтоорієнтованих сервісів у сфері освіти.

3.2 Логічна структура бази даних

Для забезпечення ефективного зберігання та обробки даних розроблено логічну структуру бази даних, яка на поточному етапі реалізовано через файлову систему, але спроєктовано з урахуванням майбутнього переходу до реляційної моделі, наприклад, SQLite.

Логічна структура включає ключові сутності, їхні атрибути та зв’язки, що відповідають принципам нормалізації для уникнення дублювання та забезпечення логічної цілісності.

Опис основних сутностей:

- Сутність "Користувач" (Users):

- Атрибути:

- user_id (INTEGER) — унікальний ідентифікатор користувача, отриманий із Telegram.

- name (TEXT) — ім’я користувача, отримане з профілю.

- group (TEXT) — назва обраного гуртка, наприклад, "🎨 Малювання".

-contact (TEXT) — контактна інформація, наприклад, "@water_in_your_brain".

Призначення: Зберігання інформації про користувачів, які взаємодіють із ботом.

- Сутність "Розклад" (Schedule):

- Атрибути:

- day (TEXT) — день тижня, наприклад, "Понеділок".

- time (TEXT) — час занять, наприклад, "16:00 - 17:30".

- group (TEXT) — назва гуртка, до якого відноситься розклад.

Призначення: Зберігання даних про розклад занять для всіх доступних гуртків.

Зв’язки між сутностями:

Сутність "Користувач" пов’язана з сутністю "Розклад" через атрибут group, що дозволяє визначити, які заняття доступні для конкретного користувача залежно від його вибору.

У поточній реалізації зв’язок реалізовано опосередковано через зіставлення даних у файлах users.json та schedule.txt. Наприклад, користувач із user_id "899240710" обрав гурток "🎨 Малювання", і система може отримати відповідний розклад із schedule.txt.

Поточна реалізація: На даному етапі дані зберігаються у двох файлах:

users.json: Містить словник із даними користувачів, де ключем є user_id, а значенням — об’єкт із атрибутами:

```
{
  "899240710": {
    "name": "Іванка",
    "group": "🎨 Малювання",
    "contact": "@water_in_your_brain"
  }
}
```

schedule.txt: Зберігає розклад у текстовому форматі:

- 🎨 Малювання:
- Понеділок: 16:00 - 17:30
- Середа: 16:00 - 17:30

Перспективи розвитку: Перехід до реляційної бази даних дозволить усунути обмеження файлової системи, такі як можливе дублювання записів (наприклад, два записи для "Іванки" у users.json) та повільність при великій кількості користувачів.

Логічна структура підготовлена для реалізації таблиць із зовнішніми ключами, що забезпечить швидкий доступ до даних і їхню цілісність (Таблиця 3.2 Сутність "Користувачі”. і Таблиця 3.3. Сутність "Розклад".

Таблиця 3.2 - Сутність "Користувачі”.

Назва поля	Тип даних	Опис
user_id	INTEGER	Унікальний ідентифікатор користувача
name	TEXT	Ім’я користувача
group	TEXT	Обраний гурток
contact	TEXT	Контактна інформація

Таблиця 3.3 - Сутність "Розклад"

Назва поля	Тип даних	Опис
day	TEXT	День тижня
time	TEXT	Час занять
group	TEXT	Назва гуртка

3.3 Фізична модель бази даних

Фізична модель бази даних у даному проєкті описує поточну реалізацію зберігання даних, яка базується на файловій системі з використанням двох основних файлів: users.json і schedule.txt. Ця модель забезпечує базові потреби системи в управлінні інформацією про користувачів та розклад занять, дозволяючи ефективно обробляти запити в умовах невеликої кількості даних. Нижче представлено детальний опис структури, вмісту та функціонального призначення цих файлів, а також їхньої ролі в роботі Telegram-бота.

Поточна реалізація:

- Файл "users.json":

- Формат: JSON (JavaScript Object Notation) — структурований текстовий формат для зберігання даних у вигляді словника.

Структура: Файл містить словник, де ключем є user_id (унікальний ідентифікатор користувача в Telegram, тип — рядок), а значенням — об’єкт із атрибутами користувача:

- name (TEXT) — ім’я користувача, отримане з профілю Telegram.

- group (TEXT) — назва обраного гуртка, наприклад, "🎨 Малювання".

- contact (TEXT) — контактна інформація, наприклад, Telegram-юзернейм у форматі "@username".

Приклад вмісту:


```
{
  "899240710": {
    "name": "Іванка",
    "group": "🎨 Малювання",
    "contact": "@water_in_your_brain"
  },
  "413977352": {
    "name": "Даня",
    "group": "🎶 Музика",
    "contact": "@zipyliaD"
  }
}
```


Призначення: Зберігання інформації про всіх користувачів, які взаємодіють із ботом. Дані оновлюються щоразу, коли користувач обирає гурток або реєструється, за допомогою функцій load_data і save_data з модуля file_storage.py.

Особливості: Формат JSON забезпечує легкість читання та запису, однак відсутність механізмів перевірки унікальності призводить до можливого дублювання записів (наприклад, два записи для одного user_id).

Файл "schedule.txt":

Формат: Простий текстовий файл із структурованим вмістом.

Структура: Файл містить розклад занять, організований за гуртками. Кожен гурток представлений назвою, за якою слідують дні тижня та час занять:

- Назва гуртка (TEXT) — наприклад, "🎨 Малювання".
- День тижня та час (TEXT) — наприклад, "Понеділок: 16:00 - 17:30".

Приклад вмісту:

- 🎨 Малювання:
- Понеділок: 16:00 - 17:30
- Середа: 16:00 - 17:30
- 🎶 Музика:
- Вівторок: 15:30 - 17:00
- Четвер: 15:30 - 17:00

Призначення: Зберігання розкладу занять для всіх гуртків, що дозволяє користувачам переглядати доступний час, а адміністраторам — оновлювати його. Доступ до файлу забезпечується функціями get_schedule і update_schedule з модуля schedule_utils.py.

Особливості: Текстовий формат є простим для редагування, але не підтримує складні запити чи автоматичну перевірку формату введених даних.

Зв’язки між файлами:

Дані у файлі users.json пов’язані з schedule.txt через поле group. Наприклад, якщо користувач із user_id "899240710" обрав "🎨 Малювання", система може отримати розклад для цього гуртка з schedule.txt.

Функціонально зв’язок реалізовано через програмну логіку: при запиті розкладу бот читає schedule.txt і фільтрує дані за значенням group із users.json.

Робота з даними:

Читання та запис: Модуль file_storage.py забезпечує завантаження (load_data) і збереження (save_data) даних у users.json. Аналогічно, schedule_utils.py відповідає за доступ до schedule.txt.

Обробка помилок: У разі відсутності файлу schedule.txt бот повертає повідомлення "Розклад ще не створено." Для users.json передбачено ініціалізацію порожнього словника при першому запуску.

Переваги та обмеження:

Переваги: Поточна файлова система є простою у реалізації та достатньою для невеликої кількості користувачів (до 50-100 записів). Вона не потребує додаткових інструментів, таких як сервер бази даних, і легко інтегрується з Python через стандартні бібліотеки.

Обмеження: Відсутність механізмів цілісності даних може призводити до дублювання записів у users.json. Текстовий формат schedule.txt ускладнює фільтрацію чи сортування даних, а обробка великої кількості записів може бути повільною через послідовне читання файлів.

Сутність "Groups" (Гуртки):

Представлення у файловій системі: Сутність "Groups" не має окремого файлу для зберігання, але її дані інтегровані у файл schedule.txt, де назви гуртків використовуються як заголовки для групування розкладу. Додатково список доступних гуртків визначено у коді бота, зокрема в модулі user_handlers.py, у вигляді клавіатури group_kb: Таким чином, "гуртки" представлені як перелік назв, доступних для вибору користувачем.

Поля (аналогія):

id :Унікальний ідентифікатор гуртка не зберігається явно, але може бути представлений внутрішньою логікою програми (наприклад, як індекс у списку гуртків). У поточній реалізації ідентифікатор не використовується, оскільки зв’язок із розкладом базується на назві гуртка.

Name: Назва гуртка, наприклад, "🎨 Малювання", яка є обов’язковою і використовується як ключ для ідентифікації гуртка у файлі schedule.txt та словнику users у users.json.

description: Опис гуртка у поточній реалізації відсутній, але може бути доданий як коментар у коді або у вигляді окремого текстового файлу в майбутньому.
У файлі schedule.txt назви гуртків виступають як заголовки:

- 🎨 Малювання:

- Понеділок: 16:00 - 17:30

- Середа: 16:00 - 17:30

- 🎶 Музика:

- Вівторок: 15:30 - 17:00

- Четвер: 15:30 - 17:00

Призначення: Сутність "Groups" забезпечує каталог гуртків, доступних для вибору користувачами. Назви гуртків використовуються для зв’язку з розкладом у schedule.txt та для збереження вибору користувача у users.json.
Сутність "Schedule" (Розклад):

Представлення у файловій системі: Сутність "Schedule" безпосередньо реалізована у файлі schedule.txt, який містить розклад занять, структурований за гуртками. Кожен запис у файлі пов’язаний із конкретним гуртком через його назву.

Поля (аналогія):

- id (еквівалент у коді): Унікальний ідентифікатор запису розкладу відсутній, оскільки файлова система не підтримує автоматичну генерацію ідентифікаторів. Кожен запис ідентифікується за комбінацією назви гуртка, дня тижня та часу.

- group_id (еквівалент у коді): Ідентифікатор гуртка замінений його назвою (наприклад, "🎨 Малювання"), яка використовується як заголовок у schedule.txt і співвідноситься з полем group у users.json.

- day (TEXT, NOT NULL): День тижня, наприклад, "Понеділок", є обов’язковим і зазначається у кожному записі розкладу.

- time (TEXT, NOT NULL): Час занять, наприклад, "16:00 - 17:30", є обов’язковим і вказується поряд із днем тижня.

Приклад представлення: У файлі schedule.txt розклад виглядає так:

🎨 Малювання:

- Понеділок: 16:00 - 17:30

- Середа: 16:00 - 17:30

 Робототехніка:

- П’ятниця: 16:00 - 18:00

- Субота: 10:00 - 12:00

Призначення: Сутність "Schedule" забезпечує зберігання розкладу занять для кожного гуртка. Користувачі можуть переглядати розклад за допомогою команди "📅 Переглянути розклад", а адміністратори — оновлювати його через функцію "✏️ Змінити розклад". Дані доступні для фільтрації за назвою гуртка через програмну логіку.

Зв’язки між сутностями:

У поточній реалізації зв’язок між "Groups" і "Schedule" забезпечується через назву гуртка (name), яка є заголовком у schedule.txt. Наприклад, якщо користувач обрав "🎨 Малювання" (зберігається у users.json у полі group), бот може отримати відповідний розклад із schedule.txt, використовуючи цю назву як ключ для пошуку.

Робота з даними:

Читання та запис: Функції get_schedule і update_schedule у модулі schedule_utils.py відповідають за читання та оновлення файлу schedule.txt. Наприклад, при оновленні розкладу адміністратором старий вміст файлу перезаписується новим текстом.

Обмеження: Відсутність унікальних ідентифікаторів (id) ускладнює точну ідентифікацію записів, а текстовий формат не підтримує складні запити чи автоматичну валідацію даних.

Переваги та обмеження:

3.4 Тестування Telegram-бота і екранні зображення

Для забезпечення коректної роботи Telegram-бота було розроблено низку сценаріїв тестування, які охоплюють основні функції системи. Тестування проводилося в реальному середовищі Telegram із залученням тестових користувачів.

Кожен сценарій включає опис дій, очікувані результати та ілюстрації екранних зображень для демонстрації функціоналу.

Сценарій №1: Вхід адміністратора та перегляд/підтвердження записів

Таблиця 3.4 - Сценарій №1 Вхід адміністратора та перегляд/підтвердження записів

Дія користувача	Очікувана відповідь системи
/start	Вітаємо! Оберіть дію з меню
Панель адміна (Рис. 1 Виклик кнопки «Адмін-панель»)	🔐 Введіть пароль для входу до адмін панелі:Формат: Пароль: ваш_пароль
Пароль: mysecretpass(Рис. 2 Введення паролю адміністратора)	✅ Вхід до адмін панелі успішний!
✅ Вхід до адмін панелі успішний!(Рис. 3 Успішність входу)	Доступ до кнопок «Змінити розклад» та «Переглянути всіх користувачів»
Змінити розклад(Рис. 4 Кнопка зміни розкладу)	📝 Введіть новий розклад
Переглянути всіх користувачів» (Рис. 5 Кнопка перегляду користувачів)	Список із зареєстрованими користувачами

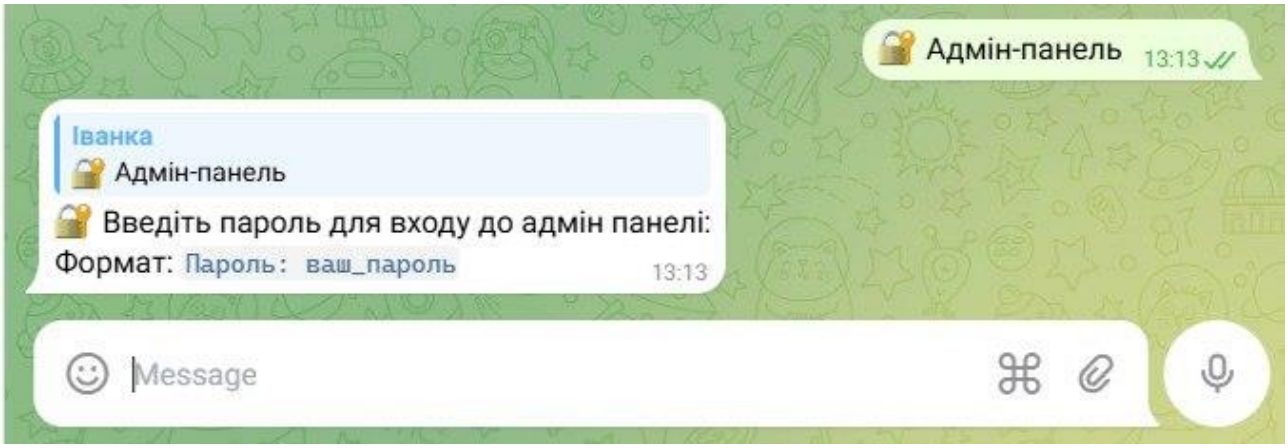


Рисунок 3.1 - Виклик кнопки «Адмін-панель»

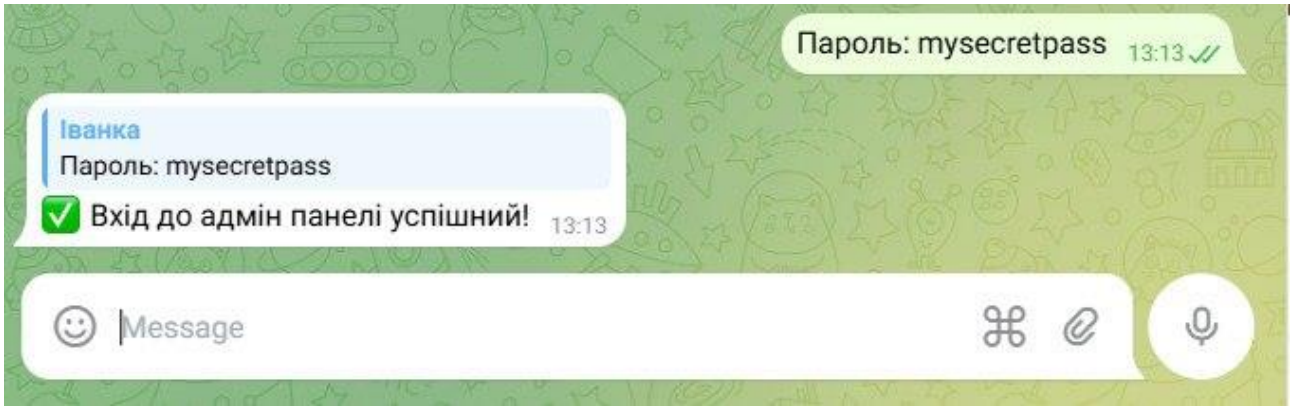


Рисунок 3.2 - Введення паролю адміністратора

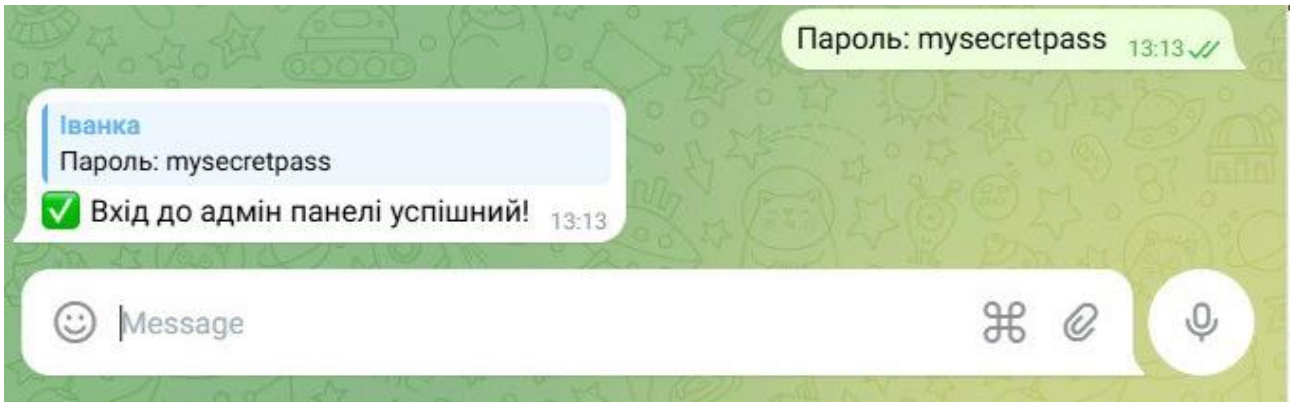


Рисунок 3.3 - Успішність входу

Рисунок 3.4 - Кнопка зміни розкладу

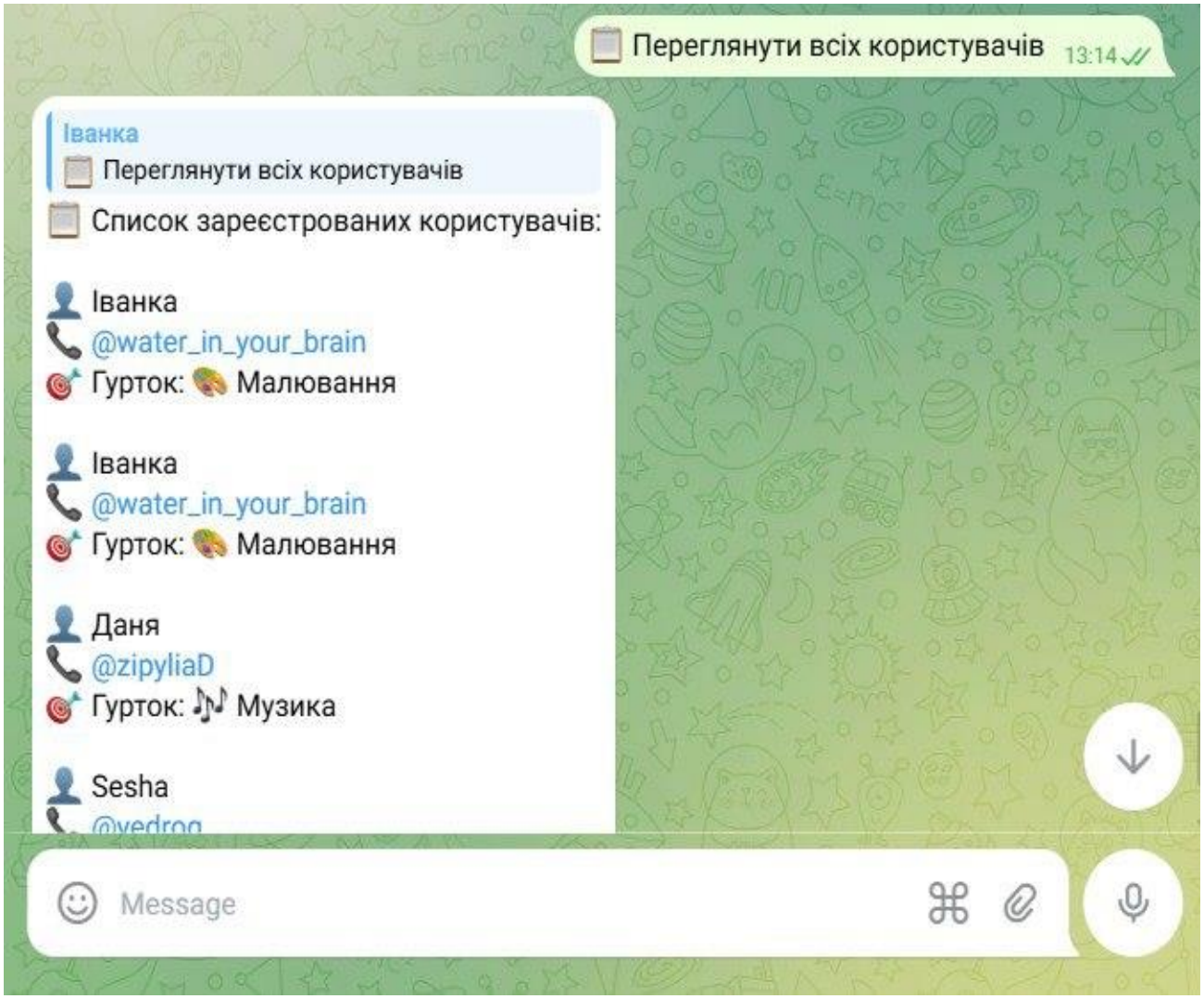


Рисунок 3.5 - Кнопка перегляду користувачів

Сценарій №2: Реєстрація нового користувача

Таблиця 3.5 - Сценарій №2 Реєстрація нового користувача

Дія користувача	Очікувана відповідь
/start(Рис. 6 Реєстрація на гурток)	Вітаємо! Оберіть дію з меню
🎯 Обрати гурток	Оберіть гурток: [список гуртків]
🎨 Малювання/Музика/Робототехніка	✅ Ви обрали гурток: 🎨 Малювання Музика/Робототехніка
📝 Зареєструватися	✅ Ви зареєстровані на гурток 🎨 Малювання!

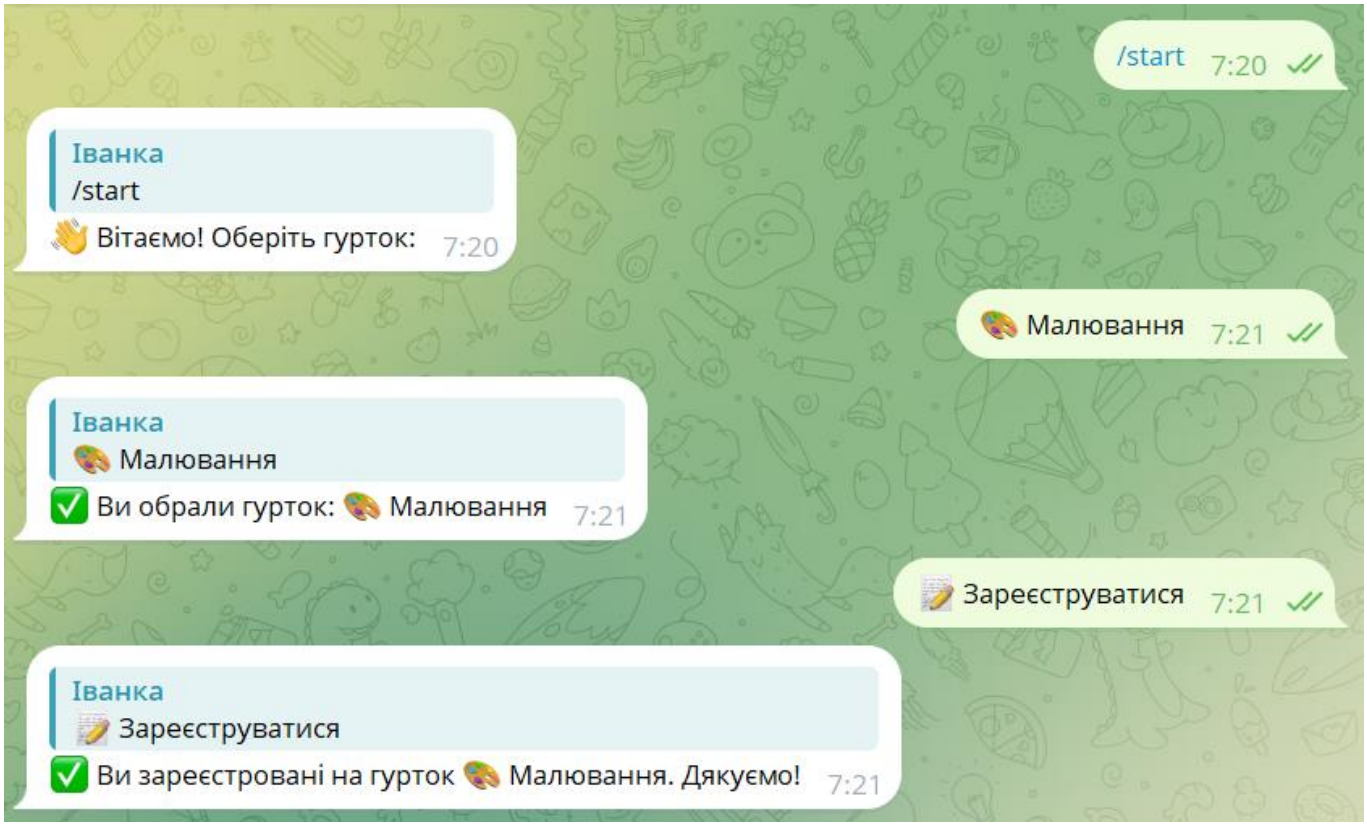


Рисунок 3.6 - Реєстрація на гурток

Сценарій №3: Перегляд розкладу

Таблиця 3.6 - Сценарій №3 Перегляд розкладу

Дія користувача	Очікувана відповідь
Переглянути розкладу (Рис. 7 Перегляд розкладу)	Розклад: [вміст schedule.txt, наприклад, Понеділок: 16:00 - 17:30]

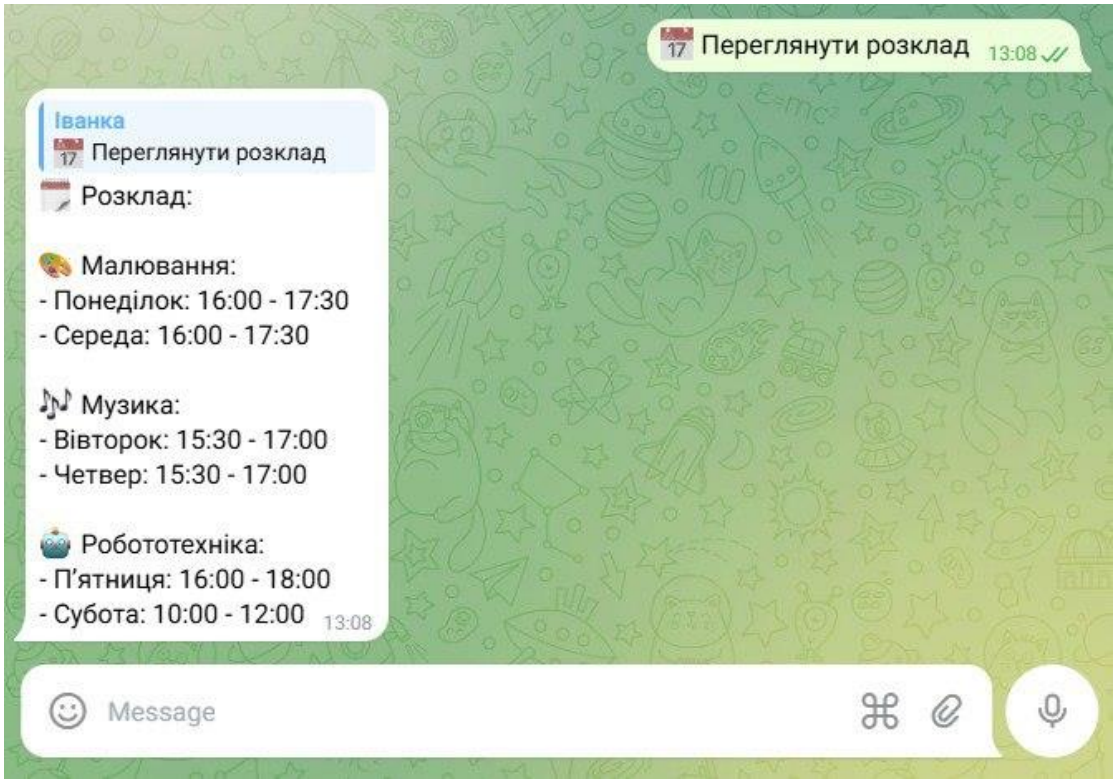


Рисунок 3.7 - Перегляд розкладу

Сценарій №4: Виклик кнопки зв'язку з адміністратором та допомоги.

Таблиця 3.7 - Сценарій №4

Дія користувача	Очікувана відповідь
Зв'язатися з адміністратором(Рис.8)	Напишіть адміну: @water_in_your_brain
? Допомога(Рис.9)	Для реєстрації оберіть гурток, а потім натисніть «Зареєструватися». Якщо виникли питання — звертайтесь до адміністратора.

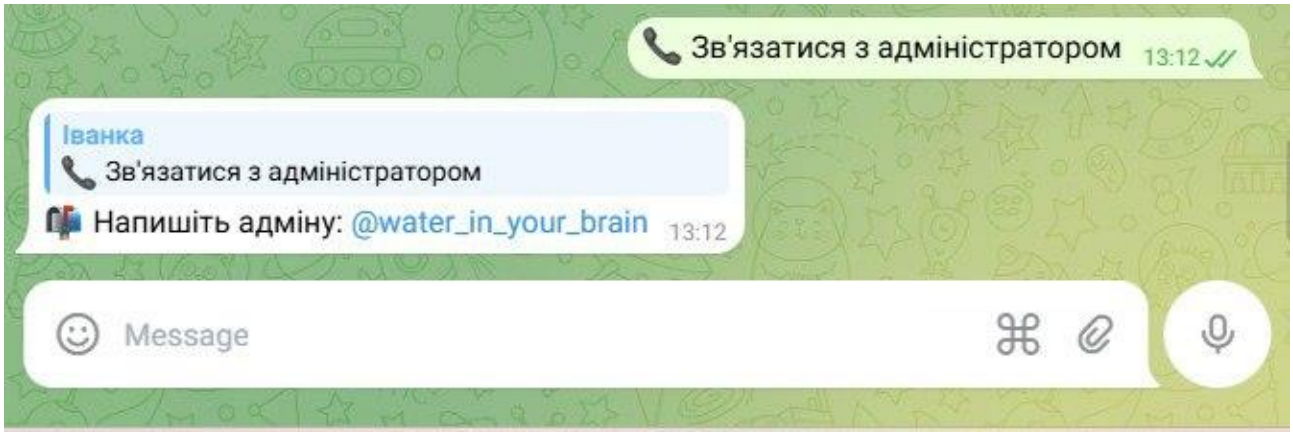


Рисунок 3.8 - Зв’язатися з адміністратором

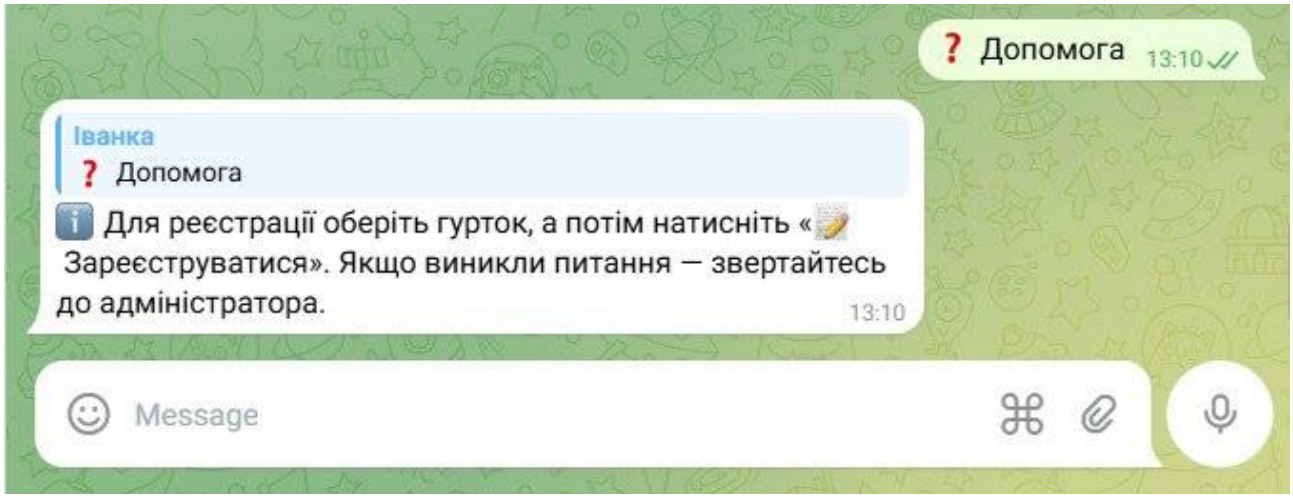


Рисунок 3.9 - Допомога з функціоналом бота

Висновки тестування: Тестування підтвердило стабільність роботи бота, відповідність функціональних вимог та зручність інтерфейсу. Виявлені проблеми, такі як зникнення клавіатури, було усунуто додаванням параметра reply_markup. Система готова до використання, хоча планується подальше вдосконалення, зокрема перехід на базу даних

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи досягнуто поставленої мети – розроблено Telegram-бот для автоматизації управління освітніми просторами, зокрема позашкільними дитячими гуртками. Робота охопила теоретичний аналіз, практичну реалізацію та тестування системи, що дозволило отримати наступні результати:

Проведено комплексний аналіз сучасних інформаційних технологій в освіті, включаючи системи управління навчанням (LMS), чат-боти та месенджери. Виявлено, що Telegram-боти є ефективним рішенням для невеликих освітніх закладів завдяки їхній простоті, економічності та інтеграції з популярною платформою.

Досліджено інформаційні процеси в позашкільних освітніх закладах, що дозволило визначити ключові потреби автоматизації, такі як спрощення комунікації між адміністрацією, батьками та учнями, а також оптимізація управління розкладом і реєстрацією.

Розроблено Telegram-бот на мові Python із використанням бібліотеки Aiogram, яка забезпечує асинхронну обробку запитів і високу продуктивність. Бот реалізує інтуїтивно зрозуміле головне меню з чітко визначеними функціональними модулями, що включають перегляд розкладу, вибір гуртків, реєстрацію, зв’язок із адміністратором, а також захищену адмін-панель для управління даними.

Забезпечено функціонал для двох груп користувачів:

- Для батьків і учнів: зручний інтерфейс для перегляду розкладу, вибору гуртків і реєстрації, що зменшує потребу в особистих візитах чи телефонних дзвінках.

- Для адміністраторів: адмін-панель із парольним доступом, яка дозволяє редагувати розклад і переглядати списки зареєстрованих користувачів, підвищуючи ефективність управління.

Проведено тестування бота, яке охопило ключові сценарії використання:

вхід адміністратора, реєстрація користувача, перегляд розкладу та зв’язок із адміністратором. Усі функції працюють коректно, а виявлені проблеми, такі як зникнення клавіатури, усунуто шляхом додавання параметра reply_markup. Тестування підтвердило відповідність бота вимогам зручності, надійності та функціональності.

Розроблена система використовує файлову систему (users.json, schedule.txt) для зберігання даних, що є економічно вигідним рішенням для невеликих закладів. Водночас визначено перспективи переходу на СУБД, такі як SQLite або PostgreSQL, для масштабування та підтримки складніших запитів.

Практична цінність роботи полягає у створенні інструменту, який значно спрощує адміністративні процеси в позашкільних освітніх закладах. Бот зменшує навантаження на персонал, підвищує доступність інформації для батьків і учнів, а також сприяє підвищенню ефективності комунікації. Розроблена система є гнучкою та може бути адаптована до потреб інших освітніх просторів.

Новизна роботи полягає у розробці спеціалізованого Telegram-бота, адаптованого до потреб позашкільної освіти, з акцентом на автоматизацію рутинних завдань і забезпечення інтуїтивного користувацького досвіду. Використання асинхронної бібліотеки Aiogram і модульної архітектури забезпечує високу продуктивність і можливість подальшого розширення функціоналу.

Перспективи подальшого розвитку включають:

- Перехід на реляційну базу даних для підвищення ефективності управління даними.

- Інтеграцію додаткових функцій, таких як автоматичні нагадування про заняття чи оплату.

- Впровадження аналітичних інструментів для моніторингу відвідуваності та популярності гуртків.

- Додавання підтримки кількох мов для залучення ширшої аудиторії.

- Посилення безпеки адмін-панелі шляхом впровадження двофакторної автентифікації.

Розроблений Telegram-бот є прикладом ефективного застосування сучасних інформаційних технологій для вирішення актуальних завдань у сфері освіти, демонструючи потенціал чат-ботів для оптимізації адміністративних процесів і покращення користувацького досвіду.

ПЕРЕЛІК ПОСИЛАНЬ

1. Telegram Bot API Documentation [Електронний ресурс]. – Режим доступу: https://core.telegram.org/bots/api.
2. Aiogram Documentation [Електронний ресурс]. – Режим доступу: https://docs.aiogram.dev/.
3. Python Official Documentation [Електронний ресурс]. – Режим доступу: https://www.python.org/doc/.
4. Moodle Official Website [Електронний ресурс]. – Режим доступу: https://moodle.org/.
5. Downey A. B. Think Python: How to Think Like a Computer Scientist. – 2nd ed. – O'Reilly Media, 2015. – 292 p.
6. Котлер Ф. Основи маркетингу / Пер. з англ. – К.: Видавництво Соломії Павличко «Основи», 2003. – 672 с.
7. Sommerville I. Software Engineering. – 10th ed. – Pearson, 2015. – 816 p.
8. Pressman R. S. Software Engineering: A Practitioner's Approach. – 8th ed. – McGraw-Hill Education, 2014. – 976 p.
9. Глушаков С. В., Ковальчук О. В. Інформаційні системи та технології: Підручник. – Х.: Фоліо, 2018. – 512 с.
10. Бойко В. В., Савинков В. М. Проектування баз даних інформаційних систем. – К.: Вища школа, 2006. – 367 с.
11. Russell S., Norvig P. Artificial Intelligence: A Modern Approach. – 4th ed. – Pearson, 2020. – 1152 p.
12. Лутц М. Изучаем Python / Пер. с англ. – 5-е изд. – М.: ДМК Пресс, 2019. – 832 с.
13. Шилдт Г. Python. Полное руководство / Пер. с англ. – М.: Вильямс, 2020. – 624 с.
14. Google Classroom Official Help [Електронний ресурс]. – Режим доступу: https://support.google.com/edu/classroom/.
15. Microsoft Teams for Education [Електронний ресурс]. – Режим доступу:

16. SQLite Official Documentation [Електронний ресурс]. – Режим доступу: https://www.sqlite.org/docs.html.

17. PostgreSQL Documentation [Електронний ресурс]. – Режим доступу: https://www.postgresql.org/docs/.

18. Tanenbaum A. S., Bos H. Modern Operating Systems. – 4th ed. – Pearson, 2014. – 1136 p.

19. Субботін С. О. Інформаційні системи: Навчальний посібник. – К.: КНЕУ, 2017. – 420 с.

20. Heroku Documentation [Електронний ресурс]. – Режим доступу: https://devcenter.heroku.com/.

21. Fowler M. Patterns of Enterprise Application Architecture. – Addison-Wesley, 2002. – 560 p.

22. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. – Addison-Wesley, 1994. – 416 p.

23. Кнут Д. Искусство программирования. Том 1. Основные алгоритмы / Пер. с англ. – М.: Вильямс, 2019. – 720 с.

24. Cormen T. H., Leiserson C. E., Rivest R. L., Stein C. Introduction to Algorithms. – 4th ed. – MIT Press, 2022. – 1312 p.

25. Ткачук М. В. Основи інформаційних технологій: Підручник. – Львів: Видавництво ЛНУ, 2016. – 380 с.

26. JSON Official Website [Електронний ресурс]. – Режим доступу: https://www.json.org/.

27. BotFather Guide [Електронний ресурс]. – Режим доступу: https://core.telegram.org/bots#6-botfather.

28. Norman D. A. The Design of Everyday Things. – Revised ed. – Basic Books, 2013. – 368 p.

29. Krug S. Don’t Make Me Think: A Common Sense Approach to Web Usability. – 3rd ed. – New Riders, 2014. – 216 p.

30. Пономаренко В. С. Інформаційні системи і технології в економіці: Навчальний посібник. – К.: Академія, 2015. – 544 с.

31. Nielsen J. Usability Engineering. – Morgan Kaufmann, 1993. – 362 p.

32. McConnell S. Code Complete: A Practical Handbook of Software Construction. – 2nd ed. – Microsoft Press, 2004. – 960 p.
33. Martin R. C. Clean Code: A Handbook of Agile Software Craftsmanship. – Prentice Hall, 2008. – 464 p.
34. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений / Пер. с англ. – М.: Бином, 2018. – 720 с.
35. Зубенко В. В. Автоматизація інформаційних процесів: Навчальний посібник. – Одеса: ОНУ, 2019. – 296 с.
36. Finite State Machines in Software Development [Електронний ресурс]. – Режим доступу: https://www.tutorialspoint.com/finite-state-machines-in-software-development.
37. Weisfeld M. The Object-Oriented Thought Process. – 5th ed. – Addison-Wesley, 2019. – 240 p.
38. Громов Ю. Ю. Інформаційні технології в освіті: Навчальний посібник. – Х.: ХНУ, 2020. – 328 с.
39. Ousterhout J. A Philosophy of Software Design. – 2nd ed. – Yaknyam Press, 2021. – 190 p.
40. EDUCBA: Introduction to Chatbots [Електронний ресурс]. – Режим доступу: https://www.educba.com/introduction-to-chatbots/.

ДОДАТОК А. ФРАГМЕНТИ ПРОГРАМНОГО КОДУ

```
from aiogram import Bot, Dispatcher, executor, types
from admin_panel import set_users_dict, process_admin_password, process_admin_message, handle_admin_login
from file_storage import load_data, save_data
from user_handlers import handle_start, handle_group_choice
import asyncio

API_TOKEN = '7599198811:AAGQ_rurdlArgcNWK1vLbIMyNkuUMPT-0XA'

bot = Bot(token=API_TOKEN)
dp = Dispatcher(bot)

users = {}

# Ініціалізація
load_data(users)            # Завантаження користувачів
set_users_dict(users)       # Передача словника у адмін-панель

@dp.message_handler(commands=['start'])
async def start_handler(message: types.Message):
    await handle_start(message, users)

@dp.message_handler(lambda message: message.text == "🔐 Адмін-панель")
@dp.message_handler(commands=['admin'])
```

```
async def admin_login_handler(message: types.Message):
    await handle_admin_login(message)

@dp.message_handler(lambda message: message.text.startswith("Пароль:")
async def admin_password_handler(message: types.Message):
    await process_admin_password(message)

@dp.message_handler()
async def general_message_handler(message: types.Message):
    # Спершу перевіряємо, чи це повідомлення для адміна
    await process_admin_message(message)
    # Потім обробляємо звичайні повідомлення користувачів
    await handle_group_choice(message, users)

if name == 'main':
    executor.start_polling(dp, skip_updates=True)



from aiogram import types
from aiogram.types import ReplyKeyboardMarkup, KeyboardButton
from schedule_utils import get_schedule, update_schedule
import json
import os

ADMIN_PASSWORD = "mysecretpass"
ADMIN_ID = None  # встановлюється після входу з паролем

admin_kb = ReplyKeyboardMarkup(resize_keyboard=True)
admin_kb.add(KeyboardButton("📋 Переглянути всіх користувачів"))
admin_kb.add(KeyboardButton("✏️ Змінити розклад"))
```

```
users = {}

SAVE_FILE = "users.json"
class AdminState:
    editing_schedule = False

def load_users():
    global users
    if os.path.exists(SAVE_FILE):
        with open(SAVE_FILE, "r", encoding="utf-8") as f:
            data = json.load(f)
            if data:
                users.update(data)

def save_users():
    with open(SAVE_FILE, "w", encoding="utf-8") as f:
        json.dump(users, f, ensure_ascii=False, indent=2)

def set_users_dict(external_users):
    global users
    users = external_users
    if not users:
        load_users()

def is_admin(user_id):
    return user_id == ADMIN_ID

async def handle_admin_login(message: types.Message):
    await message.reply(
        "🔐 Введіть пароль для входу до адмін панелі:\nФормат: `Пароль: ваш_пароль`",
```

```
        parse_mode="Markdown"
async def process_admin_password(message: types.Message):
    global ADMIN_ID
    text = message.text.strip()
    if text.startswith("Пароль:"):
        entered_password = text.replace("Пароль:", "").strip()
        if entered_password == ADMIN_PASSWORD:
            ADMIN_ID = message.from_user.id
            await message.reply("✅ Вхід до адмін панелі успішний!", reply_markup=admin_kb)
        else:
            await message.reply("❌ Невірний пароль. Спробуйте ще раз.")
    else:
        await message.reply("❗ Будь ласка, введіть пароль у форматі: `Пароль: ваш_пароль`", parse_mode="Markdown")

async def show_user_list(message: types.Message):
    if not is_admin(message.from_user.id):
        await message.reply("⛔ У вас немає доступу до цієї команди.")
        return

    if not users:
        await message.reply("Немає зареєстрованих користувачів.")
        return

    report = "📋 Список зареєстрованих користувачів:\n\n"
    for uid, info in users.items():
        name = info.get('name', 'Невідомо')
        group = info.get('group', 'Не вибрано')
        contact = info.get('contact', 'Немає')
```

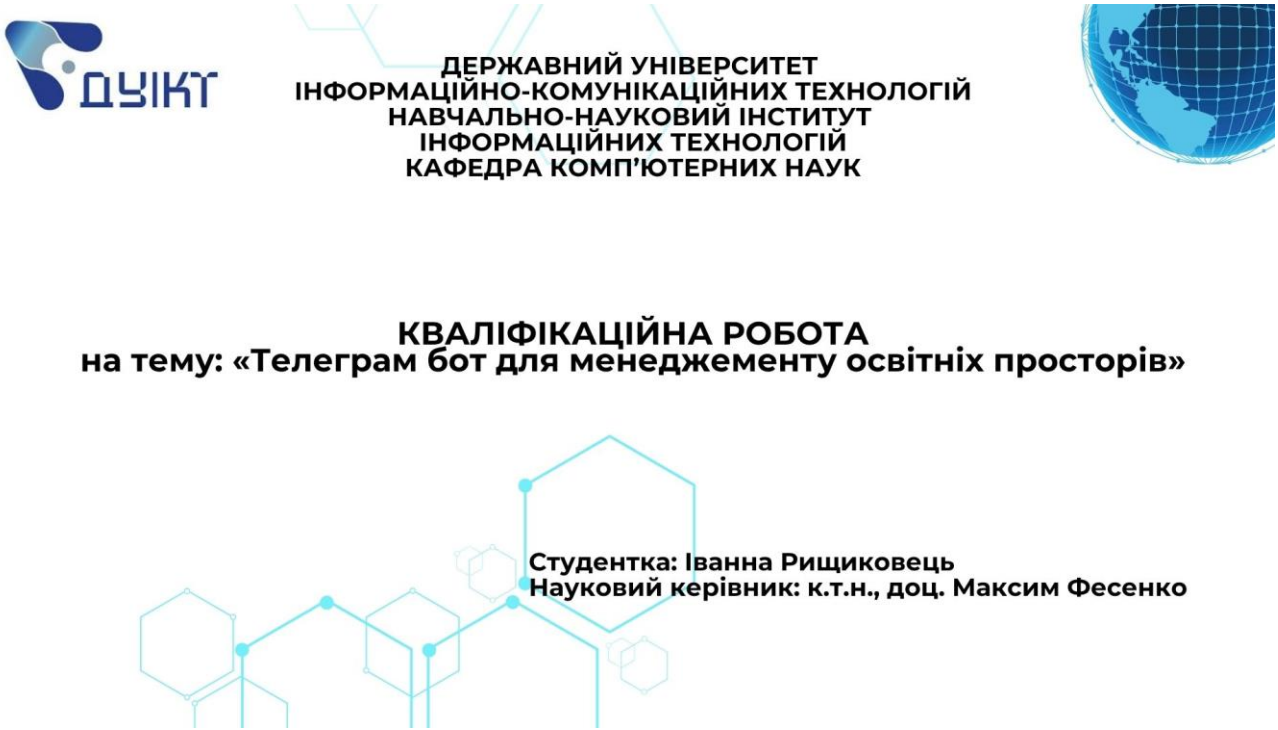
```
        report += f"👤 {name}\n📞 {contact}\n🎯 Гурток: {group}\n\n"
    await message.reply(report)

async def handle_admin_command(message: types.Message):
    if not is_admin(message.from_user.id):
        await message.reply("⛔ У вас немає доступу до цієї команди.")
        return

    if message.text == "➖ Змінити розклад":
        AdminState.editing_schedule = True
        await message.reply("📝 Введіть новий розклад:")

async def process_admin_message(message: types.Message):
    user_id = message.from_user.id
    if is_admin(user_id):
        if AdminState.editing_schedule:
            update_schedule(message.text.strip())
            AdminState.editing_schedule = False
            await message.reply("✅ Розклад оновлено.")
        elif message.text == "📋 Переглянути всіх користувачів":
            await show_user_list(message)
        elif message.text == "➖ Змінити розклад":
            await handle_admin_command(message)
        else:
            await message.reply("⚠️ Невідома команда адміністратора.")
    else:
        # Якщо користувач не адмін, зберігаємо його контакт
        if user_id in users:
            if not users[user_id].get("contact"):
                users[user_id]["contact"] = f"@{message.from_user.username}" if
```

```
save_users()



SCHEDULE_FILE = "schedule.txt"

def get_schedule():
    try:
        with open(SCHEDULE_FILE, "r", encoding="utf-8") as f:
            return f.read()
    except FileNotFoundError:
        return "Розклад ще не створено."

def update_schedule(new_schedule):
    with open(SCHEDULE_FILE, "w", encoding="utf-8") as f:
        f.write(new_schedule)

from aiogram import types
from file_storage import save_data
from schedule_utils import get_schedule

group_kb = types.ReplyKeyboardMarkup(resize_keyboard=True)
group_kb.add("🎨 Малювання", "🎶 Музика", "🤖 Робототехніка")
group_kb.add("📅 Переглянути розклад", "📝 Зареєструватися")
group_kb.add("❓ Допомога", "📞 Зв'язатися з адміністратором")
group_kb.add("🔐 Адмін-панель")

async def handle_start(message: types.Message, users):
    user_id = message.from_user.id
    users[user_id] = {
```

```
        "name": message.from_user.full_name,
"group": None,
        "contact": f"@{message.from_user.username}" if message.from_user.username
else "Без юзернейму"
    }
    save_data(users)
    await message.reply("👋 Вітаємо! Оберіть гурток:", reply_markup=group_kb)

async def handle_group_choice(message: types.Message, users):
    user_id = message.from_user.id
    text = message.text.strip()

    if user_id not in users:
        users[user_id] = {
            "name": message.from_user.full_name,
            "group": None,
            "contact": f"@{message.from_user.username}"
 if message.from_user.username else "Без юзернейму"
        }

    if text in ["🎨 Малювання", "🎶 Музика", "🤖 Робототехніка"]:
        users[user_id]["group"] = text
        save_data(users)
        await message.reply(f"✅ Ви обрали гурток: {text}")

    elif text == "📅 Переглянути розклад":
        schedule = get_schedule()
        await message.reply(f"🗓 Розклад:\n\n{schedule}")

    elif text == "📝 Зареєструватися":
```

```
        group = users[user_id].get("group")
if group:
            await message.reply(f"✅ Ви зареєстровані на гурток {group}. Дякуємо!")
        else:
            await message.reply("⚠️ Спочатку оберіть гурток перед реєстрацією.")

    elif text == "❓ Допомога":
        await message.reply("ℹ️ Для реєстрації оберіть гурток, а потім натисніть «📝 Зареєструватися». Якщо виникли питання — звертайтесь до адміністратора.")

    elif text == "📞 Зв'язатися з адміністратором":
        await message.reply("📩 Напишіть адміну: @water_in_your_brain")

    elif text == "🔐 Адмін-панель":
        await message.reply("Щоб увійти до адмін-панелі, введіть пароль у форматі: `Пароль: ваш_пароль`", parse_mode="Markdown")

    else:
        await message.reply("❓ Будь ласка, оберіть опцію з клавіатури.")
```

ДОДАТОК В. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ



4

Мета і завдання

Мета:
Розробка Telegram-бота для зручного менеджменту освітніх послуг.

Завдання:

- Провести аналіз існуючих рішень.
- Визначити вимоги до бота.
- Реалізувати основний функціонал.
- Протестувати систему.

5

Об'єкт і предмет дослідження

Об'єкт:
Процес цифрової взаємодії у позашкільній освіті.

Предмет:
Інформаційна система в Telegram для реєстрації та адміністрування гуртків.

8

Технології

- Python 3.11
- Бібліотека aiogram (асинхронний фреймворк для ботів)
- JSON для збереження користувачів
- TXT-файл для розкладу
- Telegram Bot API
- Visual Studio Code

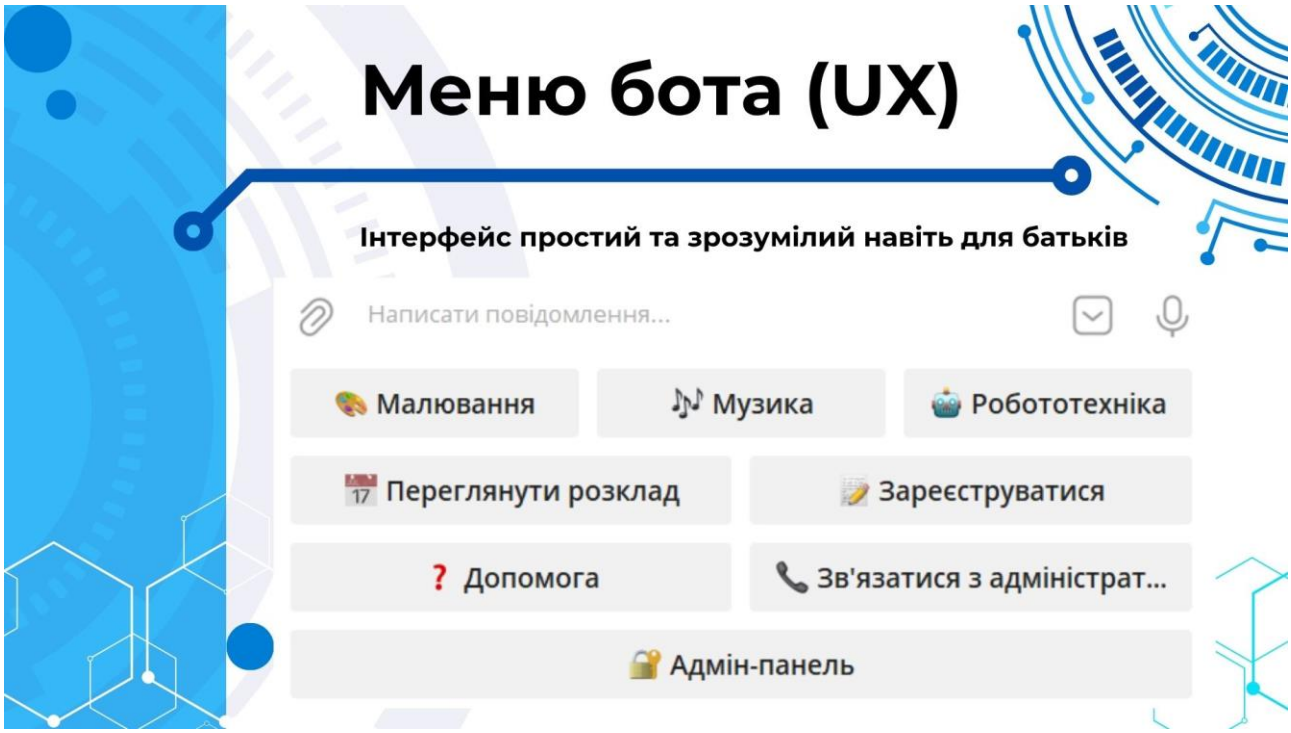
3

Структура бота

- users.json — база користувачів
- schedule.txt — розклад занять
- bot.py — запуск і логіка команд
- admin_panel.py — адмін-функціонал
- handlers.py — взаємодія з користувачем

Меню бота (UX)

Інтерфейс простий та зрозумілий навіть для батьків



Архітектура логіки Telegram-бота

🔹 Telegram-бот розроблено за допомогою Python та бібліотеки aiogram, яка дозволяє ефективно обробляти вхідні повідомлення в асинхронному режимі.

🔹 Основні логічні блоки:

- 📥 Командні обробники:
  - /start — запуск бота та меню.
  - /register — збереження користувача в JSON.
  - /schedule — читання розкладу з файлу.

Архітектура логіки Telegram-бота

🔄 Робота з текстовими повідомленнями:

- при надсиланні назви гуртка бот перевіряє, чи існує такий гурток у schedule.txt, і записує користувача в users.json.
- 🔹 Асинхронна архітектура забезпечує:
- високу швидкодію;
- одночасне обслуговування декількох користувачів;
- стабільність без використання зовнішньої БД.
- 🔍 Особливість: бот зберігає всі дані у вигляді текстових і структурованих файлів, що спрощує його розгортання без потреби в базі даних.

Модель збереження даних

📁 Telegram-бот не використовує традиційну базу даних, натомість обирається модель збереження у файловій системі:

1. 📄 users.json — основний файл, у якому зберігаються:
   - user_id — Telegram ID користувача;
   - full_name — ім'я;
   - selected_course — гурток, на який записаний;

2. 📄 schedule.txt — файл із розкладом:
   - кожен рядок містить назву гуртка, день та час;

Модель збереження даних

Переваги файлової моделі:

- простота реалізації;
- можливість ручного редагування даних;
- не потребує встановлення БД.

🔴 Недоліки:

- не придатна для великих обсягів даних;
- відсутність транзакційності;
- складніше реалізувати зв'язки між даними.

🧩 Така модель підходить для MVP-версії (мінімально життєздатного продукту), проте може бути замінена на БД у наступних ітераціях проєкту.

Результати реалізації

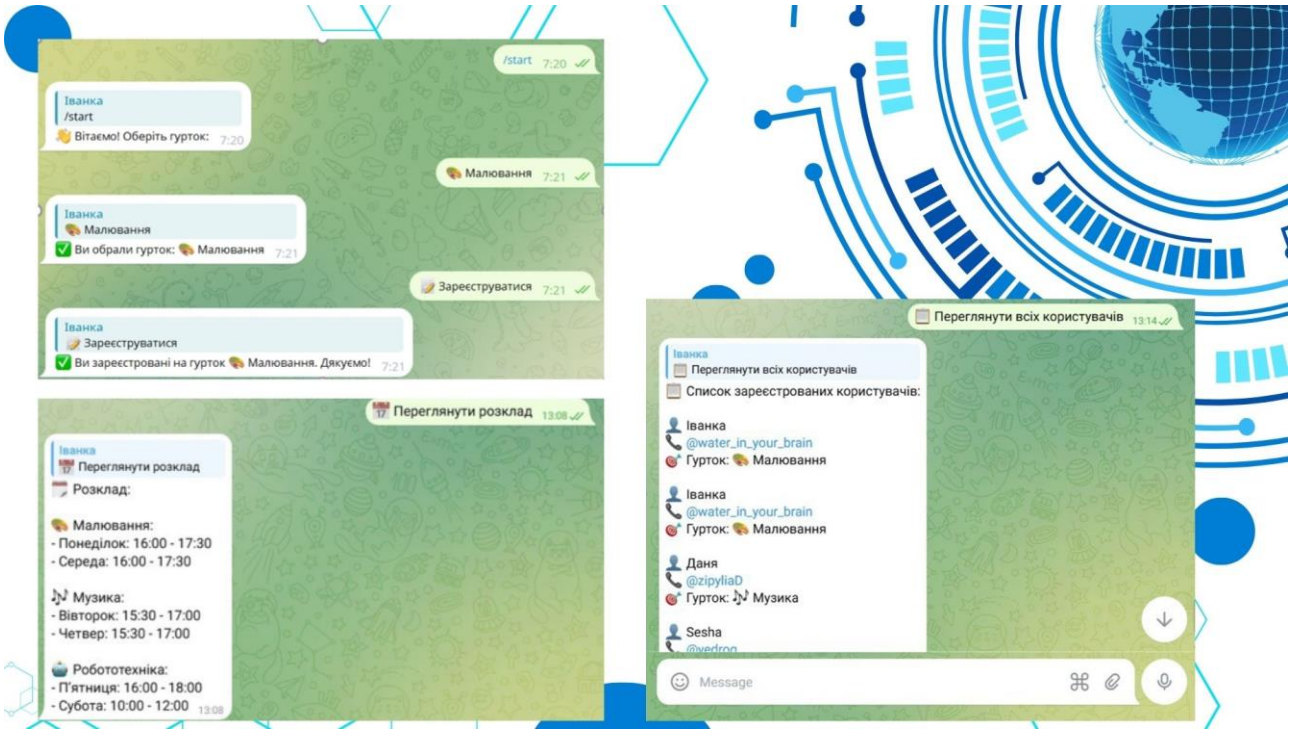
✅ Telegram-бот повністю розроблений та протестований.

✅ Реалізовано:

- реєстрацію користувача;
- запис на гурток за назвою;
- виведення розкладу;
- обробку повідомлень в асинхронному режимі.

✅ Усі дані зберігаються в текстових файлах:

- users.json — список учасників;
- schedule.txt — розклад гуртків.



Висновки

📌 Створено функціональний Telegram-бот для освітнього середовища.
📌 Досягнуто цілей: автоматизація, зручність, доступність.
📌 Проєкт легко адаптується для шкіл, гуртків та позашкільних установ.
📌 Можливе подальше вдосконалення з мінімальними затратами.