МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ім. Ю.М. Потебні
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

КАФЕДРА ЕЛЕКТРОНІКИ, ІНФОРМАЦІЙНИХ СИСТЕМ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Кваліфікаційна робота

перший (бакалаврський)
(рівень вищої освіти)

на тему **Розробка Telegram-бота для організації власної активності користувача**

Виконав: студент 4 курсу, групи 6.1212-пзс-з
спеціальності 121 Інженерія програмного забезпечення
(код і назва спеціальності)
освітньої програми Програмне забезпечення систем
(код і назва освітньої програми)
Д. Є. Басс
(ініціали та прізвище)
Керівник доцент, к.ф.-м.н. І.А.Скрипник
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)
Рецензент директор ТОВ «Дісітел»
П.О. Лютий
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)

Запоріжжя
2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ім. Ю.М. Потебні
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

Кафедра електроніки, інформаційних систем та програмного забезпечення

Рівень вищої освіти _____перший (бакалаврський)_____

Спеціальність ___121_Інженерія програмного забезпечення_____
(код та назва)

Освітня програма ___Програмне забезпечення систем____
(код та назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри_____Тетяна КРИТСЬКА
“_01_” __березня_____2026_року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Басс Дарині Євгенівні
(прізвище, ім’я, по батькові)

1. Тема роботи Розробка Telegram-бота для організації власної активності користувача

керівник роботи Скрипник Ірина Анатоліївна, к. ф.-м. н., доцент
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від 23.12.2025 р. № 1830-с

2. Строк подання студентом кваліфікаційної роботи 18.05.2026 р.

3. Вихідні дані бакалаврської роботи
   • комплект нормативних документів;
   • технічне завдання до роботи.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
   • огляд та збір літератури стосовно теми кваліфікаційної роботи;
   • огляд та аналіз існуючих рішень та аналогів;
   • дослідження проблеми створення мобільних застосунків;
   • створення програмного продукту та його опис;
   • перелік вимог для роботи програми;
   • тестування програмної системи та розробка висновків та пропозицій.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
13 слайдів презентації

6. Консультанти розділів бакалаврської роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата Завдання прийняв

7. Дата видачі завдання 01.03.2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Аналіз предметної області	02.03-15.03.26	виконано
2	Формулювання основної задачі дипломної роботи та узгодження її з науковим керівником	16.03.26	виконано
3	Аналіз існуючих методів рішення	17.03-22.03.26	виконано
4	Аналіз сучасних засобів розробки Telegram-ботів	23.03-29.03.26	виконано
5	Узгодження подальших дій з науковим керівником	30.03.26	виконано
6	Формування вимог до оточення	30.03-01.04.26	виконано
7	Проєктування Telegram-бота	02.04-20.04.26	виконано
8	Програмна реалізація застосунку	20.04-03.05.26	виконано
9	Тестування застосунку	04.05-10.05.26	виконано
10	Представлення отриманих результатів науковому керівнику	11.05.26	виконано
11	Оформлення звіту	11.05-17.05.26	виконано
12	Оформлення презентації.	.05-18.05.26	виконано

Студент _____ Басс Д.Є.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Скрипник І.А.
(підпис) (прізвище та ініціали)

Нормоконтроль пройдено
Нормоконтролер _____ Скрипник І.А.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Сторінок – 77.

Рисунків – 31.

Джерел – 17.

Басс Д.Є. Розробка Telegram-бота для організації власної активності користувача: кваліфікаційна робота бакалавра спеціальності 121 «Інженерія програмного забезпечення»/ наук. керівник І. А. Скрипник. Запоріжжя : ЗНУ, 2026. 77 с.

Метою дипломного проєкту є створення доступного, надійного та зручного Telegram-бота, який дозволить користувачам організовувати, систематизувати та зберігати власні задачі у вигляді списку нотаток, а також нагадувань про свої завдання та дедлайни.

У результаті дипломної роботи було розроблено Telegram-бот, який може застосовуватися для особистого планування, збереження нотаток, контролю нагадувань і відстеження виконаних завдань. Поставленої мети було досягнуто за допомогою аналізу предметної області, дослідження сучасних методів розробки, формування вимог до оточення та до системи. Важливими етапами стали проєктування застосунку за допомогою UML-діаграм і формування функціональних вимог.

Дослідження предметної області проєкту було представлено та опубліковано у збірнику Міжнародної наукової інтернет-конференції на тему "Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення" (випуск 109) [1].

Ключові слова: *Telegram-бот, планування, організація, нотатки, нагадування, Telegram Bot API, Python, aiogram, SQLite, aiosqlite.*

ABSTRACT

Pages – 77.

Drawings – 31.

Sources – 17.

Bass D. Development of a Telegram Bot for Organizing the User's Personal Activity: bachelor's qualification work in specialty 121 "Software Engineering" / scientific supervisor I. A. Skrypnyk. Zaporizhzhia : ZNU, 2026. 77 p.

The aim of the diploma project is to create an accessible, reliable and convenient Telegram bot that will allow users to organize, systematize and store their personal tasks in the form of a list of notes, as well as reminders about their tasks and deadlines.

As a result of the diploma work, a Telegram bot was developed that can be used for personal planning, storing notes, managing reminders and tracking completed tasks. The set aim was achieved through the analysis of the subject area, the study of modern development methods, and the formation of requirements for the environment and the system. Important stages included designing the application using UML diagrams and forming functional requirements.

The study of the project subject area was presented and published in the collection of the International Scientific Internet Conference on the topic "Information Society: Technological, Economic and Technical Aspects of Formation" (Issue 109) [1].

Keywords: *Telegram bot, planning, organization, notes, reminders, Telegram Bot API, Python, aiogram, SQLite, aiosqlite.*

ЗМІСТ

ВСТУП 7
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 12
1.1 Історія Telegram. Визначення Telegram-бота 12
1.2 Опис проблеми 14
1.3 Огляд існуючих методів рішення 16
1.4 Постановка задачі 21
2 ДОСЛІДЖЕННЯ ПРОГРАМНИХ ЗАСОБІВ РЕАЛІЗАЦІЇ TELEGRAM-БОТА 25
2.1 Аналіз сучасних мов програмування для розробки Telegram-бота 25
2.2 Аналіз сучасних систем керування базами даних Telegram-бота 28
2.3 Огляд інших технологій для розробки Telegram-бота 31
3 РОЗРОБКА TELEGRAM-БОТА 34
3.1 Опис предметної області 34
3.2 Вимоги до оточення 36
3.2.1 Вимоги до апаратного забезпечення 36
3.2.2 Вимоги до програмного забезпечення 38
3.2.3 Вимоги до користувачів 39
3.3 Архітектура системи 42
3.4 Специфікації даних 45
3.5 Функціональні та нефункціональні вимоги 47
3.5.1 Функціональні вимоги 47
3.5.2 Нефункціональні вимоги 51
3.6 Засоби реалізації 52
3.7 Модулі та алгоритми 54
3.8 Проєкт інтерфейсу 63
3.9 Реалізація та тестування 72
ВИСНОВКИ 75
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 76

ВСТУП

Актуальність теми

В умовах інтенсивного цифрового розвитку особливої актуальності набуває ефективне управління особистими завданнями та інформаційними потоками. Підвищене інформаційне навантаження, а також поєднання навчальної, професійної та особистої діяльності зумовлює потребу у зручних, інтегрованих та доступних інструментах планування.

Невід'ємною частиною життя багатьох людей є соціальні мережі та месенджери, які сьогодні є одним з основних механізмів зв'язку та комунікації в суспільстві. Велика кількість платформ забезпечує різноманіття вибору для користувачів, згідно з їхніми інтересами, віком та цифровими можливостями. Telegram є одним із найпопулярніших месенджерів серед української молоді, незважаючи на відсутність значної популярності у вікових групах старше 35 років [2]. Тому кращим варіантом для створення зручного, доступного та інтегрованого в різні системи застосунку для організації власної активності користувача буде бот на основі Telegram. Цей месенджер також дає низку переваг при розробці, наприклад, готове середовище для взаємодії з користувачем, зручний інтерфейс та наявність Telegram Bot API.

Огляд значної кількості існуючих програм-аналогів демонструє значну кількість проблем, пов'язаних з примітивністю застосунків, труднощами систематизації та орієнтацією на одну платформу. Оскільки месенджер Telegram вже доступний для всіх сучасних операційних систем (Android, iOS, Windows, Mac, Linux, macOS, також існує веб-версія) [3], використання саме цієї платформи надасть низку переваг та задовольнить частину потреб. Таким чином, створення застосунку для організації та систематизації власної активності користувача на основі Telegram є актуальною темою для розробки програмного забезпечення.

Мета дослідження

Метою дипломного проєкту є створення доступного, надійного та зручного Telegram-бота, який дозволить користувачам організовувати, систематизувати та зберігати власні задачі у вигляді списку нотаток, а також користувачі зможуть отримувати нагадування про свої завдання та дедлайни.

Завдання дослідження

Проаналізувати існуючі рішення у сфері додатків для нагадувань, приділити особливу увагу Telegram-ботам у цій галузі. Ознайомитися з їхніми функціями, можливостями, інтерфейсом та особливостями. Дослідити існуючі програмні засоби реалізації, які використовуються для розробки Telegram-ботів, такі як відповідні мови програмування та сучасні системи керування базами даних. Сформувати вимоги до оточення, а саме до апаратного забезпечення, до програмного забезпечення, а також до користувачів. Обґрунтувати вибір оптимальних засобів реалізації для розробки Telegram-бота, беручи до уваги функціональні та нефункціональні вимоги до програмного забезпечення. Спроєктувати та розробити Telegram-бота, який буде мати функціонал створення та управління нотатками та нагадуваннями.

Об’єкт дослідження

Об’єктом дослідження є процес розробки бота для організації власної активності користувача на платформі месенджера Telegram.

Предмет дослідження

Предметом дослідження є технології та методи створення власного програмного забезпечення у вигляді Telegram-бота для задоволення планувальних потреб користувача з метою підвищення власної продуктивності та зниження тривожності.

Методи дослідження

Теоретичні: проведення детального аналізу літературних джерел, наукових публікацій, статей, підручників та науково-технічної літератури, що стосуються технологій розробки Telegram-бота. Порівняння бібліотек для роботи з Telegram, таких як aiogram, python-telegram-bot, Telebot та pyTelegramBotAPI. Дослідження та зіставлення існуючих рішень.

Практичні: проєктування, програмування, тестування програмного коду.

Практичне значення одержаних результатів

Практичне значення одержаних результатів полягає у створенні готового до використання Telegram-бота, який надає користувачам зручний інструмент для планування, організації та систематизації власної активності. Розроблений програмний засіб може використовуватися активними користувачами Telegram, які поєднують у цьому месенджері різні сфери життя, а саме особисте, робоче, навчальне тощо.

Розроблюване програмне забезпечення може стати основою для подальшого комерційного бізнес-проєкту, за допомогою розширення функціоналу та монетизації.

Апробація результатів

Ідею проєкту було представлено та опубліковано у збірнику Міжнародної наукової інтернет-конференції на тему "Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення" (випуск 109).

Результати роботи було представлено на університетській науково-практичній конференції студентів, аспірантів, докторантів і молодих вчених «Молода наука-2026»

Глосарій

Telegram – популярний безкоштовний хмарний месенджер, що надає можливості спілкування та обміну повідомленнями між користувачами.

Бот – спеціальна комп'ютерна програма, яка автоматично виконує заздалегідь визначені дії або сценарії в інтернеті, імітуючи людську поведінку.

Telegram Bot API – офіційний інтерфейс на основі HTTPS, який дозволяє розробникам створювати програми-боти для месенджера Telegram.

Telegram-бот – автоматизована програма, що працює всередині месенджера, виконуючи завдання за сценарієм або запитом користувача (продажі, інформування, підтримка) без участі людини.

Месенджер – програма або вебсервіс для миттєвого обміну текстовими, голосовими повідомленнями, фото, відео та файлами через інтернет у реальному часі.

Соціальні мережі – онлайн-платформи та веб-сайти, призначені для побудови соціальних взаємовідносин, спілкування, обміну інформацією та контентом між користувачами.

Інформаційне навантаження – обсяг інформації, який користувач повинен обробляти протягом певного часу, що може впливати на ефективність діяльності.

Планування – процес визначення завдань, встановлення пріоритетів і розподілу часу для досягнення поставлених цілей.

Особиста продуктивність – здатність користувача ефективно виконувати завдання з мінімальними витратами часу та ресурсів.

Нотатки (замітки) – структуровані текстові записи, які використовуються для збереження інформації, ідей або завдань.

Нагадування – функція програмного забезпечення, що дозволяє повідомляти користувача про заплановані події або завдання у визначений час.

Дедлайн – кінцевий термін виконання завдання або досягнення певної мети.

Програмне забезпечення (ПЗ) – сукупність програм, процедур і даних, що забезпечують функціонування комп’ютерної системи.

Кросплатформність – здатність програмного забезпечення працювати на різних операційних системах без значних змін.

API (Application Programming Interface) – набір правил і засобів, що дозволяють різним програмам взаємодіяти між собою.

База даних (БД) – організована структура для зберігання, обробки та управління даними.

Система керування базами даних (СКБД) – програмне забезпечення для створення, ведення та використання баз даних.

Функціональні вимоги – вимоги до системи, що описують її основні функції та можливості.

Нефункціональні вимоги – вимоги до якості роботи системи (швидкодія, надійність, безпека тощо).

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Історія Telegram. Визначення Telegram-бота

Telegram, як платформа для спілкування, почав свою історію у 2013 році та спочатку планувався експериментальним проєктом Павла Дурова, де тестувалася технологія зашифрованого листування – MTProto, автором якої був його брат Микола Дуров. Майже весь час існування месенджера засновник зазнавав тиску з боку деяких держав та неодноразово зіткнувся з примусом надавати ключі шифрування. Проте Павло зайняв принципову позицію не передавати доступ державним органам до шифрування свого месенджера, оскільки в його ідеології свобода – понад усе [4]. Технологія MTProto виявилася настільки надійною, що засновник Telegram вільно публікував шматки зашифрованого тексту з особистого чату, проте спеціалістам не вдалося розшифрувати написане [5]. У результаті конфліктів навколо платформи, її популярність зросла настільки, що у 2014 році кількість активних користувачів сягнула мільйона, через що команда розробників була вимушена розширяти функціонал та додавати нові можливості для користувачів. Одним із таких удосконалень стало заснування Telegram-ботів. У червні 2015 року розробники месенджера представили Bot API – інтерфейс програмування, який дозволив стороннім розробникам створювати власних ботів [6]. Цей етап розвитку став надзвичайно важливим, оскільки значно розширив функціональні можливості платформи. Від того часу Telegram-боти почали активно впроваджуватися у сфери бізнесу, освіти, розваг, електронної комерції та особистої продуктивності.

Основною метою Telegram-ботів є автоматизація рутинних процесів, підтримка клієнтів, інформування, поширення контенту, маркетинг і продажі, освіта та навчання, а також розваги. Завдяки своїй інтеграції у месенджер, боти не потребують встановлення додаткового програмного забезпечення, що

робить їх зручними та доступними в будь-який час для широкого кола користувачів.

Станом на 2026 рік майже кожна велика компанія має свого Telegram-бота для цілодобової підтримки клієнтів, це не дивно, оскільки функціональні можливості стали досить широкими. Як правило, за допомогою такого програмного забезпечення клієнт може отримати миттєві відповіді на типові запитання, ознайомитись з каталогом товарів, оформити замовлення та оплатити його, забронювати столик в ресторані, отримати розсилку новин тощо. Тому функціонал Telegram-бота майже необмежений, основою взаємодії є обробка команд, збереження інформації про користувача та надсилання повідомлень у відповідь, також можлива інтеграція з іншими сервісами та базами даних.

Офіційної загальноприйнятої класифікації для Telegram-ботів не існує, оскільки функціонал обмежується фантазією розробника, автори різних статей виділяють різні типи та дають їм власні назви. Проте, проаналізувавши декілька джерел інформації, залежно від призначення, Telegram-боти можна класифікувати таким чином:

- Чат-бот (веде діалог, відповідає на часті запитання).
- Інформаційний бот (надсилає новини, погоду).
- Бот-менеджер (видалення спаму в каналах, блокування користувачів).
- Бот-помічник (планер, збереження файлів).
- Аналітичний бот (оцінка аудиторії в каналі).
- Інтерактивний бот (ігри, опитування).

Таким чином, можна стверджувати, що Telegram-бот – це програмний застосунок, що функціонує в межах месенджера Telegram та взаємодіє з користувачем шляхом обміну повідомленнями. Бот здатен автоматично обробляти запити користувача, виконувати задані команди та надавати необхідну інформацію без участі людини-оператора.

У межах даної дипломної роботи розглядається бот-помічник для організації власної активності користувача. «Помічники» такого типу

спрямовані на допомогу користувачу у плануванні повсякденних справ та сприяють підвищенню особистої продуктивності, зменшенню інформаційного перевантаження та зниженню рівня стресу і тривожності, пов’язаних з дедлайнами.

1.2 Опис проблеми

XXI століття вважається епохою інформаційних технологій (ІТ), оскільки саме у цей період відбувається стрімкий розвиток цифрових рішень, обчислювальних систем та глобальних мереж передачі даних. Застосування ІТ охоплює всі сфери життя, зокрема освіту, медицину, бізнес, державне управління, а також повсякденне життя. Така універсальність та масштаб впливу ІТ визначають їх провідну роль у формуванні сучасного інформаційного суспільства. Водночас стрімкий розвиток ІТ та впровадження у повсякденне життя може свідчити про постійні безконтрольні потоки даних, що значно підвищують інформаційне навантаження на людину порівняно з попередніми періодами [7]. Сучасні користувачі отримують інформацію з такою інтенсивністю, яка виходить за межі когнітивних можливостей людини [8], тобто користувач не може ефективно обробляти всі дані, отримані з соціальних мереж та онлайн-платформ, що призводить до загострення проблем із увагою та концентрацією, погіршення пам’яті, підвищення рівня стресу, тривожності та зниження продуктивності. Вищезазначені пункти негативно впливають на психічне здоров’я користувача, а це може призводити до загострення певних психічних розладів, зокрема тривожних розладів, порушень сну, синдрому дефіциту уваги та гіперактивності (СДУГ) та емоційного вигорання.

СДУГ – це нейробіологічний розлад розвитку, що характеризується стійкою неуважністю, гіперактивністю та імпульсивністю, а також є невиліковним та в більшості випадків зберігається у дорослому віці, впливаючи на навчання, роботу та самоконтроль. Як свідчать дані наукових

досліджень, проблема СДУГ має високу поширеність серед дітей та молоді [9] і залишається постійною та зростаючою проблемою громадського здоров'я [10]. Статистика свідчить, що у 2018 році частка діагностованих випадків СДУГ (9,8%) була майже удвічі більшою, ніж у 1997 році (5,5%) [11].

Як зазначалось раніше, навіть без діагностованих проблем із психічним здоров’ям, інформаційне перевантаження негативно впливає на життя активного користувача. Рішенням проблеми може стати ведення планерів і щоденників справ для освітніх, робочих та особистих сфер життя, що значно знизить навантаження на мозок людини. Проте, аналіз поведінки користувачів свідчить про низку типових проблем у сфері управління завданнями. У більшості спостерігається роздробленість збереження інформації, а саме нотатки розміщуються у різних середовищах (паперові блокноти, щоденники, календарі, месенджери, застосунки тощо), що ускладнює їх централізацію, систематизацію та контроль загалом. З цього випливає відсутність чіткої системи пріоритезації та структурування завдань, що призводить до хаотичного зберігання даних і втрати важливості інформації. Тільки частина з перерахованих вище середовищ зберігання завдань може надіслати нагадування про строки виконання та надати можливість зафіксувати факт виконання, який є важливою частиною системи винагороди мозку [12]. Невід’ємною частиною проблеми нагадувань є різноманіття гаджетів та операційних систем, які в більшості випадків несинхронізовані один з одним та не знаходяться в центрі уваги користувача впродовж всього часу. Таким чином, існує потреба у створенні єдиного, доступного інструменту, що забезпечуватиме користувача централізованим зберіганням, структуризацією, пріоритезацією та нагадуваннями про завдання в межах доступної та широко використовуваної платформи.

1.3 Огляд існуючих методів рішення

Станом на 2026 рік існує велика кількість інструментів для створення нотаток та нагадувань. Існуюче різноманіття закриває різні потреби, від простих застосунків для нагадувань, вбудованих в операційні системи, до повноцінних ігрових застосунків з винагородами та серіями виконаних завдань. У межах дослідження було проведено огляд існуючих рішень, різноманітних варіантів ведення планерів та створення нагадувань, найпопулярніші будуть представлені далі.

@SkeddyBot – Telegram-бот для створення нагадувань природною мовою, також має веб-версію застосунку. (див. Рис. 1)

Основні функціональні можливості: створення простих нагадувань у вільній формі, створення повторюваних нагадувань із певною циклічністю, редагування та видалення наявних нагадувань, створення нотатки [13].

Переваги:

- Бот працює у месенджері Telegram, тому нагадування можуть приходити на будь-який пристрій.
- Сприймає природну мову, тобто «Зателефонувати керівнику завтра о 14:00», тому дуже швидке створення нагадувань.
- Інтуїтивно зрозумілий інтерфейс та прості команди.
- Веб-застосунок, через який також відбувається керування, але нагадування все одно приходять у чат Telegram.
- Створення складних нагадувань з інтервалами або певними датами через веб-сайт.

Недоліки:

- Обмежений та невеликий функціонал.
- Українська мова наразі недоступна.
- Один список для всіх задач.

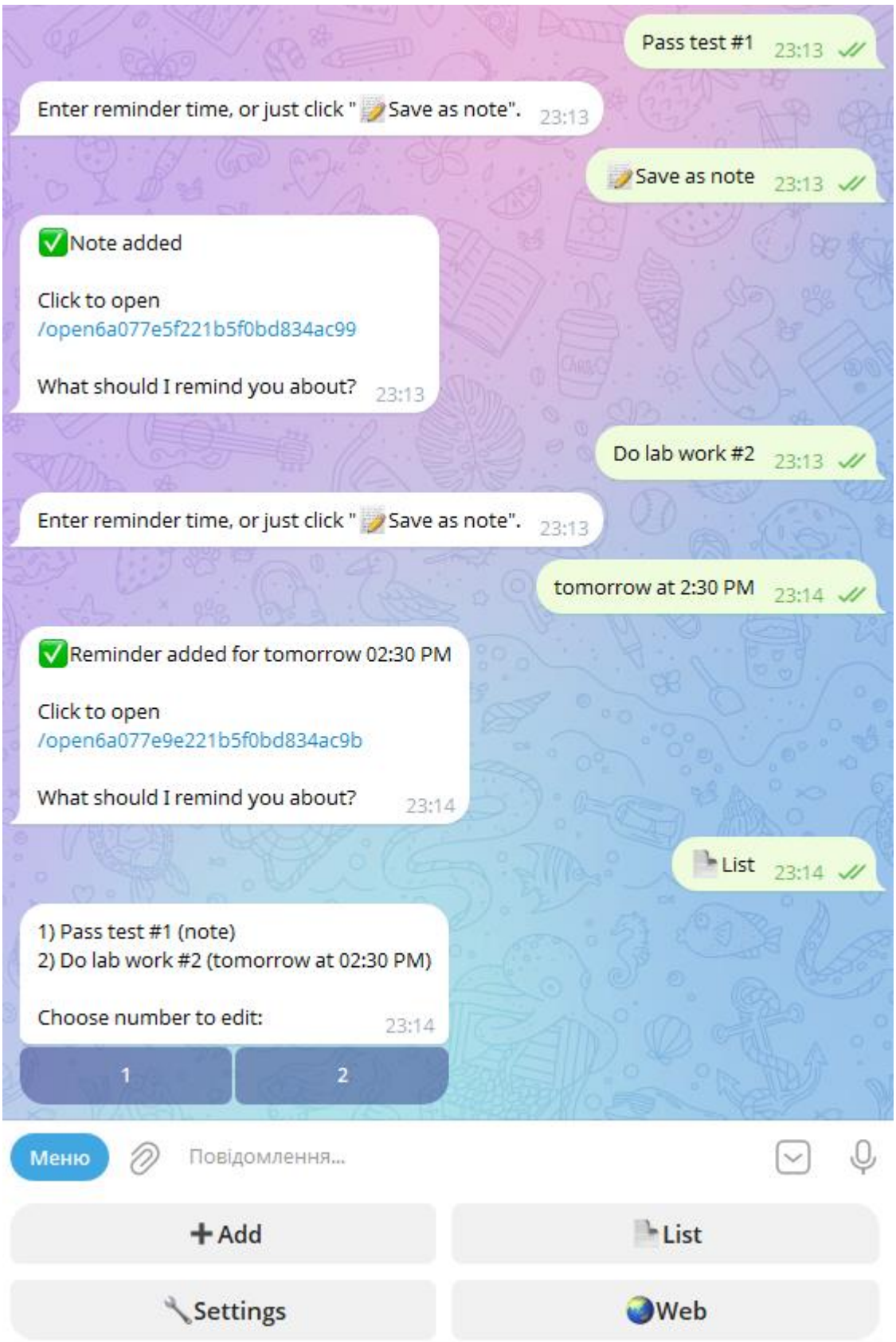


Рисунок 1 — *Інтерфейс Telegram-бота SkeddyBot.*

Нагадування – системний додаток для створення простих нагадувань, який підтримується продукцією Apple. (див. Рис. 2)

Основні функціональні можливості: створення нагадувань та управління ними. Можна додавати нагадування, писати нотатки до нього, встановлювати дату і час, додавати геолокацію як тригер, сортувати нагадування по списках (вбудованих або власноруч створених), обирати рівень пріоритетності для задачі та відмічати нагадування як виконане [14].

Переваги:

- Безкоштовна вбудована програма в операційну систему від компанії Apple.
- Простий, зрозумілий інтерфейс.
- Інтеграція з системою та продукцією Apple, тобто підтримується iOS, macOS, iPadOS, watchOS.
- Зручна організація сортування нагадувань за списками та пріоритетами.
- Гнучкі налаштування за часом та місцем.

Недоліки:

- Можливість використання програми тільки продукцією Apple, недоступна для інших операційних систем.
- Обмежена функціональність.
- Сповіщення схожі на системні, тому користувач не завжди звертає увагу.
- Деякі користувачі скаржаться на проблеми з синхронізацією між декількома пристроями.

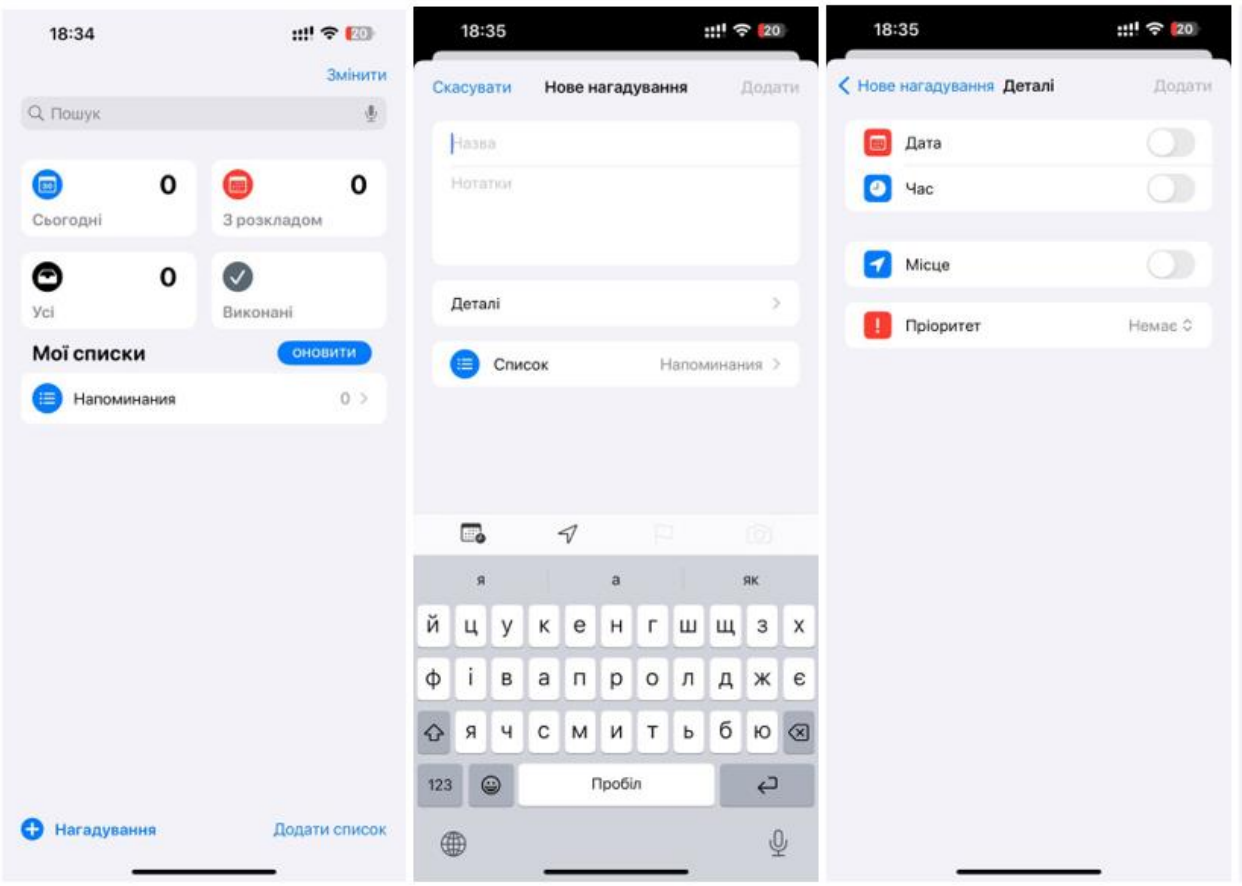


Рисунок 2 — *Інтерфейс додатку «Нагадування».*

Habitica – мобільний додаток для створення складних нагадувань, а саме звичок, щоденних завдань та завдань в ігровому стилі. (див. Рис. 3, 4)

Основні функціональні можливості:

Відстеження виконання завдань через гейміфікацію з винагородами, розподіл завдань на звички, щоденні справи та цілі, створення власного персонажа, відстеження прогресу, нагадування про справи користувача, створення команд для виконання спільних завдань [15] .

Переваги:

- Безкоштовні базові можливості.
- Кросплатформність, застосунок доступний на iOS, Android та в браузері.
- Можливе відстеження прогресу, статистика.
- Вбудоване розділення на три групи різних типів задач.
- Гейміфікація щоденних справ, підтримка мотивації.

Недоліки:

- Складний інтерфейс через велику кількість функцій.
- Неможливість створення власного групування.
- Нагадування про всі справи один раз на добу, а не повідомлення про кожну справу.
- Складність заробітку внутрішньої валюти без покупки підписки;
- Серія щоденних входів у застосунок, яка може приносити більше шкоди, аніж користі.
- Відсутність української мови.

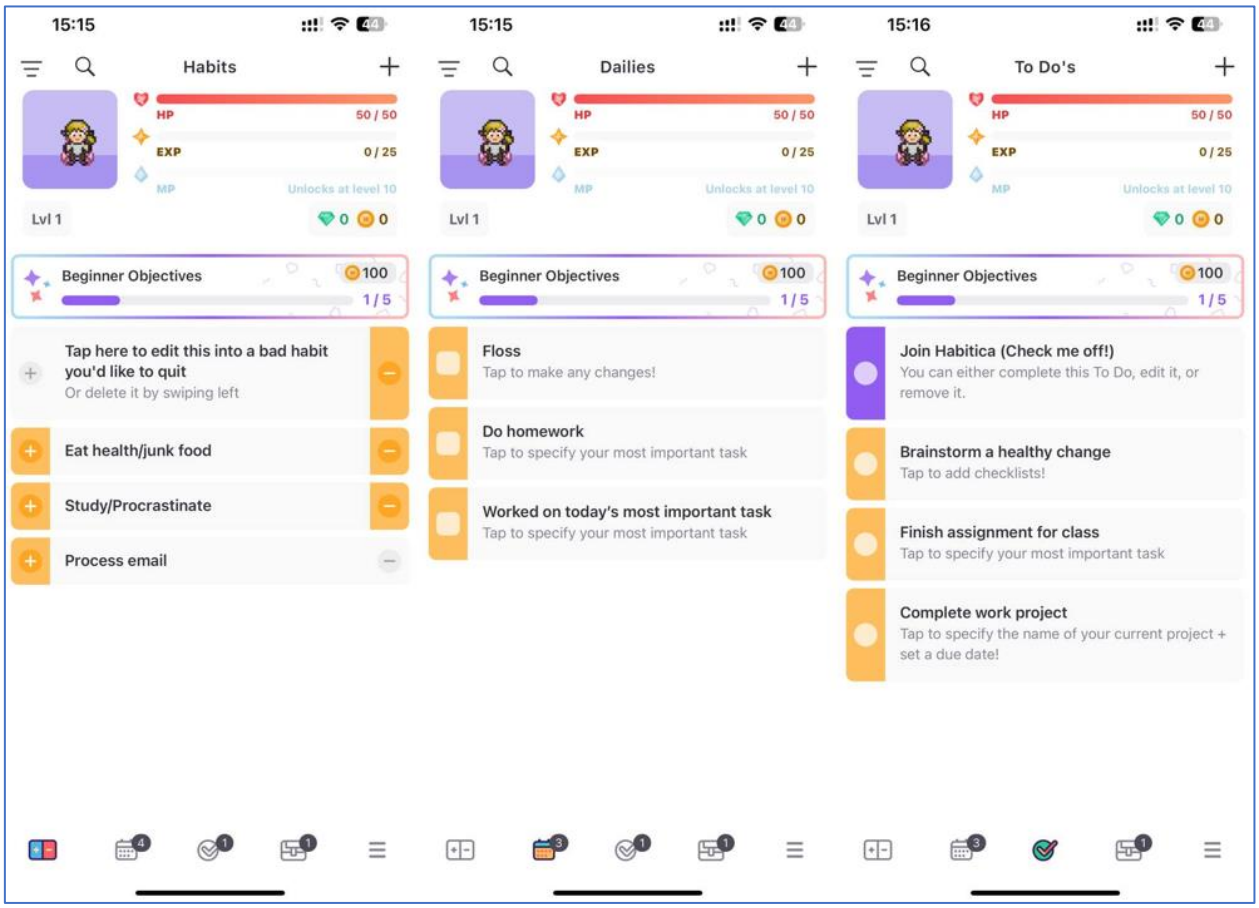


Рисунок 3 — *Інтерфейс додатку «Habitica».*

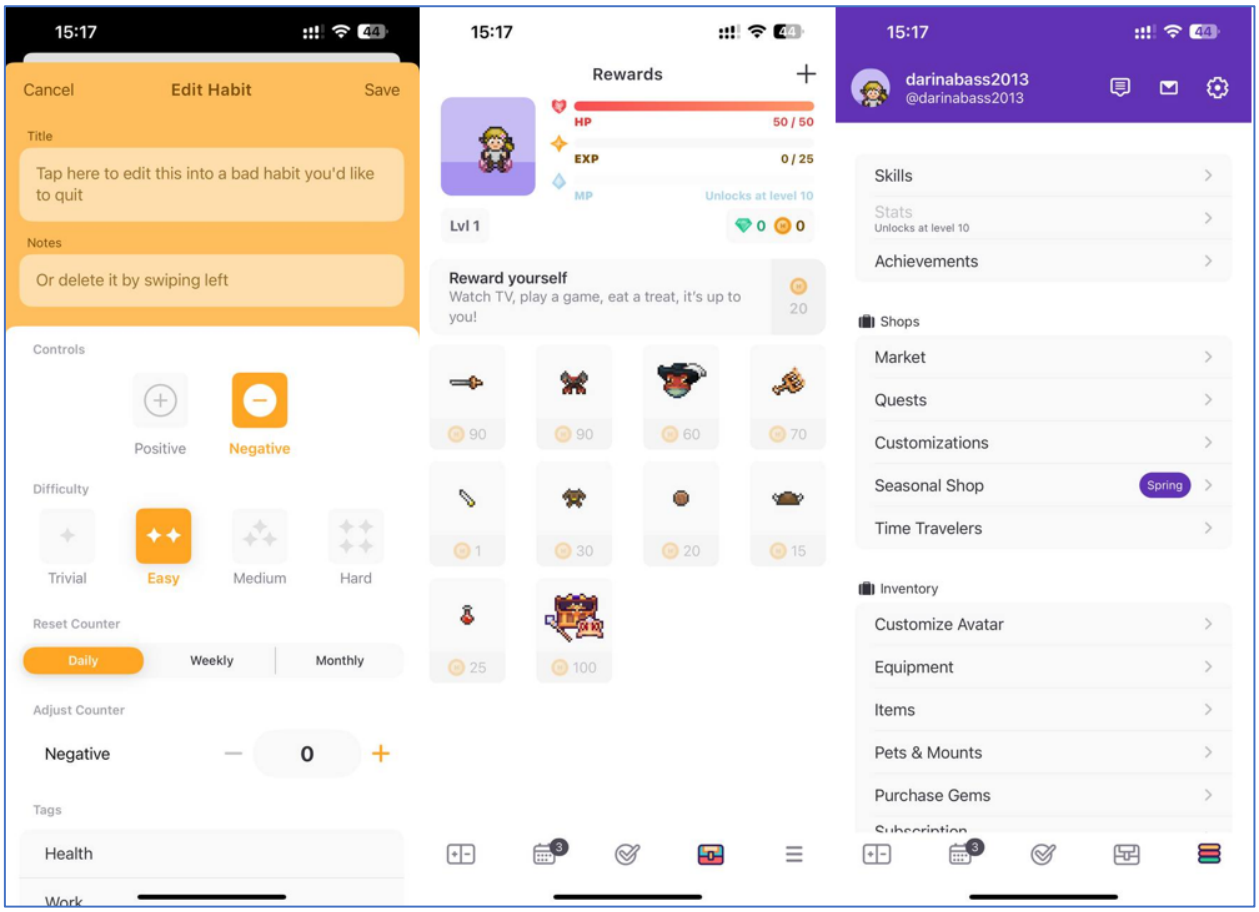


Рисунок 4 — *Інтерфейс додатку «Habitica».*

1.4 Постановка задачі

Мета дипломної роботи: створення Telegram-бота, який дозволить користувачам організовувати, систематизувати та зберігати власні задачі у вигляді списку нотаток, а також користувачі зможуть отримувати нагадування про свої завдання та дедлайни. Бот повинен надавати інтуїтивно зрозумілий інтерфейс для створення, редагування та видалення нагадувань та нотаток, дозволяти їхній перегляд, надавати можливість позначати завдання як виконані та виводити статистику про виконані завдання.

Для досягнення поставленої мети необхідно вирішити такі завдання:

1. **Провести аналіз літературних джерел** щодо месенджера Telegram, довести актуальність розробки бота саме для цієї соціальної мережі,

ознайомитися з історією Telegram-ботів, класифікувати їх та сформувати загальне визначення поняття Telegram-бота.

2. **Дослідити проблему**, яку вирішить розроблюваний застосунок, описати для яких користувачів розраховане таке програмне забезпечення та чим воно буде корисне, таким чином довести потрібність у реалізації.

3. **Провести огляд існуючих рішень** описаної проблеми методом аналізу програм-аналогів, описати основні можливості розглянутих застосунків, виявити переваги та недоліки для формування списку необхідних функцій розроблюваного Telegram-боту.

4. **Дослідити програмні засоби реалізації Telegram-бота**, а саме проаналізувати сучасні мови програмування для розробки та системи керування базами даних, а також ознайомитися з іншими можливими технологіями розробки. Розглянути особливості розробки Telegram-бота, такі як робота з Telegram Bot API, асинхронне програмування та управління станами.

5. **Сформувати вимоги до оточення**, такі як вимоги до апаратного забезпечення, які охоплюють рекомендовану апаратну конфігурацію для ефективної роботи описуваної системи, вимоги до програмного забезпечення, встановлення якого необхідне для функціонування описуваної системи, а також вимоги до користувачів, де можна описати портрет користувача.

6. **Сформувати функціональні та нефункціональні вимоги до Telegram-бота** на основі проведеного дослідження (див. Рис. 5). Вимоги повинні охоплювати:

- Створення, перегляд, редагування та видалення нотаток.
- Створення, перегляд, редагування та видалення нагадувань із прив’язкою до часу, також вони можуть бути циклічними.
- Можливість групування нагадувань за списком та пріоритетом.
- Демонстрація статистики виконаних завдань.
- Інтуїтивно зрозумілий інтерфейс та команди.
- Коректна доставка нагадувань та даних про користувача.

7. **Спроєктувати архітектуру програмного забезпечення**, обрати відповідний архітектурний шаблон та визначити основні модулі системи, а саме модуль обробників команд, модуль роботи з базою даних, модуль статистики.

8. **Розробити програмний код Telegram-бота** за допомогою мови програмування Python та системи керування базами даних SQLite, відповідно до сформованих вимог до застосунку та розробленої архітектури даних.

9. **Тестування програмного забезпечення** з метою перевірки його функціональності, ефективності та виправлення можливих проблем та недоліків.

10. **Оцінка результатів виконаної роботи** методом зіставлення висунутих вимог та перевірки їх виконання у готовій системі, визначити можливі напрями та способи масштабування проєкту.

Результатом роботи є Telegram-бот, реалізований засобами Telegram Bot API, для створення нагадувань, який надасть користувачам зручний інструмент для планування та систематизації власних справ, а також є основою для створення бізнес-проєкту, розширення, масштабування та монетизації.

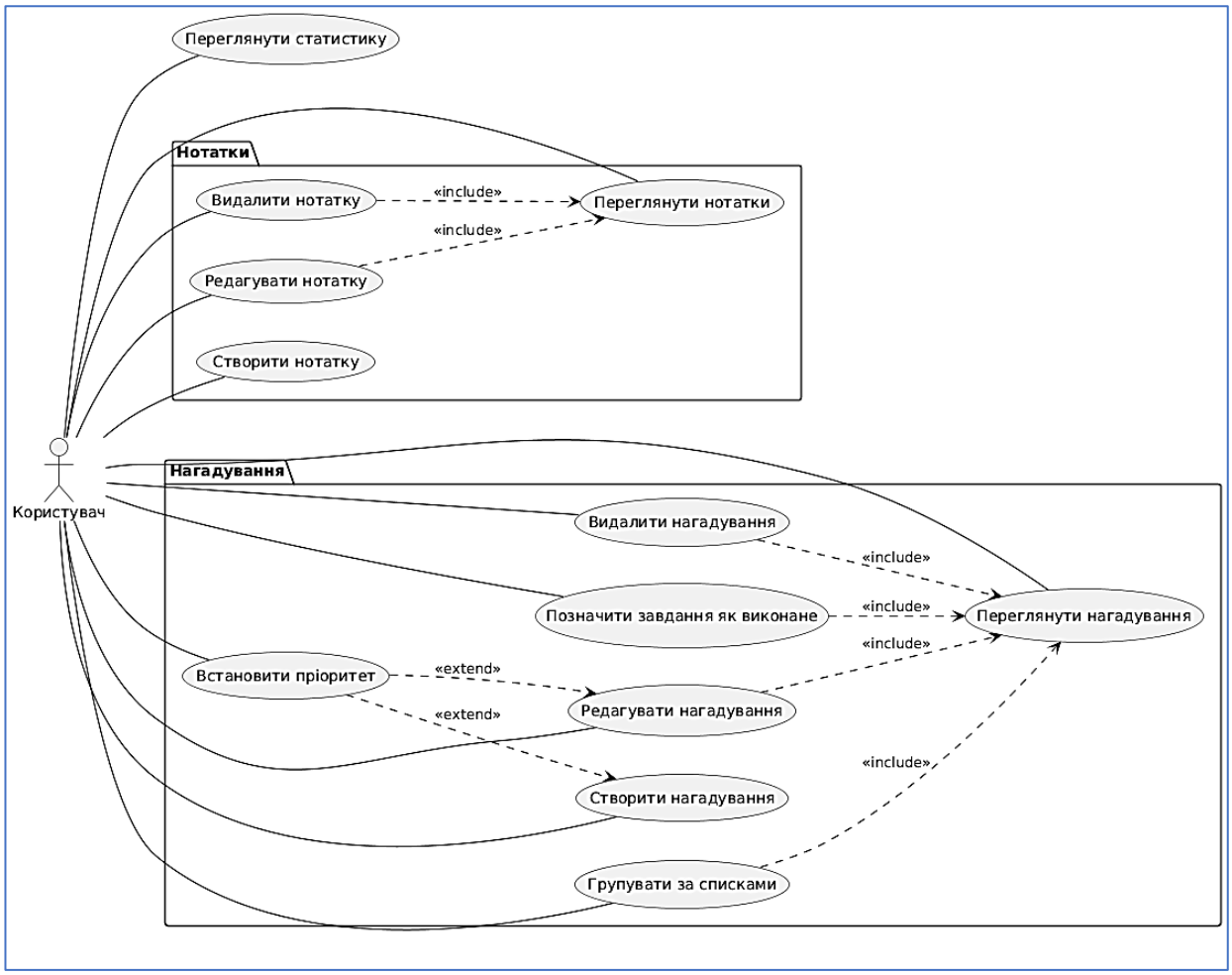


Рисунок 5 — *Стартова Use Case Diagram.*

2 ДОСЛІДЖЕННЯ ПРОГРАМНИХ ЗАСОБІВ РЕАЛІЗАЦІЇ TELEGRAM-БОТА

2.1 Аналіз сучасних мов програмування для розробки Telegram-бота

Розробка програмного сервісу у вигляді Telegram-бота для організації власної активності користувача потребує обґрунтованого вибору технологічного стеку, зокрема мови програмування та програмних бібліотек для взаємодії з API платформи Telegram. При виборі мови програмування необхідно враховувати ряд ключових факторів, що впливають на ефективність реалізації та подальшу підтримку застосунку. Насамперед важливим є наявність зручних та доступних бібліотек для роботи з API месенджера Telegram, що дозволяє значно спростити процес розробки. Також важливими є продуктивність мови, здатність обробляти велику кількість одночасних запитів користувачів, можливість інтеграції з базами даних та підтримка асинхронного програмування, що забезпечує ефективну роботу з паралельними операціями. Таким чином, беручи до уваги ці аспекти, можна обрати найоптимальнішу мову програмування, яка відповідатиме визначеним вимогам. Для роботи з Telegram API доступні бібліотеки різними мовами програмування, а саме Python, JavaScript, Java, PHP, Ruby, C#, Go, Swift, Kotlin та Rust [1, 16].

Python – одна з найпоширеніших мов програмування для створення Telegram-ботів.

Переваги:

- Простий і зрозумілий синтаксис, що дозволяє розробку навіть початківцям, але буде перевагою і для досвідчених програмістів.
- Велика кількість спеціалізованих бібліотек для роботи з Telegram API (aiogram, python-telegram-bot), які значно спрощують розробку.

- Підтримка асинхронного програмування (async/await), що дозволяє виконувати декілька завдань одночасно.
- Активна спільнота, велика кількість навчальних матеріалів та багато готових рішень, що спрощує пошук рішень.
- Легка інтеграція з базами даних, веб-сервісами та сторонніми API.
- Мінімальна кількість коду для реалізації базового функціоналу бота.

Недоліки:

- Нижча продуктивність у порівнянні з компільованими мовами (Java, C#, Go).
- Більше споживання оперативної пам’яті.
- Може вимагати використання додаткових інструментів для масштабування.
- Менш ефективний для створення високонавантажених систем без додаткової оптимізації.

JavaScript (Node.js) – популярне середовище для розробки серверних застосунків і ботів.

Переваги:

- Асинхронна модель виконання (event loop), що дозволяє ефективно обробляти велику кількість одночасних запитів.
- Висока швидкість обробки I/O-операцій (мережеві запити, робота з API).
- Велика екосистема бібліотек (наприклад, Telegraf, node-telegram-bot-api), які значно спрощують створення ботів для Telegram.
- Активна спільнота та постійний розвиток технологій.
- Простота інтеграції з веб-додатками та REST API.

Недоліки:

- Менш зручний для складних обчислень і обчислювально інтенсивних задач.

- Залежність від великої кількості сторонніх пакетів (npm), що може впливати на стабільність проєкту.
- Складність підтримки великих проєктів через асинхронність.
- Не завжди оптимальний вибір для великих корпоративних систем.

Java – об’єктно-орієнтована мова програмування, яка використовується для розробки серверних, корпоративних і кросплатформних застосунків.

Переваги:

- Висока продуктивність завдяки компіляції у байткод і оптимізації JVM.
- Сувора статична типізація, яка дозволяє виявляти помилки ще на етапі компіляції.
- Висока надійність і стабільність, що робить мову придатною для великих і довготривалих проєктів.
- Хороша масштабованість — підходить для систем з великою кількістю користувачів.
- Розвинена екосистема бібліотек і фреймворків (Spring).
- Зручна робота з багатопоточністю.
- Кросплатформність — “write once, run anywhere”.

Недоліки:

- Більш складний і громіздкий синтаксис у порівнянні з Python або JavaScript.
- Більший обсяг коду для реалізації простих задач.
- Повільніший старт розробки (більше налаштувань і шаблонного коду).
- Потребує більше ресурсів для запуску (через JVM).

PHP – часто використовується для веб-розробки, має просту інтеграцію з Telegram API, але поступається сучасним мовам у гнучкості.

Ruby – відома своєю простотою та продуктивністю, але менш популярна для Telegram-ботів.

C# – використовується в середовищі .NET, має хорошу підтримку для створення серверних застосунків.

Go – сучасна мова з високою продуктивністю та вбудованою підтримкою паралельності.

Swift – використовується переважно для iOS-розробки, рідко застосовується для ботів.

Kotlin – сучасна мова для JVM, альтернатива Java з більш зручним синтаксисом.

Rust – мова з високою продуктивністю та безпечним управлінням пам’яттю, але складніша у вивченні.

Огляд сучасних мов програмування для розробки Telegram-бота демонструє широкий вибір, кожна з яких має свої переваги та недоліки. Таким чином, враховуючи вимоги до простоти, найкращим вибором у цьому випадку буде мова програмування Python, оскільки вона найоптимальніша для початківців у цій сфері та підтримує велику кількість розвинених бібліотек для роботи з Telegram API.

2.2 Аналіз сучасних систем керування базами даних Telegram-бота

Розробка Telegram-бота потребує обґрунтованого вибору системи керування базами даних, оскільки саме вона забезпечує зберігання, обробку та швидкий доступ до інформації. При виборі СКБД необхідно звернути увагу на простоту інтеграції СКБД із обраною мовою програмування та тип СКБД, що визначає спосіб організації та обробки даних. Важливу роль відіграє продуктивність системи та швидкість обробки запитів, особливо в умовах одночасної роботи з кількома користувачами.

Враховуючи ці аспекти, можна обрати оптимальну систему керування базами даних, яка буде відповідати вимогам. Для створення Telegram-бота зазвичай обирають такі СКБД, як SQLite, PostgreSQL, MongoDB, MySQL та Redis, кожна з яких має свої особливості та сферу застосування.

SQLite

Переваги:

- Не потребує встановлення окремого сервера, оскільки є вбудованою реляційною СКБД [17] та часто використовується для різноманітних ботів.
- Простота використання та швидке підключення до застосунку.
- Мінімальні вимоги до ресурсів.
- Зручна для прототипування та навчальних цілей.
- Зберігає всі дані в одному файлі, що спрощує перенесення та резервне копіювання.
- Наявність бібліотеки aiosqlite для асинхронних ботів на Python.

Недоліки:

- Обмежена масштабованість при зростанні кількості користувачів, у разі виходу на комерційний рівень краще розглянути інші варіанти.
- Обмежена підтримка великої кількості одночасних записів.
- Не підходить для високонавантажених систем.

PostgreSQL

Переваги:

- Потужна реляційна СКБД з підтримкою складних запитів і аналітики.
- Добре масштабується та підходить для середніх і великих проєктів.
- Висока надійність і цілісність даних завдяки підтримці транзакцій (ACID).
- Підтримка розширень і додаткових можливостей (JSON, геодані тощо).
- Активна спільнота та регулярні оновлення.

Недоліки:

- Потребує окремого налаштування та адміністрування.
- Вищі вимоги до ресурсів у порівнянні з SQLite.
- Складніша у використанні.
- Може бути надлишковою у випадку Telegram-бота.

MongoDB

Переваги:

- Нереляційна (NoSQL) СКБД, що зберігає дані у форматі JSON-подібних документів.
- Гнучка структура даних (схема може змінюватися без складних міграцій).
- Висока швидкість роботи з великими обсягами даних.
- Добре підходить для швидко змінюваних або неструктурованих даних.
- Легко масштабується горизонтально.
- Зручна інтеграція з сучасними веб-технологіями.

Недоліки:

- Менш сувора цілісність даних у порівнянні з реляційними СКБД.
- Відсутність повноцінної підтримки складних транзакцій (у класичному вигляді).
- Може вимагати додаткової логіки на рівні застосунку.
- Не завжди ефективна для складних реляційних зв’язків.
- Потребує досвіду для правильного проєктування структури даних, менш доцільна для початківців.

MySQL – популярна реляційна СКБД, яка широко використовується у веб-розробці завдяки простоті та високій швидкості роботи.

Redis – високопродуктивне сховище в оперативній пам’яті, що зазвичай використовується для кешування або тимчасового зберігання даних.

У ході аналізу сучасних систем керування базами даних було розглянуто переваги та недоліки ряду СКБД, які можуть бути використані для розробки Telegram-бота. У цьому випадку використання SQLite буде доцільним, оскільки є вбудованою СКБД, дані мають структурований характер та наявна спеціалізована бібліотека aiosqlite для роботи асинхронних ботів на Python.

2.3 Огляд інших технологій для розробки Telegram-бота

Окрім класичного підходу до розробки із використанням мов програмування та систем керування базами даних, існують також альтернативні рішення, які дозволяють створювати ботів без написання програмного коду (no-code) або з його мінімальним використанням (low-code). Такі технології актуальні для швидкого створення прототипів, невеликих сервісів або для користувачів без глибоких знань.

До основних альтернативних підходів створення Telegram-ботів належать:

- No-code платформи – сервіси, які дозволяють створювати ботів через графічний інтерфейс без написання коду.
- Low-code платформи – інструменти, що передбачають мінімальне програмування для розширення функціоналу.
- Конструктори чат-ботів – спеціалізовані сервіси для створення сценаріїв взаємодії з користувачем.
- Інтеграційні платформи – сервіси автоматизації, що дозволяють підключати Telegram до інших застосунків (наприклад, CRM, Google Sheets тощо).

Найбільш відомими прикладами таких інструментів є ManyChat, Chatfuel та Botpress.

ManyChat – це візуальний конструктор (drag-and-drop) для створення сценаріїв діалогу без програмування, платформа є одним із найпопулярніших інструментів для створення чат-ботів без програмування. Сервіс працює з

ботами в Instagram, Facebook Messenger, WhatsApp або Telegram, а також дозволяє налаштовувати автоматичні відповіді та розсилки повідомлень. З метою забезпечення персоналізації взаємодії, адміністратор бота може сегментувати аудиторію за допомогою тегів.

Недоліки:

- Обмежена гнучкість логіки та обмежені можливості для створення складних сервісів.
- Орієнтація переважно на маркетингові задачі.
- Залежність від тарифів платформи.

Chatfuel – це платформа-конструктор для створення чат-ботів, яка дозволяє будувати логіку взаємодії за допомогою блоків і переходів між ними. Сервіс забезпечує автоматизацію відповідей на повідомлення, підтримує використання штучного інтелекту для генерації відповідей та дозволяє збирати дані користувачів через інтерактивні форми. Платформа надає можливість створення розсилок, сегментації аудиторії та налаштування тригерів для запуску певних сценаріїв.

Недоліки:

- Обмежені можливості кастомізації інтерфейсу та логіки.
- Складніше масштабувати великі проєкти.
- Обмежена підтримка складних багатокрокових сценаріїв.
- Частково платний функціонал навіть для базових задач.

Botpress – це універсальна платформа для створення, розгортання та керування чат-ботами та агентами штучного інтелекту. Сервіс надає візуальний редактор для створення сценаріїв, а також дозволяє додавати власний код для реалізації складної логіки. Платформа підтримує інтеграцію з API, базами даних та іншими зовнішніми сервісами, що робить її придатною для створення складних програмних рішень. Botpress також забезпечує роботу з контекстом діалогу, дозволяючи зберігати інформацію про користувача та адаптувати відповіді. Додатково підтримуються інструменти штучного інтелекту, що дозволяють створювати розвинуті чат-боти.

Недоліки:

- Складніший у використанні, ніж класичні no-code рішення.
- Потребує технічних знань для повного використання можливостей.
- Надлишковий для простих ботів.
- Вища складність підтримки.

Огляд інших технологій для розробки Telegram-ботів демонструє цікаві альтернативні рішення для поставленої задачі у вигляді no-code та low-code розробок. Такі боти можуть бути створені та використані для простих задач, оскільки їхня робота здебільшого базується на заздалегідь визначених сценаріях. Їхніми перевагами, порівняно з традиційною розробкою, вважаються менший час розробки, спрощений процес створення бота та мінімальні вимоги щодо знань програмування. Водночас такі сервіси через обмежену гнучкість не дозволяють зробити гнучкого бота зі складною логікою взаємодії. Створення повноцінного програмного сервісу класичними підходами буде мати переваги у вигляді розширеного функціоналу та здатності до масштабування.

3 РОЗРОБКА TELEGRAM-БОТА

3.1 Опис предметної області

Предметною областю даної дипломної роботи є розробка Telegram-бота, який охоплює процеси організації, збереження, систематизації та контролю особистої активності користувача. У межах цього проєкту під особистою активністю маються на увазі повсякденні справи, навчальні та робочі завдання, які користувач бажає зафіксувати, впорядкувати та виконати у визначений час або зберігати важливу інформацію. Основним завданням ботів-помічників є автоматизація рутинних дій та полегшення життя користувача в цілому. Розроблюваний бот виконує роль цифрового щоденника для зберігання нотаток, створення нагадувань, структурування записів за списками, встановлення пріоритетів та відстеження статистики виконаних завдань. Telegram, як середовище функціонування бота, забезпечує кросплатформний доступ до системи з різних пристроїв, що забезпечує своєчасну доставку повідомлень користувачу, який, наприклад, працює через комп'ютер з Windows, навчається на планшеті з Android та використовує в особистому житті телефон з iOS.

Основним учасником предметної області є користувач, який використовує всі можливі функції бота. Взаємодія людини із системою відбувається за допомогою нижнього головного меню, текстових повідомлень, inline-кнопок та деяких службових команд, що дозволить зробити роботу з ботом зрозумілою навіть для користувачів із базовим рівнем цифрової грамотності.

Інформаційними об'єктами предметної області є запис, нотатка, нагадування, пріоритет, список, статус та статистика. Користувач є власником створених у боті даних та має доступ лише до власної інформації. Запис – узагальнена одиниця інформації, яка обов'язково належить до нагадування або нотатки. Нагадування містить текст, дату та час, пріоритет, статус та прив'язку

до списку. Нотатка, у свою чергу, має лише текст, пріоритет та формальну прив’язку до списку, оскільки для нотаток існує лише один список. Пріоритет дозволяє визначити важливість запису відповідним маркуванням та краще орієнтуватися серед створених нотаток і нагадувань. Списки використовуються для групування нагадувань, нотатки мають лише один загальний список. Наприклад, користувач може створювати окремі списки для навчання, роботи, побутових справ або особистих планів. Нагадування також мають статус, який відображає поточний стан цього запису, який може бути активним, надісланим або виконаним. Завдяки цьому система коректно обробляє нагадування та не надсилає вже виконані завдання. Статистика у цьому боті використовується для узагальнення активності користувача, як звіт про успішність виконання.

Інтерфейс взаємодії з ботом побудований таким чином, щоб користувач міг виконувати основні дії без запам’ятовування складних команд та сценаріїв. У деяких випадках передбачено підтвердження дії та кнопки «Назад» для зручності використання. Такі моменти забезпечують покрокову взаємодію та зменшують ймовірність помилок під час використання бота.

Межі предметної області розроблюваної системи визначаються її призначенням. Цей Telegram-бот не є повноцінною системою керування проєктами, корпоративною CRM-системою або календарним сервісом для колективної роботи. Його функціональність спрямована саме на індивідуальне використання: збереження нотаток, створення нагадувань, групування записів, пріоритезацію справ, контроль виконання та перегляд статистики. Такий обсяг функцій є достатнім для вирішення поставленої задачі, оскільки система спрямована на просту та швидку організацію особистої активності користувача.

3.2 Вимоги до оточення

3.2.1 Вимоги до апаратного забезпечення

Для ефективної роботи Telegram-бота буде достатньо мінімального обладнання, а саме базового ПК з Windows 7 або смартфона з iOS 11.0 / Android 6.0 для роботи користувацького вузла та ПК з Windows 10 для роботи серверного вузла.

Апаратне забезпечення для функціонування користувацького вузла Telegram-бота для ПК через Telegram Desktop:

Мінімальна апаратна конфігурація:

- Процесор – 1 ядро, 1.2 ГГц.
- Оперативна пам’ять – 1 ГБ.
- Дисплей – роздільна здатність 1024×768.
- Накопичувач – 200 МБ вільного місця.
- Мережа – доступ до Інтернету зі швидкістю 512 Кбіт/с.

Рекомендована апаратна конфігурація:

- Процесор – 2 ядра, 2.0 ГГц.
- Оперативна пам’ять – 4 ГБ і більше.
- Дисплей – роздільна здатність 1920×1080 (Full HD).
- Тип накопичувача – SSD.
- Мережа – доступ до Інтернету зі швидкістю 5 Мбіт/с.

Апаратне забезпечення для функціонування користувацького вузла Telegram-бота для смартфонів через Telegram:

Мінімальна апаратна конфігурація:

- Процесор – 4 ядра, 1.2 ГГц.
- Оперативна пам’ять – 1 ГБ.
- Дисплей – сенсорний.
- Внутрішня пам’ять – 150 МБ вільного місця.
- Мережа – мобільний Інтернет (3G) або Wi-Fi.

Рекомендована апаратна конфігурація:

- Процесор – 8 ядер, від 1.8 ГГц.
- Оперативна пам’ять – 3 ГБ і більше.
- Внутрішня пам’ять – 500 МБ вільного місця.
- Мережа – мобільний Інтернет (4G/5G) або стабільний Wi-Fi.

Апаратне забезпечення для функціонування користувацького вузла Telegram-бота для веб-версії Telegram:

Мінімальна апаратна конфігурація:

- Процесор – 1 ядро, 1.5 ГГц.
- Оперативна пам’ять – 2 ГБ.
- Дисплей – роздільна здатність 1024×768.
- Накопичувач – 100 МБ вільного місця.
- Мережа – доступ до Інтернету зі швидкістю 1 Мбіт/с.

Рекомендована апаратна конфігурація:

- Процесор – 2 ядра, 2.0 ГГц.
- Оперативна пам’ять – 4 ГБ.
- Дисплей – роздільна здатність від 1920×1080 (Full HD).
- Тип накопичувача – SSD.
- Мережа – доступ до Інтернету зі швидкістю 10 Мбіт/с.

Апаратне забезпечення для функціонування серверного вузла Telegram-бота:

Мінімальна апаратна конфігурація:

- Процесор – 1 ядро, 1.5 ГГц.
- Оперативна пам’ять – 2 ГБ.
- Накопичувач – 500 МБ вільного місця.
- Тип накопичувача – HDD або SSD.
- Мережа – стабільний доступ до Інтернету зі швидкістю 1 Мбіт/с.

Рекомендована апаратна конфігурація:

- Процесор – 2 ядра, 2.0 ГГц.
- Оперативна пам’ять – 8 ГБ.

- Накопичувач – 10 ГБ вільного місця.
- Тип накопичувача – SSD.
- Мережа – доступ до Інтернету зі швидкістю 10 Мбіт/с.

У ході тестування розроблюваного застосунку будуть використовуватися такі апаратні конфігурації:

Для ПК (як сервер та як користувач):

- Процесор – 4 ядра, 3.6 ГГц.
- Оперативна пам’ять – 16 ГБ.
- Накопичувач – 720 ГБ вільного місця.
- Тип накопичувача – HDD та SSD.
- Мережа – стабільний доступ до Інтернету зі швидкістю 35 Мбіт/с.

Для смартфону (як користувач):

- Процесор – 6 ядер, 3.23 ГГц.
- Оперативна пам’ять – 6 ГБ.
- Внутрішня пам’ять – 40 ГБ вільного місця.
- Мережа – мобільний Інтернет (5G) та стабільний Wi-Fi.

3.2.2 Вимоги до програмного забезпечення

Для функціонування Telegram-бота необхідно мати відповідне програмне середовище на користувацькому та серверному вузлах. Оскільки бот працює через інтерфейс Telegram, вимоги до користувацького пристрою будуть ідентичні вимогам до встановлення месенджеру.

Вимоги до програмного забезпечення користувацького вузла:

Застосунок: Telegram (версія 8.7 і вище), Telegram Desktop (версія 3.7 і вище), Google Chrome (версія 90+ і вище), Mozilla Firefox (версія 88+ і вище), Microsoft Edge (версія 90+ і вище) або Safari (версія 14+ і вище).

Операційна система: iOS (11.0 і вище), Android (6.0 і вище), macOS (10.13 і вище) або Windows (7 і вище).

Вимоги до програмного забезпечення серверного вузла:

Операційна система: Windows (10 і вище), Ubuntu (20.04 і вище) або macOS (10.13 High Sierra і вище).

Середовище виконання: Python (3.10 і вище).

СКБД: SQLite (3.x).

У ході тестування розроблюваного застосунку буде використовуватися таке програмне забезпечення:

Для ПК (як сервер та як користувач): Telegram Desktop (версія 6.7.2), Google Chrome (версія 146.0.7680.178), Windows (версія 10), Python (версія 3.14.3), SQLite.

Для смартфону (як користувач): Telegram (версія 12.6.1), iOS (версія 17.6.1).

3.2.3 Вимоги до користувачів

Основні типи користувачів:

1. Школяр / Студент.

Вік: 15-25 років.

Інтереси: Тайм-менеджмент, виконання завдань обмежених часом, підготовка до іспитів, відслідковування виконаних завдань, навчання та саморозвиток.

Мотивація: Тримати всі завдання у зручному місці, швидко додавати нагадування, бачити прогрес виконаних завдань.

Проблеми: Велика кількість завдань з різних дисциплін, складні застосунки для планування, відсутність щоденника, додаткові позашкільні активності.

Як користується ботом:

- Створює нагадування/нотатки про уроки/лекції.
- Налаштовує нагадування.
- Додає списки дисциплін.
- Відмічає виконані завдання.

- Пріоритезує задачі.
- Переглядає та аналізує статистику виконаних завдань.

2. Працюючий спеціаліст / фрілансер.

Вік: 20-40 років.

Інтереси: Якісне та вчасне виконання робочих задач, тайм-менеджмент, продуктивність, оптимізація робочих процесів, контроль задач та дедлайнів, work-life balance.

Мотивація: Керування робочими та особистими справами, швидкий доступ до нотаток без зайвих застосунків

Проблеми: Перевантаження інформацією, багатозадачність, тривожність про забуті справи.

Як користується ботом:

- Створює нагадування про робочі задачі.
- Додає нотатки про рутинні особисті справи.
- Сортує нотатки та нагадування по списках.
- Встановлює пріоритети.
- Редагує та видаляє задачі в будь-який момент.
- Використовує статистику як KPI.

3. Користувач з низькою технічною залученістю.

Вік: 40-60 років.

Інтереси: Освоєння технологій, зручне зберігання особистих справ та важливих дат, мінімізація складних застосунків.

Мотивація: легке створення нагадувань та нотаток, простий та зручний доступ до списку справ.

Проблеми: Складні застосунки, перевантажені інтерфейси, відсутність досвіду використання гаджетів.

Як користується ботом:

- Створює прості нагадування про важливі події.
- Створює нотатки про повсякденні справи.
- Відмічає завдання як виконані.

Поведінка користувачів:

Регулярні користувачі: Школярі, студенти, працівники; ті, хто активно планують свій час та мають багато завдань.

Активні користувачі: Люди, які створюють тільки важливі нагадування.

Одноразові користувачі: Люди, які використали бот, щоб ознайомитися з функціями.

Цифрові звички:

Час онлайн: 1-2 години на день.

Пристрої: Смартфони, планшети, комп’ютери, смарт-годинники з під’єднаним смартфоном.

Звички користування інтернетом: часто використовують месенджери, застосунки для планування, нагадувань.

Основні потреби користувачів:

- Простий, інтуїтивно зрозумілий інтерфейс для додавання, редагування та видалення нагадувань та нотаток.
- Створення списків та додавання пріоритетів.
- Можливість відмітити нагадування як виконане.
- Огляд статистики успішно виконаних завдань.

Вимоги до користувача

1. Технічні вимоги:

– Користувач повинен мати доступ до інтернету.
– Користувач використовує пристрій, який відповідає мінімальній апаратній конфігурації.
– Користувач використовує пристрій із встановленим та налаштованим програмним забезпеченням.
– Користувач має обліковий запис в Telegram.

2. Рівень цифрової грамотності:

– Користувач розуміє принцип роботи Telegram-ботів.
– Користувач базово вміє користуватися месенджерами.

– Користувач вміє виконувати базові дії у месенджерах (надсилає повідомлення, переходить між чатами).

3. Поведінкові вимоги:

– Користувач регулярно використовує бота.
– Користувач слідкує за своїми задачами та оновлює їх.
– Користувач отримує повідомлення від бота.
– Користувач коректно вводить дані.

4. Функціональні очікування від користувача:

– Користувач створює нагадування та нотатки.
– Користувач встановлює дату та час для нагадувань.
– Користувач надає пріоритет нагадуванню.
– Користувач переглядає нагадування та нотатки.
– Користувач створює списки.
– Користувач сортує нагадування по списках.
– Користувач відмічає завдання як виконане.
– Користувач переглядає статистику.

3.3 Архітектура системи

Для розроблюваного Telegram-бота було обрано клієнт-серверну архітектуру, що відповідає особливостям роботи ботів у месенджері Telegram. Користувач взаємодіє з ботом через клієнтський застосунок Telegram, який передає повідомлення та натискання кнопок через сервери Telegram до серверної частини бота, тому користувач не взаємодіє напряму з програмним кодом або базою даних. Серверна частина застосунку обробляє отримані дані, виконує необхідні операції, звертається до бази даних і надсилає відповідь користувачу через Telegram Bot API. З погляду користувача система представлена як один Telegram-бот із кнопковим інтерфейсом. Однак з погляду реалізації вона складається з декількох взаємопов’язаних підсистем:

- Клієнтський рівень користувача, представлений застосунком Telegram.
- Зовнішній сервіс Telegram Bot API, який забезпечує обмін повідомленнями між користувачем і ботом.
- Серверна частина Telegram-бота, у якій реалізовано основну логіку роботи системи.
- Підсистема збереження даних, представлена базою даних SQLite.

Клієнтська підсистема представлена застосунком Telegram, який користувач може мати на смартфоні, комп'ютері, планшеті або у вигляді веб-версії. Через цю підсистему користувач надсилає повідомлення, натискає кнопки головного меню, натискає inline-кнопки та отримує повідомлення з нагадуваннями. Клієнтська частина потребує встановлення додатку Telegram або будь-якого браузера, а також стабільного інтернету.

Зовнішній сервіс Telegram Bot API виконує роль моста між користувачем і серверною частиною бота. Він забезпечує отримання повідомлень від користувача, передачу оновлень серверній частині, доставку відповідей бота, а також підтримку роботи inline-кнопок і callback-запитів.

Серверна частина Telegram-бота є основною частиною розроблюваного програмного забезпечення. Вона реалізує логіку роботи: обробку команд, повідомлень і натискань кнопок, створення та редагування записів, надсилання нагадувань, формування статистики та звернення до бази даних.

Підсистема збереження даних – база даних SQLite. Вона використовується для збереження інформації про користувачів, списки, нотатки, нагадування, пріоритети, статуси та час виконання завдань. Усі дані зберігаються централізовано на серверному вузлі.

Діаграма розгортання демонструє фізичне розташування компонентів системи та спосіб їхньої взаємодії (див. Рис. 6). Користувачу потрібен застосунок Telegram, серверному вузлу потрібні середовище виконання Python,

застосунок Telegram-бота, база даних та сервіс нагадувань, хоча це частина застосунку, доцільніше буде показати її окремо через певні функції.

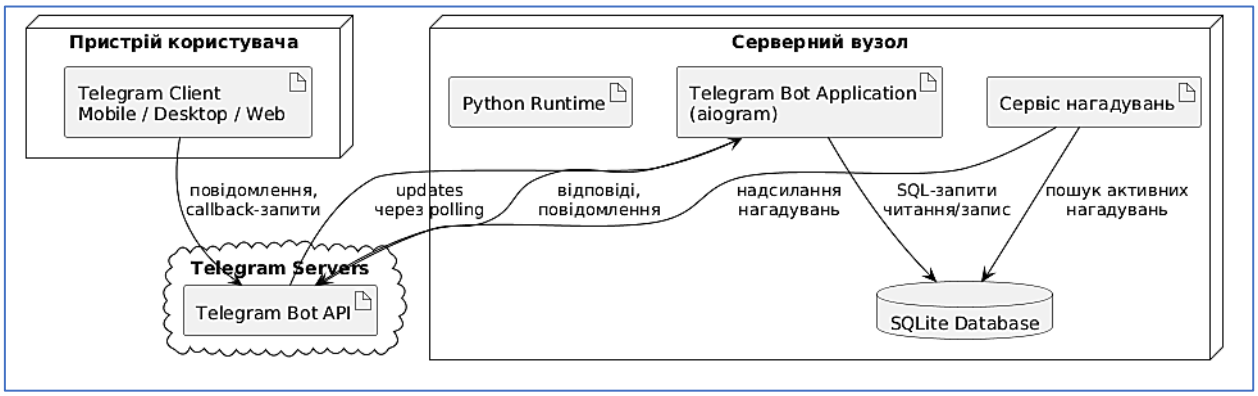


Рисунок 6 — *UML-діаграма розгортання.*

Діаграма пакетів демонструє основні структурні частини системи та залежності між ними на рівні архітектури (див. Рис. 7). Вона не деталізує внутрішню реалізацію кожного файлу, а показує, як основні компоненти програмного продукту взаємодіють між собою.

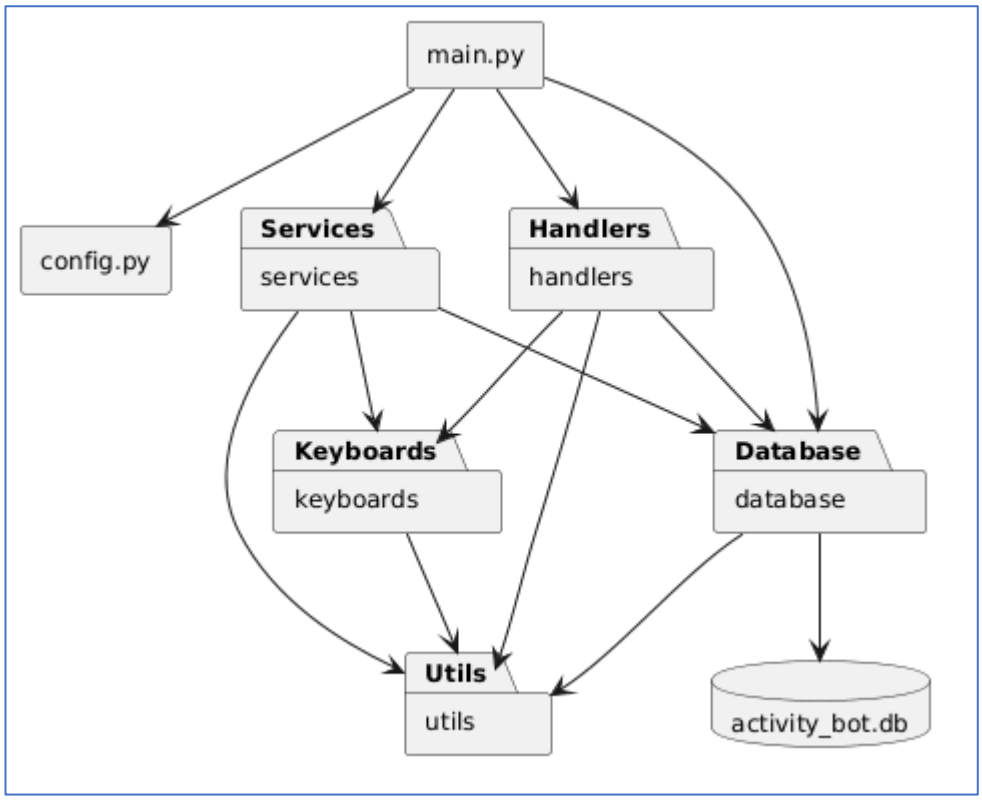


Рисунок 7 — *Діаграма пакетів.*

3.4 Специфікації даних

У Telegram-боті основним носієм даних є файл бази даних **activity_bot.db**. База даних складається з трьох таблиць (див. Рис. 8).

Таблиця users відповідає за дані користувача:

- id – внутрішній ідентифікатор користувача, створюється автоматично.
- telegram_id – унікальний Telegram ID користувача.
- username – ім’я користувача Telegram, може бути NULL.
- created_at – дата створення користувача.

Таблиця lists відповідає за списки:

- id – внутрішній ідентифікатор списку, створюється автоматично.
- user_id – посилання на користувача, зовнішній ключ.
- name – назва списку.
- created_at – дата створення списку.

Таблиця items відповідає за записи:

- id – внутрішній ідентифікатор запису, створюється автоматично.
- user_id – посилання на користувача, зовнішній ключ.
- list_id – посилання на список, зовнішній ключ.
- text – текст запису.
- item_type – тип запису: нотатка або нагадування.
- remind_at – дата і час для нагадувань, NULL для нотаток.
- priority – пріоритет: високий, середній, низький або без пріоритету.
- status – статус для нагадувань: активне, надіслане або виконане, NULL для нотаток.
- created_at – дата створення запису.
- completed_at – дата виконання нагадування.



Рисунок 8 — *ER-діаграма бази даних.*

3.5 Функціональні та нефункціональні вимоги

Одним із важливих етапів у розробці будь-якого програмного забезпечення є формування функціональних та нефункціональних вимог, оскільки саме вони визначають очікувану поведінку системи, її можливості, обмеження та якісні характеристики. Основою для формування вимог є аналіз предметної області, огляд існуючих рішень та потреби майбутніх користувачів. Функціональні вимоги описують дії, які система повинна виконувати. Нефункціональні вимоги визначають якісні характеристики системи, зокрема зручність використання, надійність, сумісність, безпеку та продуктивність.

3.5.1 Функціональні вимоги

- Після першого запуску бота через команду /start система повинна автоматично створювати запис користувача в базі даних на основі його Telegram ID. Якщо користувач уже існує в базі даних, система повинна використовувати наявний запис у майбутній роботі та не створювати дублікати.
- Система повинна надавати користувачу доступ до основних функцій через нижнє меню Telegram. Головне меню повинно містити кнопки для додавання записів, перегляду списків, перегляду статистики та отримання допомоги.
- Система повинна надавати користувачу коротку інструкцію щодо своїх можливостей. Довідкова інформація повинна бути доступна через кнопку «Допомога» та команду /help.
- Система повинна дозволяти користувачу створити новий запис. Під час створення бот повинен послідовно отримати від користувача текст запису, тип запису та пріоритет. Якщо користувач обирає тип «нагадування», система додатково повинна запросити дату й час сповіщення та надати можливість обрати список для збереження.

- Система повинна забезпечувати користувачу можливість створювати, переглядати, редагувати та видаляти текстові нотатки. Нотатка повинна містити текст та пріоритет, зберігатися в обраному списку, не мати прив’язки до конкретної дати й часу та не мати статусу.
- Система повинна надавати користувачу можливість створювати, переглядати, редагувати та видаляти нагадування. Нагадування повинно містити текст, дату й час сповіщення, список, пріоритет та статус.
- Під час створення або редагування нагадування система повинна перевіряти коректність введеної дати та часу. У разі неправильного формату бот повинен повідомити користувача про помилку та запропонувати повторити введення або обрати із запропонованих.
- Система повинна автоматично перевіряти наявність активних нагадувань, час яких настав, і надсилати відповідне повідомлення користувачу в Telegram. Після надсилання повідомлення статус нагадування повинен оновлюватися.
- Система повинна дозволяти користувачу позначити нагадування як виконане. Після виконання статус запису повинен змінюватися, а відповідна інформація повинна враховуватися під час формування статистики.
- Якщо користувач не може виконати завдання в момент отримання повідомлення, система повинна надавати можливість відкласти нагадування. Після відкладення дата або час нагадування повинні оновлюватися відповідно до реалізованої логіки.
- Система повинна дозволяти користувачу створювати, переглядати та видаляти списки. Списки використовуються для групування нагадувань за тематикою або сферою діяльності, нотатки мають єдиний список.
- Користувач повинен мати можливість обрати певний список і переглянути записи, які до нього належать. У межах списку повинні відображатися нотатки або нагадування користувача.

- Користувач повинен мати можливість видалити нотатку або нагадування. Перед видаленням система повинна запитувати підтвердження дії, щоб уникнути випадкового видалення.
- Користувач повинен мати можливість видалити створений список. Якщо список містить записи, система повинна коректно обробити пов’язані з ним дані відповідно до реалізованої логіки.
- Система повинна дозволяти користувачу встановлювати пріоритет для запису. Пріоритет може бути низьким, середнім, високим або відсутнім. Обраний пріоритет повинен зберігатися в базі даних і відображатися під час перегляду запису.
- Система повинна дозволяти користувачу редагувати запис. Для нотаток повинна бути можливість змінити текст та пріоритет. Для нагадувань також повинна бути можливість змінити текст, дату, час, пріоритет і статус.
- У процесі роботи з inline-кнопками бот повинен надавати можливість повернення до попереднього меню або списку записів. Це забезпечує зручність навігації та зменшує кількість помилкових дій користувача.
- Під час переходу між сценаріями бот повинен прибирати або оновлювати старі inline-клавіатури, щоб користувач не міг виконати застарілу дію.
- Якщо кількість списків або записів є великою, система повинна забезпечувати їх посторінковий перегляд за допомогою кнопок навігації.
- Система повинна формувати та відображати статистику користувача. Статистика повинна містити інформацію про кількість створених списків, нотаток, нагадувань та виконаних завдань.
- Система повинна відображати користувачу лише його власні списки, нотатки, нагадування та статистику. Дані інших користувачів не повинні бути доступні.

Use Case Diagram демонструє вищезазначені функції та деякі залежності між ними (див. Рис. 9).

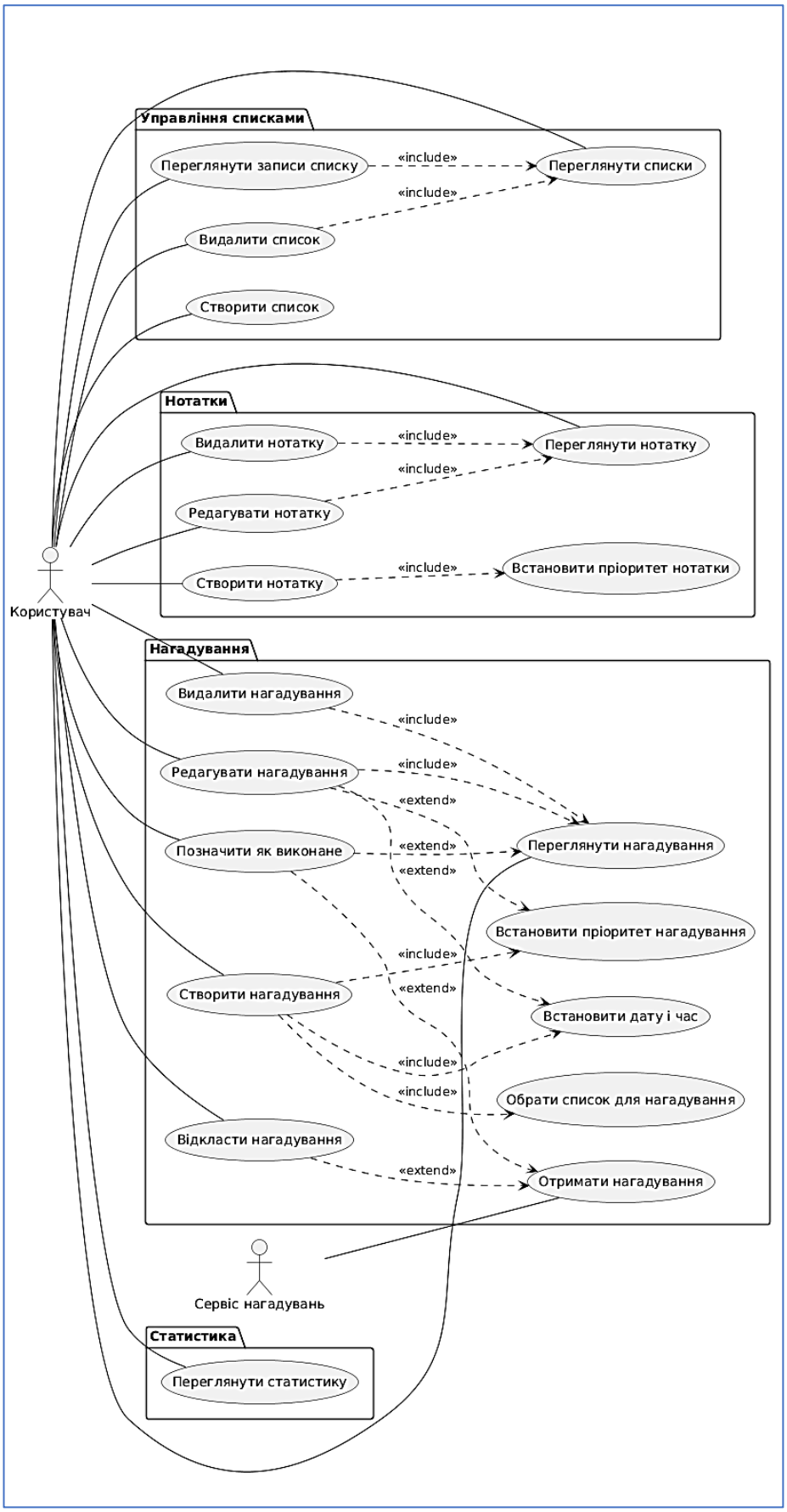


Рисунок 9 — *Фінальна Use Case Diagram.*

3.5.2 Нефункціональні вимоги

- Інтерфейс бота повинен бути простим, зрозумілим та інтуїтивним. Основні дії повинні виконуватися за допомогою кнопок, а не лише текстових команд.
- Система повинна коректно працювати на всіх пристроях, які підтримують месенджер Telegram: смартфонах, планшетах, персональних комп’ютерах і веб-версії Telegram.
- Бот повинен коректно зберігати дані користувача, надсилати нагадування у встановлений час і продовжувати роботу після тимчасових помилок або перезапуску.
- Система повинна забезпечувати ізоляцію даних користувачів. Ідентифікація користувача повинна виконуватися за унікальним Telegram ID.
- Дані в базі повинні зберігатися в узгодженому стані. Кожен запис повинен бути пов’язаний із конкретним користувачем і списком.
- Бот повинен швидко реагувати на дії користувача. Відповідь на типові команди, натискання кнопок і перехід між меню повинні виконуватися без помітної затримки для користувача.
- Програмний код повинен бути структурований за модулями: обробники команд, клавіатури, робота з базою даних, службові функції та сервіс нагадувань. Такий підхід спрощує подальше редагування й розширення функціоналу.
- Система повинна мати можливість подальшого розширення. У майбутньому до бота можуть бути додані повторювані нагадування, розширена статистика, експорт даних або інтеграція з іншими сервісами.
- У разі виникнення помилки під час обробки повідомлення або callback-запиту бот не повинен припиняти роботу повністю. Помилки повинні оброблятися таким чином, щоб користувач міг продовжити взаємодію із системою.

- Текстові повідомлення, кнопки та підказки бота повинні бути представлені українською мовою, оскільки система орієнтована на україномовного користувача.
- Бот повинен бути доступним користувачу за умови стабільної роботи серверного вузла, підключення до інтернету та доступності сервісів Telegram.
- У разі некоректного введення дати, часу або іншої інформації бот повинен надавати користувачу зрозуміле пояснення та підказку щодо правильного формату введення.
- У разі якщо користувач надсилає неочікувані повідомлення, система повинна повідомити користувачу про це та підказати, що робити далі.

Сформовані вимоги до системи допомагають визначити необхідний функціонал та характеристики розроблюваного додатку (див. Рис. 10). На основі цих вимог здійснюється подальше проєктування архітектури, структури бази даних, інтерфейсу та алгоритмів роботи Telegram-бота.

3.6 Засоби реалізації

Враховуючи аналіз мов програмування, систем керування базами даних та інших підходів для розробки Telegram-бота у попередніх розділах, а також дослідницьку роботу [1], було обрано технологічний стек, орієнтований на простоту розробки, достатню продуктивність, підтримку асинхронної взаємодії та зручність подальшого супроводу програмного продукту. Результатом вибору стало використання мови програмування Python, бібліотеки aiogram для створення Telegram-бота, SQLite як системи збереження даних та aiosqlite для асинхронної роботи з базою даних.

Для створення програмного коду Telegram-бота використовується середовище розробки та редактор коду **Visual Studio Code**. Воно забезпечує зручну роботу з Python, підсвічування синтаксису, автодоповнення, роботу з файлами проєкту та інтегрований термінал. Наявність терміналу в середовищі

розробки дозволила запускати бота, встановлювати залежності та перевіряти роботу програми без використання додаткових інструментів.

Обраною мовою програмування стала мова **Python**, завдяки простому синтаксису, великій кількості бібліотек для створення Telegram-ботів та підтримці асинхронного програмування. У проєкті Python використовується для реалізації серверної логіки бота, обробки повідомлень, роботи з базою даних, створення клавіатур, перевірки нагадувань і формування відповідей користувачу.

Для взаємодії з Telegram Bot API використано фреймворк **aiogram**. Він забезпечує обробку команд, повідомлень, callback-запитів, inline-кнопок, головного меню та станів користувача. Завдяки aiogram у боті реалізовано покрокові сценарії створення нотаток і нагадувань, редагування записів, видалення, відкладення нагадувань та перегляд статистики.

Обраною системою керування базами даних стала **SQLite**, оскільки вона не потребує окремого сервера бази даних і зберігає всю інформацію в одному файлі. У базі даних зберігаються користувачі, списки, нотатки, нагадування, пріоритети та статуси. Для поточного масштабу дипломного проєкту SQLite є достатнім і зручним рішенням.

Для роботи з SQLite використовується бібліотека **aiosqlite**, яка дозволяє виконувати SQL-запити в асинхронному режимі. Це важливо, оскільки бот повинен швидко реагувати на дії користувача та паралельно виконувати фонову перевірку нагадувань. Через aiosqlite реалізовано створення, читання, редагування та видалення даних у базі.

Для організації асинхронної роботи використовується стандартний модуль Python **asyncio**. Він дозволяє запускати основний процес бота та фоновий сервіс нагадувань одночасно. Завдяки цьому бот може обробляти повідомлення користувачів і паралельно перевіряти, чи настав час для надсилання активних нагадувань.

Для збереження службових параметрів використовуються файли **.env** та **config.py**. У .env зберігаються конфіденційні дані, зокрема токен Telegram-

бота, а config.py забезпечує доступ до цих параметрів у програмному коді. Такий підхід дозволяє не зберігати токен безпосередньо в основних файлах проєкту.

Для ізоляції бібліотек проєкту використовується віртуальне середовище **venv**. Залежності встановлюються за допомогою менеджера пакетів **pip**, а їх перелік зберігається у файлі **requirements.txt**. Це дозволяє швидко підготувати середовище для запуску бота на іншому комп'ютері або серверному вузлі.

Розробка та тестування Telegram-бота виконувалися в операційній системі **Windows**. Для запуску серверної частини необхідні встановлений Python, доступ до інтернету та коректно налаштований токен Telegram-бота. Користувацька частина потребує операційної системи пристрою, встановлений застосунок Telegram та доступ до інтернету.

Для зручності запуску Telegram-бота у структурі проєкту передбачені службові **bat-файли**. Вони є корисними під час розробки, тестування та демонстрації програмного продукту в середовищі Windows. Один з файлів використовується для підготовки середовища: встановлення залежностей та виконання інших початкових дій, необхідних перед запуском бота. Інший файл призначений для швидкого запуску програми без необхідності щоразу вводити команду вручну в терміналі.

3.7 Модулі та алгоритми

Програмна реалізація Telegram-бота має модульну структуру, що дозволяє розділяти відповідальність між окремими логічними частинами системи. Структура проєкту побудована відповідно до основних напрямів роботи бота: запуск програми, обробка дій користувача, формування клавіатур, робота з базою даних, службові функції та фоновий сервіс нагадувань. Файлова архітектура має такий вигляд (див. Рис. 10).

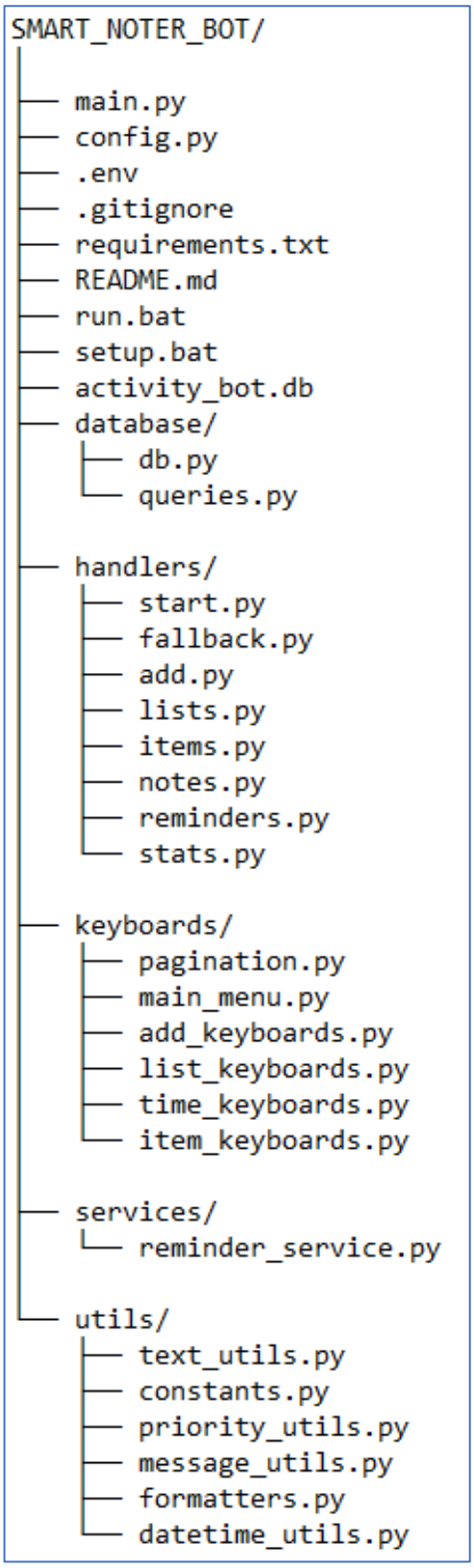


Рисунок 10 — *Файлова структура Telegram-бота.*

1. **Файли кореневого каталогу:**

Файл **main.py** є головною точкою входу до програми. У ньому виконується ініціалізація бази даних, створення об’єкта бота, створення диспетчера, підключення маршрутизаторів з обробниками та запуск фонового сервісу нагадувань.

Файл **config.py** відповідає за отримання конфігураційних параметрів, необхідних для запуску системи. Основним таким параметром є токен Telegram-бота, який використовується для підключення до Telegram Bot API.

Файл **.env** використовується для збереження конфіденційних параметрів середовища, зокрема токена бота. Файл **requirements.txt** містить перелік зовнішніх бібліотек, необхідних для роботи проєкту. Файли **setup.bat** та **run.bat** використовуються для спрощення налаштування середовища й запуску Telegram-бота в операційній системі Windows. Файл **activity_bot.db** є файлом бази даних SQLite, у якому зберігається інформація користувачів.

2. **Модуль database:**

Файл **db.py** призначений для ініціалізації бази даних. У ньому створюються необхідні таблиці, визначаються зв’язки між ними та задаються обмеження цілісності даних. Цей модуль викликається під час запуску програми, щоб гарантувати наявність потрібної структури бази перед початком роботи бота.

Файл **queries.py** містить функції для виконання SQL-запитів. Через цей модуль обробники та сервіс нагадувань працюють із даними користувачів. Основними операціями цього модуля є створення користувача, отримання списків, створення записів, оновлення тексту, зміна пріоритету, зміна часу нагадування, позначення завдання як виконаного, видалення записів та отримання статистичних даних.

Лістинг 1 — *Код створення користувача або пошуку існуючого.*

```
    async def get_or_create_user(telegram_id: int, username:
str | None):
        async with aiosqlite.connect(DB_NAME) as db:
```

```
            cursor = await db.execute(
                 "SELECT id FROM users WHERE telegram_id =
?",
                (telegram_id,)
            )
            user = await cursor.fetchone()

            if user:
                user_id = user[0]
                await _ensure_default_lists(db, user_id)
                await db.commit()
                return user_id

            cursor = await db.execute(
                 "INSERT INTO users (telegram_id, username)
VALUES (?, ?)",
                (telegram_id, username)
            )

            user_id = cursor.lastrowid

            await _ensure_default_lists(db, user_id)
            await db.commit()

            return user_id
```

Лістинг 2 — Код отримання записів зі списку користувача.

```
    async def get_items_by_list(user_id: int, list_id: int):
        async with aiosqlite.connect(DB_NAME) as db:
            db.row_factory = aiosqlite.Row
            cursor = await db.execute("""
                   SELECT id, text, item_type, remind_at,
priority, status, created_at
                FROM items
                WHERE user_id = ? AND list_id = ?
                ORDER BY created_at DESC
            """, (user_id, list_id))

            rows = await cursor.fetchall()
            return [dict(row) for row in rows]
```

Лістинг 3 — Код відкладання нагадування.

```
    async def update_item_remind_at(user_id: int, item_id:
int, new_remind_at: str):
        async with aiosqlite.connect(DB_NAME) as db:
            await db.execute("""
```

```
        UPDATE items
        SET remind_at = ?,
            status = ?,
            completed_at = NULL
        WHERE id = ?
          AND user_id = ?
          AND item_type = ?
          AND status != ?
    """, (
        new_remind_at,
        STATUS_ACTIVE,
        item_id,
        user_id,
        ITEM_TYPE_REMINDER,
        STATUS_COMPLETED
    ))

    await db.commit()
```

3. Модуль handlers:

Файл **start.py** відповідає за початкову взаємодію з користувачем. У ньому обробляється команда /start, створюється або знаходиться користувач у базі даних, після чого бот надсилає вітальне повідомлення та відображає головне меню. Також у цьому модулі може оброблятися команда /help або кнопка допомоги.

Файл **fallback.py** призначений для обробки неочікуваних повідомлень користувача. Якщо користувач надсилає текст, який не відповідає поточному сценарію або відомій команді, бот повинен не завершувати роботу з помилкою, а надати зрозумілу відповідь і підказати, які дії доступні.

Файл **add.py** реалізує сценарій додавання нового запису. У цьому модулі користувач послідовно вводить текст запису, обирає тип запису, дату й час для нагадування, пріоритет і список. Для організації такого покрокового сценарію використовуються стани, які дозволяють боту визначати, на якому етапі введення даних перебуває користувач.

Лістинг 4 — *Код додавання запису.*

```
@router.message(Command("add"))
```

```
    @router.message(F.text == "➕ Додати")
    async def add_start(message: Message, state:
FSMContext):
        await remove_old_inline_keyboard(message, state)
        await state.clear()
        user_id = await get_or_create_user(
            message.from_user.id,
            message.from_user.username
        )
        await state.update_data(user_id=user_id)
        await state.set_state(AddItemStates.waiting_text)
        await message.answer("Введіть текст запису:")
```

Файл **lists.py** відповідає за роботу зі списками. У ньому реалізовані сценарії перегляду списків, створення нового списку, видалення списку та переходу до записів, які належать певному списку.

Файл **items.py** використовується для загальної роботи із записами. Оскільки нотатки й нагадування мають спільні властивості, наприклад текст і пріоритет, частина логіки може бути винесена до загального обробника.

Файл **notes.py** містить обробники, пов’язані з нотатками. Нотатки використовуються для збереження текстової інформації без прив’язки до конкретної дати та часу. У цьому модулі реалізується перегляд, редагування й видалення нотаток.

Файл **reminders.py** містить обробники, пов’язані з нагадуваннями. У ньому реалізується перегляд нагадувань, редагування тексту, дати, часу, пріоритету, позначення завдання як виконаного та відкладення нагадування.

Файл **stats.py** відповідає за формування та відображення статистики. Він отримує дані з бази, обробляє їх і надсилає користувачу узагальнену інформацію про кількість списків, нотаток, нагадувань та виконаних завдань.

4. Модуль keyboards:

Файл **main_menu.py** містить головне меню бота. Воно забезпечує доступ до основних розділів: додавання запису, перегляду списків, швидкого перегляду нагадувань та нотаток, статистики та допомоги.

Файл **add_keyboards.py** використовується у сценаріях створення записів. У ньому формуються кнопки вибору типу запису, вибору пріоритету або інших дій, які потрібні під час додавання нотатки чи нагадування.

Файл **list_keyboards.py** містить клавіатури для роботи зі списками. Вони використовуються для вибору списку, створення нового списку, видалення та повернення до попереднього меню.

Файл **time_keyboards.py** використовується для вибору часу або дій, пов’язаних із датою та часом нагадування. Такі клавіатури спрощують введення користувача та зменшують кількість помилок.

Файл **item_keyboards.py** містить клавіатури для роботи з конкретними записами. Через них користувач може редагувати запис, змінювати пріоритет, видаляти запис, позначати нагадування як виконане або відкладати його.

Лістинг 5 — *Код створення клавіатури з вибором запису.*

```
    def items_keyboard(items, list_id: int, page: int = 0):
                page_items,     page,     total_pages     =
get_page_items(items, page)
        keyboard = []
        for item in page_items:
                  icon   =   "📝"   if   item["item_type"]   ==
ITEM_TYPE_NOTE else "🔔"
                    short_text   =   cut_text(item["text"],
ITEM_BUTTON_TEXT_LENGTH)
                   status   =   "✅"   if   item["status"]   ==
STATUS_COMPLETED else ""

            keyboard.append([
                InlineKeyboardButton(
                    text=f"{icon} {status} {short_text}",
                    callback_data=f"item_open:{item['id']}"
                )
            ])

        keyboard.append(
            pagination_row(
                page=page,
                total_pages=total_pages,
                back_callback="lists_view",
```

```
                prev_callback=f"items_page:{list_id}:{page -
1}",
                next_callback=f"items_page:{list_id}:{page +
1}"
            )
        )
                                                    return
InlineKeyboardMarkup(inline_keyboard=keyboard)
```

Файл **pagination.py** відповідає за формування клавіатур посторінкової навігації. Якщо кількість списків або записів перевищує зручну для одного повідомлення кількість, користувач може переходити між сторінками за допомогою кнопок навігації.

5. Модуль services:

Файл **reminder_service.py** реалізує фоновий сервіс нагадувань. Його завдання полягає в регулярній перевірці бази даних на наявність активних нагадувань, час яких настав. Якщо такі нагадування знайдені, сервіс надсилає користувачам повідомлення через Telegram Bot API та оновлює статус відповідних записів у базі даних.

Лістинг 6 — *Код сервісу для нагадувань.*

```
async def reminder_worker(bot: Bot):
    while True:
        reminders = await get_due_reminders()
        for reminder in reminders:
            try:
                await bot.send_message(
                    chat_id=reminder["telegram_id"],
                    text=(
                        "🔔 Нагадування\n\n"
                        f"{reminder['text']}\n\n"
                                        f"Пріоритет:
{format_priority(reminder['priority'])}\n"
                      f"Список: {reminder['list_name']}\n\n"
                        "Відкласти на:"
                    ),
                     reply_markup=reminder_notification_keyb
oard(reminder["id"])
                )
```

```
                    await mark_item_status(reminder["id"],
STATUS_NOTIFIED)
            except Exception:
                logger.exception(
                        "Не вдалося надіслати нагадування з
id=%s",
                    reminder["id"]
                )
        await asyncio.sleep(REMINDER_CHECK_INTERVAL)
```

6. Модуль utils:

Файл **constants.py** містить сталі значення, які використовуються в системі: типи записів, статуси нагадувань, стандартні назви або інші службові параметри.

Файл **datetime_utils.py** відповідає за роботу з датою та часом. У ньому можуть міститися функції перевірки формату дати, перетворення введення користувача у формат, придатний для збереження в базі даних, а також службові функції для обчислення нового часу нагадування.

Файл **formatters.py** використовується для форматування тексту повідомлень, які надсилаються користувачу. Це дозволяє підтримувати єдиний стиль відображення нотаток, нагадувань, списків і статистики.

Файл **message_utils.py** містить допоміжні функції для роботи з повідомленнями Telegram. Зокрема, такі функції можуть використовуватися для видалення або оновлення старих inline-клавіатур, збереження активного повідомлення та запобігання виконанню застарілих дій.

Файл **priority_utils.py** відповідає за обробку пріоритетів. У ньому можуть міститися функції для перетворення пріоритету у зручний для відображення формат.

Файл **text_utils.py** використовується для службової обробки тексту, зокрема для обмеження кількості символів.

Таким чином, усі функції системи розділені на логічні модулі, що дозволяє інтуїтивно розуміти, за які частини відповідає певний код. Взаємодія між модулями відбувається постійно, майже усі модулі звертаються до utils,

оскільки там розміщені службові функції, що дозволяє не повторювати код і загалом покращити його якість.

3.8 Проєкт інтерфейсу

Загальна логіка взаємодії користувача з Telegram-ботом представлена на діаграмі User Flow (див. Рис. 11). Вона демонструє основні сценарії роботи з ботом: запуск системи, перехід до головного меню, додавання нотаток і нагадувань, перегляд списків, редагування записів та перегляд статистики.

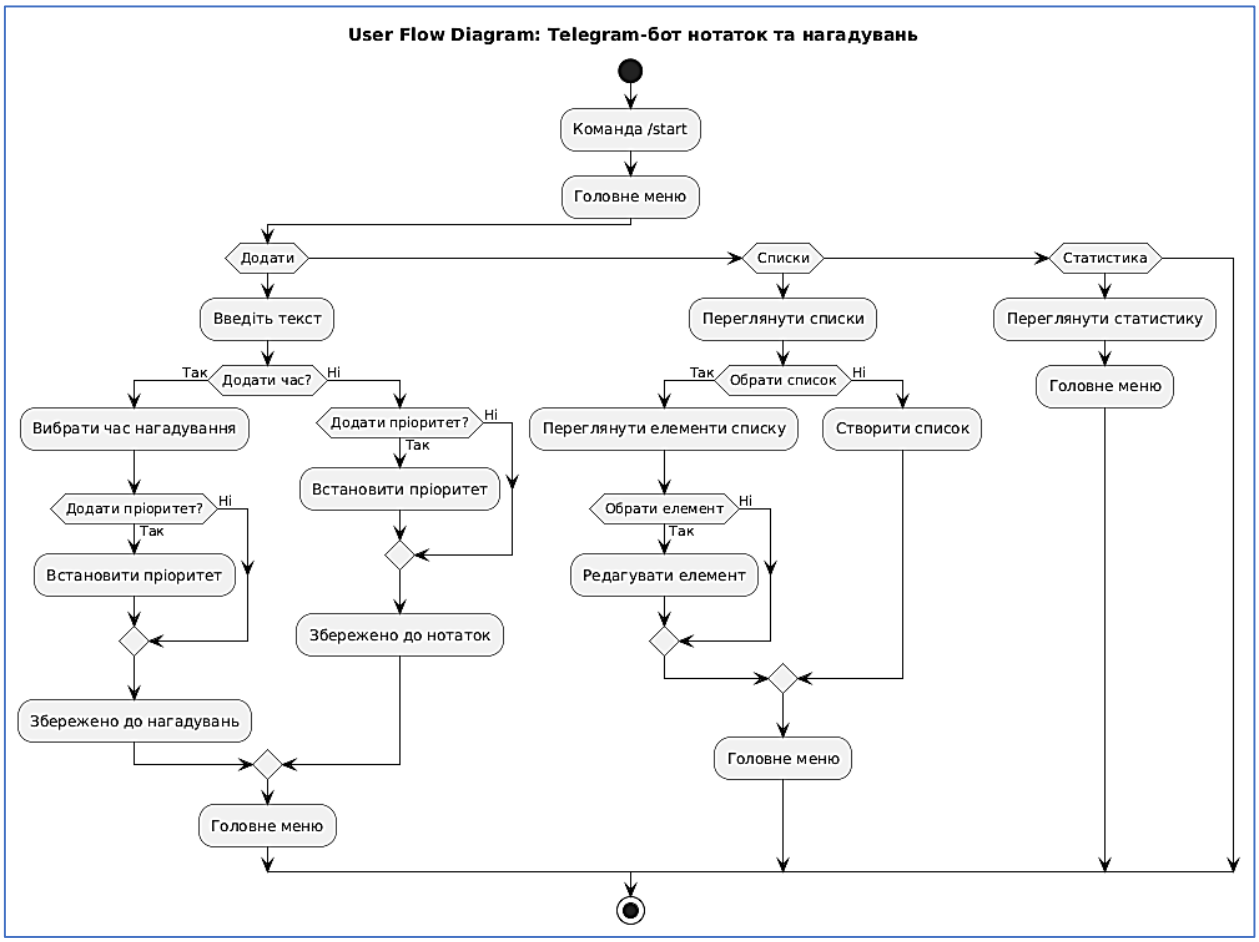


Рисунок 11 — User Flow Diagram.

Інтерфейс Telegram-бота містить здебільшого текстові повідомлення у самому застосунку Telegram, але є також два варіанти кнопок, через які можна

взаємодіяти. Коли користувач вперше знаходить бота, його зустрічає привітальне повідомлення (див. Рис. 12). Воно описує основні функції та задачі бота, праворуч знаходиться короткий опис.

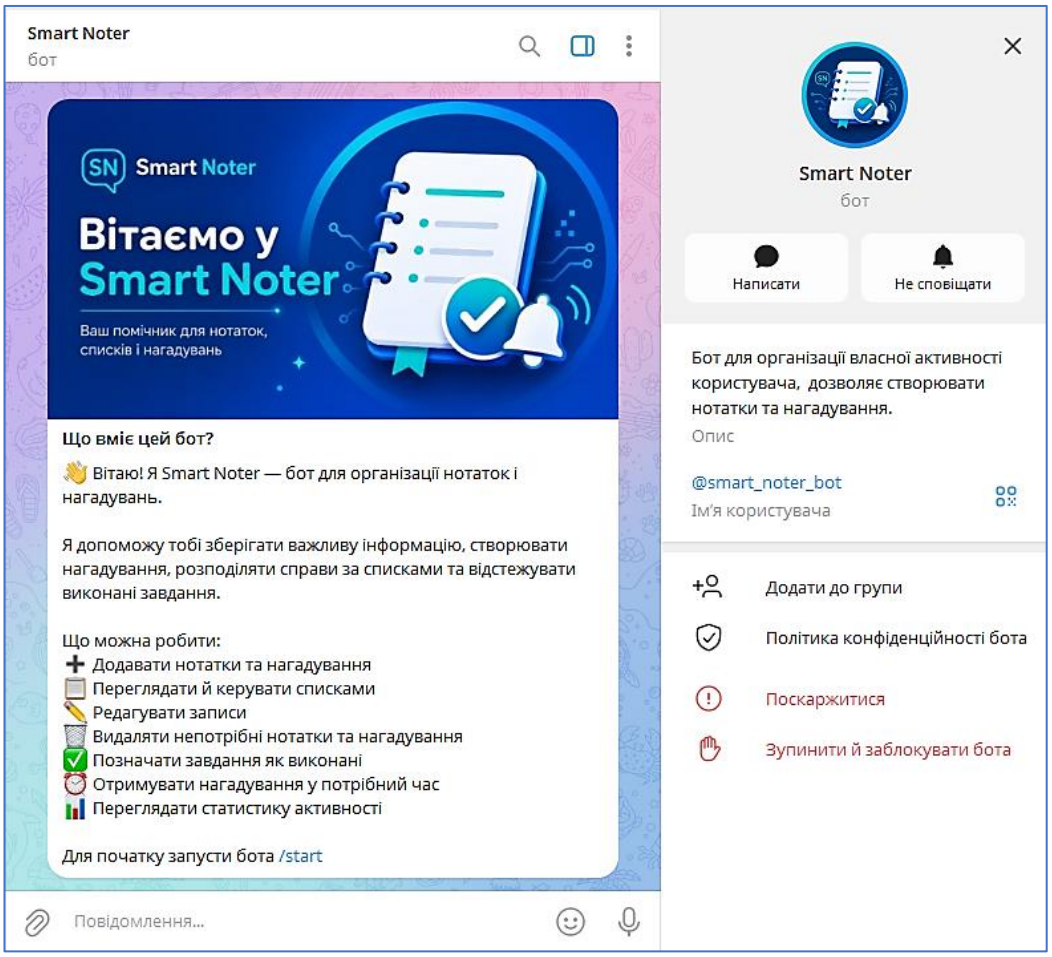


Рисунок 12 — *Інтерфейс бота перед запуском.*

Усі можливі сценарії стартують з нижнього головного меню або через службові команди, які починаються з символу-префіксу «/» (див. Рис. 13).

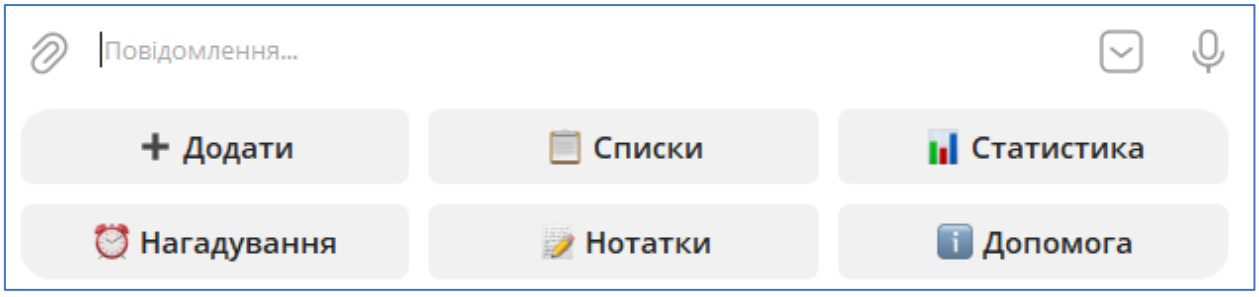


Рисунок 13 — *Головне меню.*

При натисканні на кнопку «Допомога» або введенні команди /help користувачу прийде повідомлення з описом кожної кнопки (див. Рис. 14).

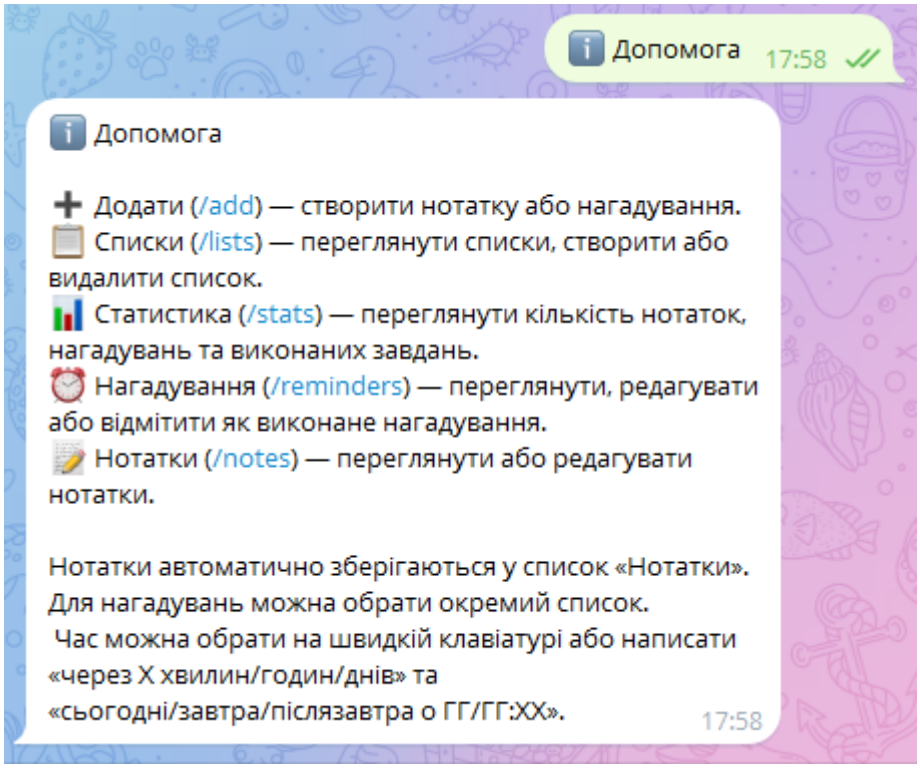


Рисунок 12 — *Кнопка «Допомога».*

При натисканні кнопки «Додати» бот попросить ввести текст запису та дозволить обрати, чим є цей запис: нагадуванням чи нотаткою (див. Рис. 15). Якщо вибрати нагадування, то потрібно ввести час або обрати із запропонованих варіантів (див. Рис. 16), бот розуміє природну мову при введенні часу (див. Рис. 17). Потім для нагадувань можна обрати список та пріоритет (див. Рис. 18). У разі якщо це нотатка, можна обрати лише пріоритет. Наприкінці бот повідомляє про успішне збереження (див. Рис. 19).

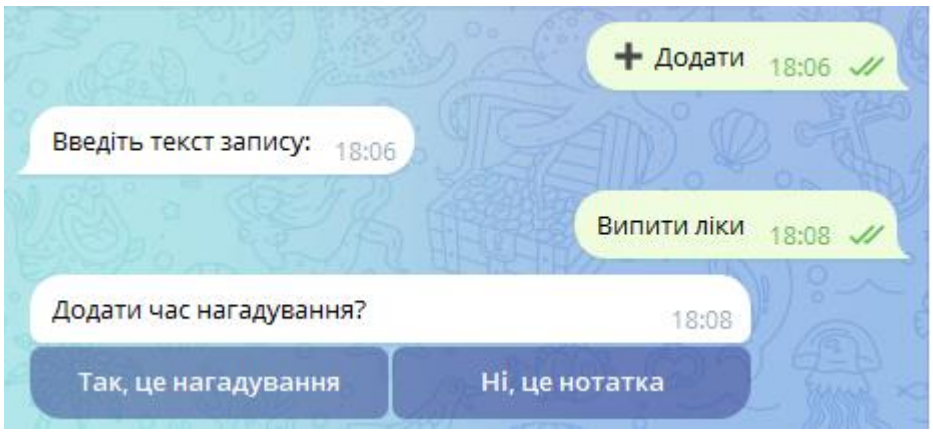


Рисунок 13 — *Кнопка "Додати".*

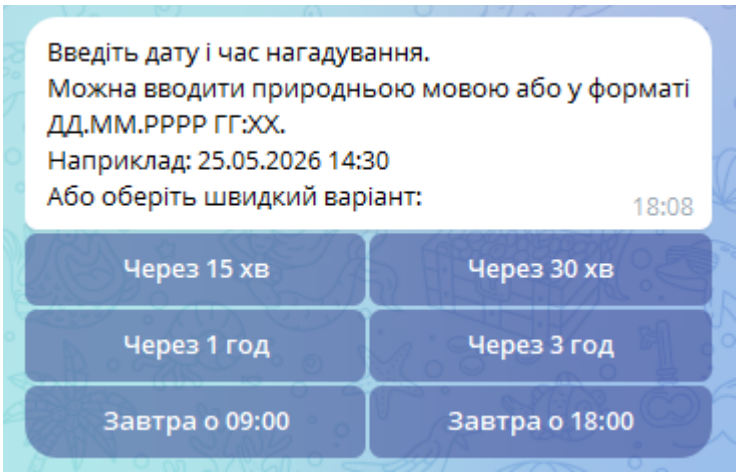


Рисунок 14 — *Вибір часу.*

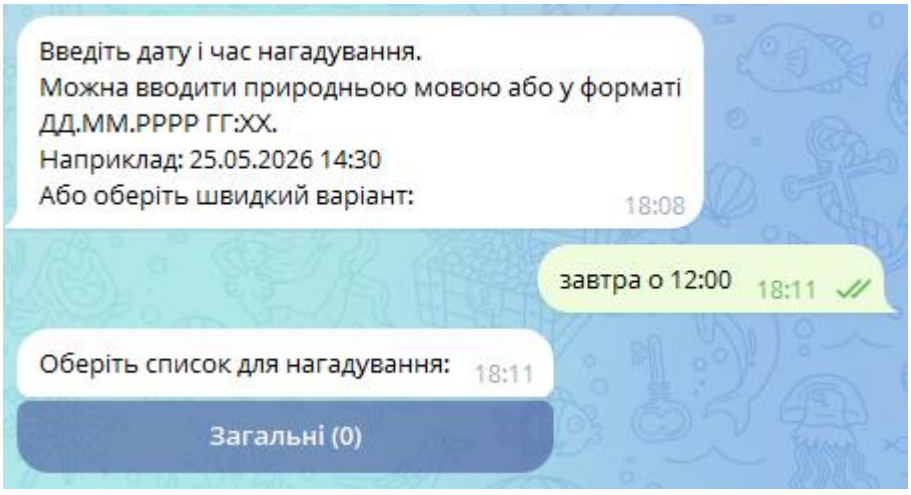


Рисунок 15 — *Вибір списку.*

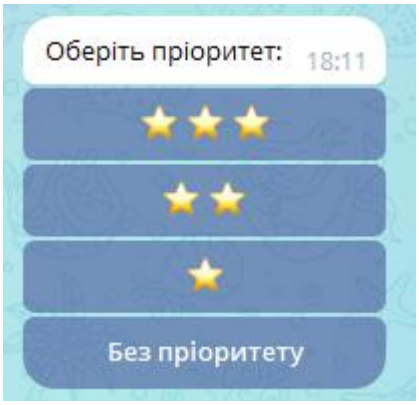


Рисунок 16 — *Вибір пріоритету.*

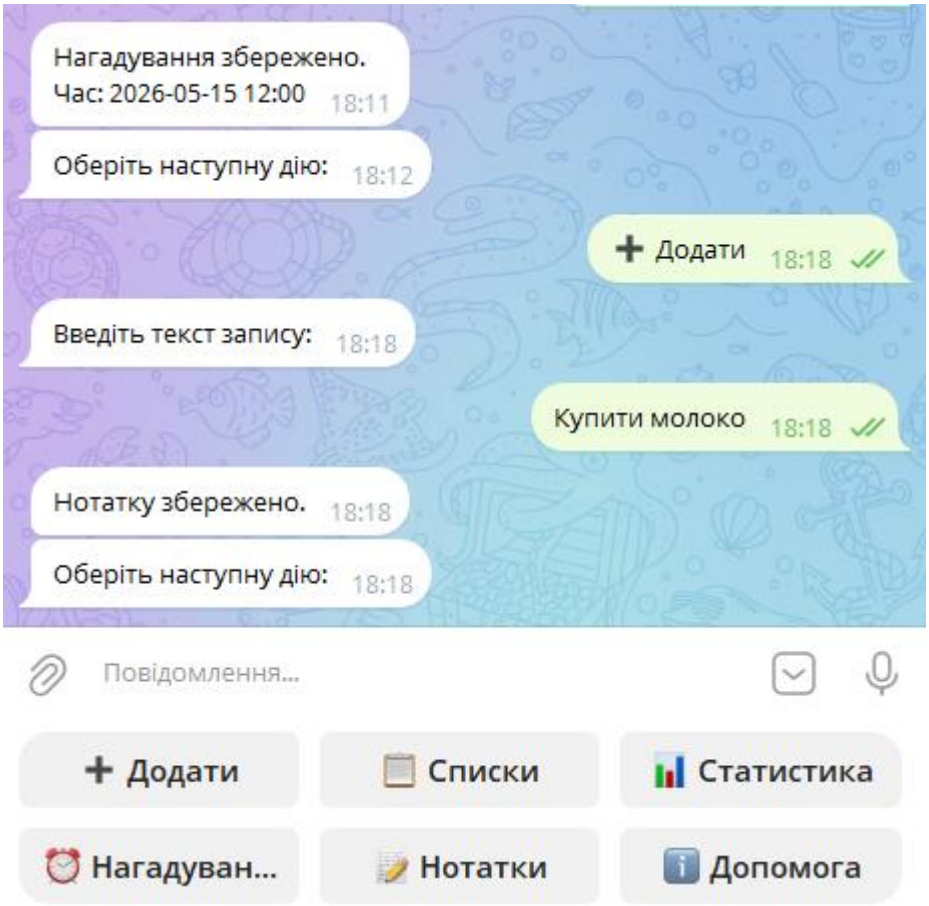


Рисунок 19 — *Збереження запису.*

При натисканні кнопки «Списки» користувач зможе обрати дію з inline-клавіатури (див. Рис. 20), після чого повідомлення зміниться на нове, на всіх етапах передбачено кнопки «Назад». Щоб переглянути вміст списку потрібно натиснути відповідну кнопку та обрати список (див. Рис. 21, 22). При створенні

списку потрібно ввести назву (див. Рис. 23), а при видаленні обрати відповідний список та підтвердити свою дію (див. Рис. 24, 25).

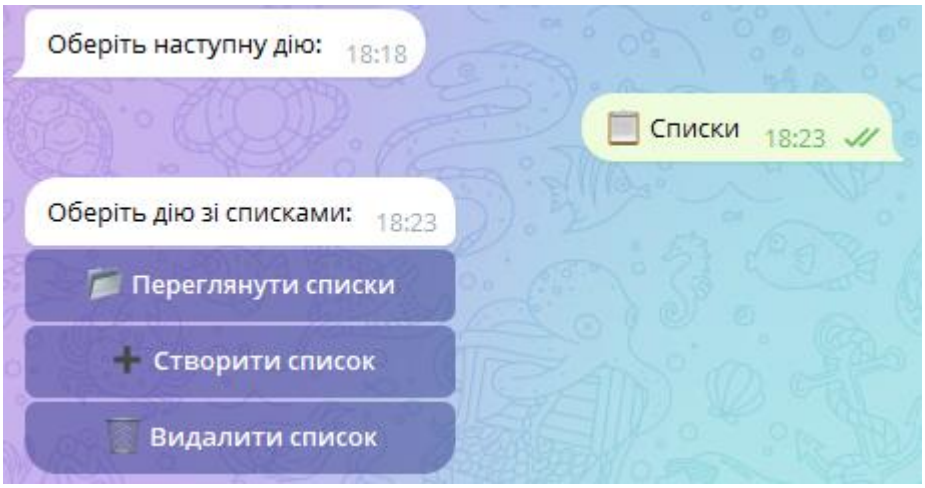


Рисунок 20 — *Кнопка «Списки».*

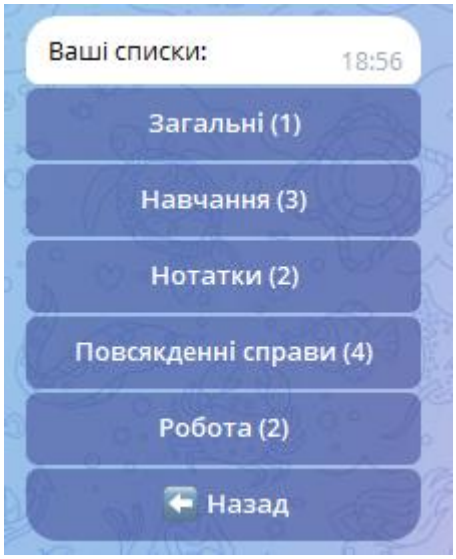


Рисунок 21 — *Перегляд списків.*

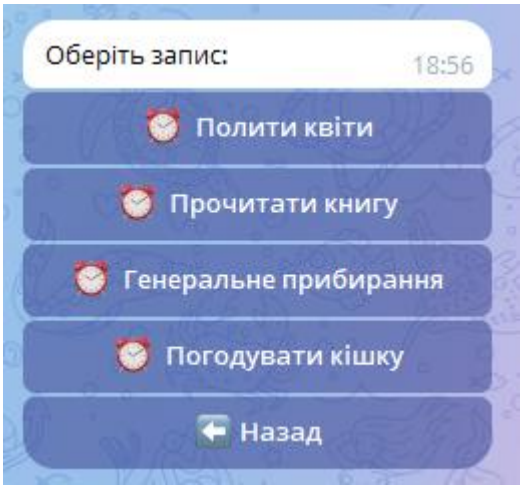


Рисунок 22 — *Перегляд вмісту списків.*



Рисунок 23 — *Створення нового списку.*

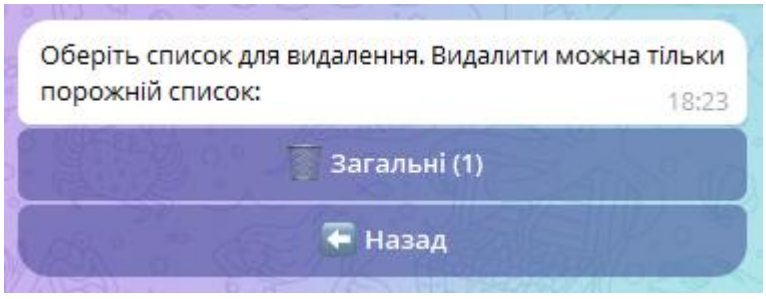


Рисунок 24 — *Вибір списку для видалення.*

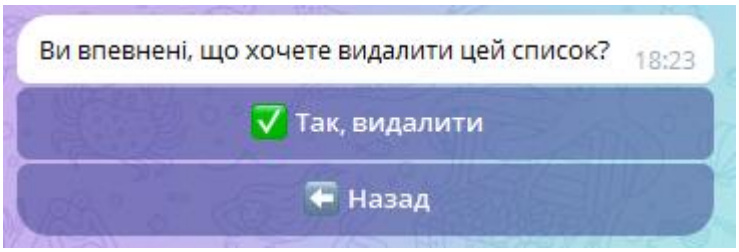


Рисунок 25 — *Підтвердження видалення.*

Після натискання кнопки «Статистика» користувач отримає повідомлення з певними показниками (див. Рис. 26).

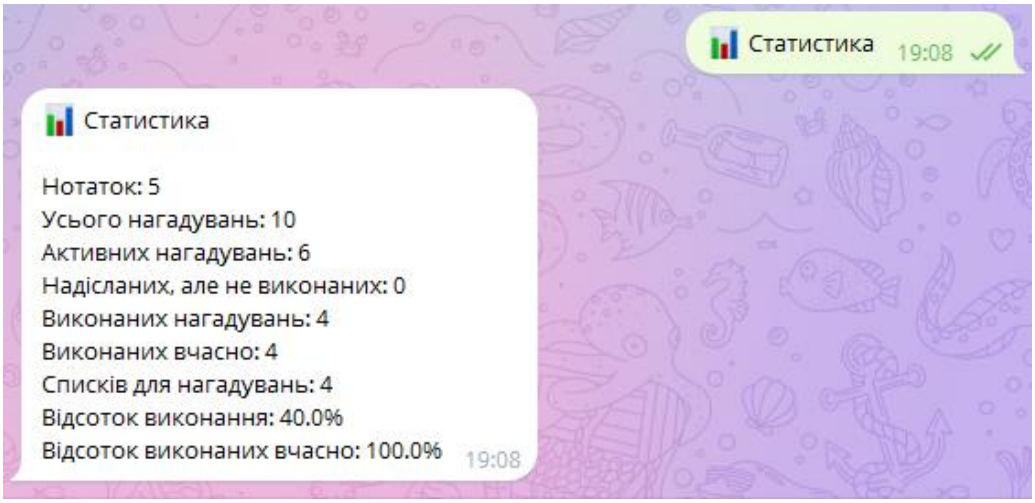


Рисунок 26 — *Кнопка «Статистика».*

При натисканні кнопок «Нагадування» або «Нотатки» буде представлено усі записи (див. Рис. 27, 29), які можна обрати для перегляду та редагування (див. Рис. 28, 30).

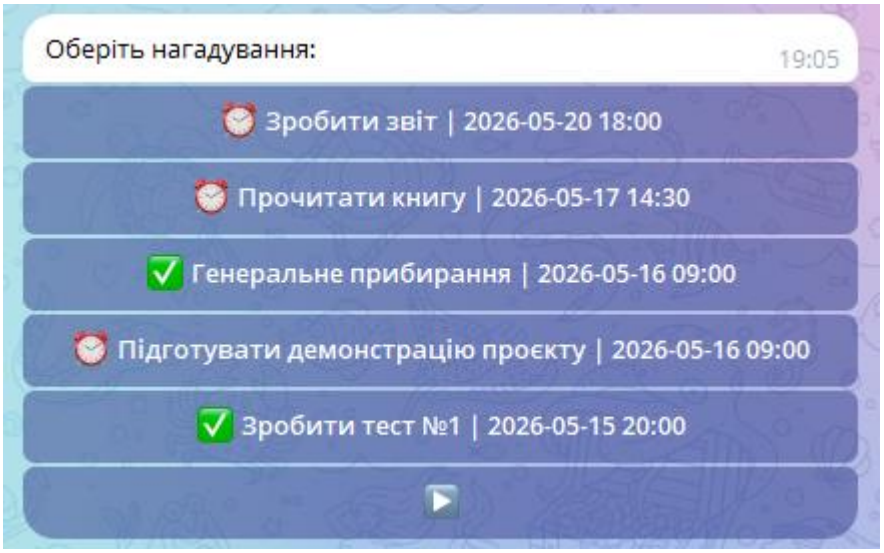


Рисунок 27 — *Кнопка «Нагадування».*

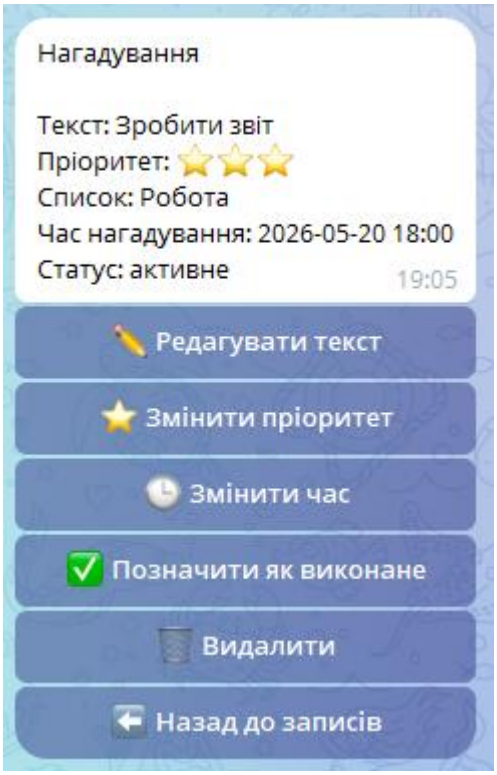


Рисунок 28 — *Редагування нагадування.*

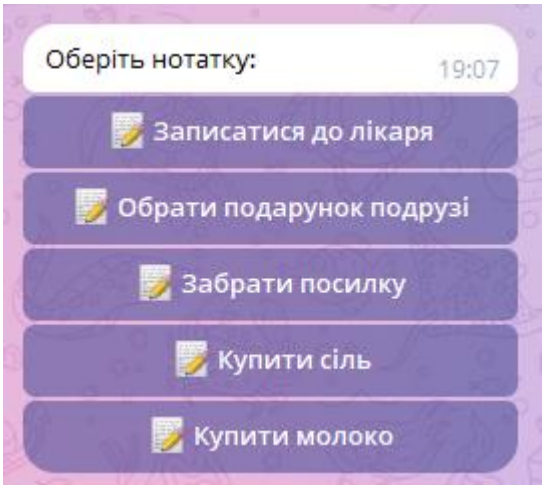


Рисунок 29 — *Кнопка «Нотатки».*

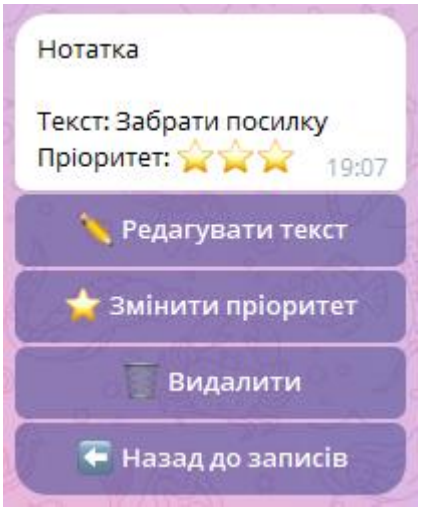


Рисунок 30 — *Редагування нотатки.*

Коли приходить повідомлення з нагадуванням, його можна відкласти або виконати (див. Рис. 31).

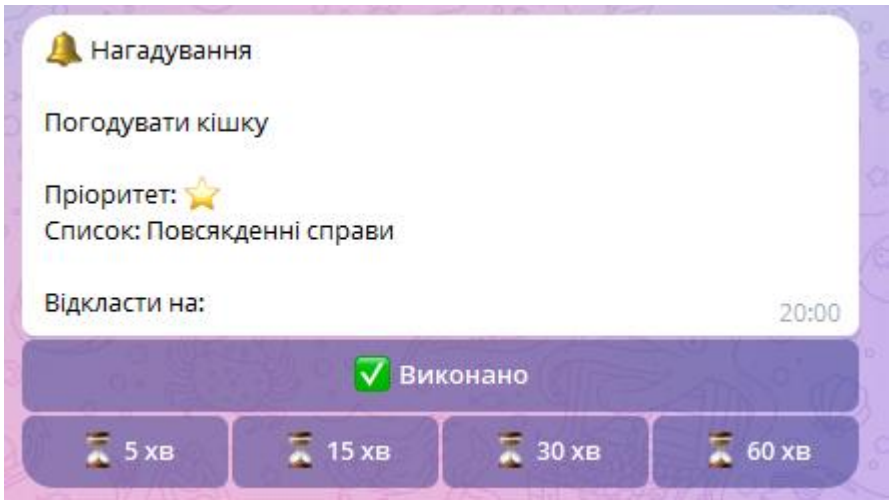


Рисунок 31 — *Нагадування.*

3.9 Реалізація та тестування

У межах дипломної роботи було розроблено Telegram-бота для організації власної активності користувача. Система реалізована мовою програмування Python із використанням бібліотеки aiogram. Для збереження даних використовується СКБД SQLite, доступ до якої здійснюється за

допомогою бібліотеки aiosqlite. Програмний код має модульну структуру та поділений на обробники, клавіатури, роботу з базою даних, сервіс нагадувань і допоміжні функції, що сприяє швидкому розумінню коду та архітектури.

Поточна версія Telegram-бота складається з Python-файлів, службових bat-файлів, файлу бази даних та допоміжних конфігураційних файлів. Загальна кількість основних Python-модулів системи становить 25. Загальна кількість Python-файлів у проєкті – 30, оскільки до цієї кількості входять також службові файли Python. Кількість рядків у Python-файлах становить 2378, а обсяг – 101,11 КБ.

Фізичні характеристики кожного модуля:

- database: 3 файли, 17,00 КБ, 436 рядків.
- handlers: 9 файлів, 55,64 КБ, 1234 рядки.
- keyboards: 7 файлів, 17,53 КБ, 439 рядків.
- services: 2 файли, 1,61 КБ, 33 рядки.
- utils: 7 файлів, 7,25 КБ, 185 рядків.
- Кореневі Python-файли: 2 файли, 2,08 КБ, 51 рядок.

Швидкодія системи перевірялася шляхом ручного виконання основних сценаріїв у Telegram. Під час тестування відповіді бота в більшості випадків надходили майже миттєво, максимальна зафіксована затримка становила приблизно 2 секунди, що є прийнятним для Telegram-бота, оскільки затримка може бути пов’язана не з кодом, а з якістю інтернет-з’єднання.

Під час роботи бота було зафіксовано два процеси Python, основний процес використовував 123,9 МБ оперативної пам’яті, другий – 3,32 МБ. Загальне споживання оперативної пам’яті становило приблизно 127,22 МБ.

Фактичне тестування Telegram-бота виконувалося на трьох Telegram-акаунтах, два з яких належать розробнику, ще один акаунт використовувався додатковим тестовим користувачем. Таким чином, кількість користувачів, окрім розробника, становила 1 особу.

Для перевірки працездатності Telegram-бота використовувалося ручне функціональне тестування методом чорної скриньки. Основна увага

приділялася перевірці поведінки системи з погляду користувача: натисканню кнопок, введенню тексту, створенню записів, отриманню нагадувань і перегляду статистики. У результаті тестування було підтверджено виконання основних функціональних вимог. Бот коректно реагує на команди, повідомлення та натискання кнопок, зберігає дані в базі, надсилає нагадування та формує статистику.

ВИСНОВКИ

В процесі виконання кваліфікаційної роботи бакалавра досягнуто наступні результати:

1. Проаналізовано предметну область завдяки роботі з літературними джерелами, підтверджено актуальність розробки, проведено огляд існуючих рішень («SkeddyBot», «Нагадування» та «Habitica»).

2. Проведено дослідження програмних засобів реалізації Telegram-ботів, обрано мову програмування Python та СКБД SQLite.

3. Спроєктовано клієнт-серверну архітектуру та модульну структуру Telegram-бота.

4. Сформовано вимоги до апаратного та програмного забезпечення, вимоги до користувачів, функціональні та нефункціональні вимоги.

5. Розроблено Telegram-бота для організації власної активності користувача. Застосунок дозволяє створювати, переглядати, редагувати та видаляти нотатки й нагадування, групувати записи за списками, встановлювати пріоритети, відкладати нагадування, позначати завдання як виконані та переглядати статистику активності користувача.

6. Проведенно тестування розробленого Telegram-бота методом ручного функціонального тестування, а саме методом «чорного ящика». Виявлені недоліки було виправлено.

7. Розроблений Telegram-бот може бути використаний як повноцінний застосунок для невеликих навантажень або як основа для комерційного проєкту шляхом масштабування, додавання нових функцій та монетизації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Басс Д.Є., Скрипник І.А. Розробка Telegram-бота для організації власної активності користувача. Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення (випуск 109): матеріали Міжнародної наукової інтернет-конференції, (м. Тернопіль, Україна, м. Ополе, Польща, 8-9 квітня 2026 р.), с.6-10. URL: http://www.konferenciaonline.org.ua/ua/article/id-2522/ (дата звернення: 10.05.2026).

2. TikTok чи Instagram? Яка соцмережа лідирує серед молоді в Україні? URL: https://life.kyiv.ua/news/tiktok-chi-instagram-yaka-socmerezha-lidiru-sered-molodi-v-ukrani (дата звернення: 29.04.2026).

3. Застосунки Telegram. URL: https://telegram.org/apps?setln=uk (дата звернення: 29.04.2026).

4. Хто створив Telegram: Біографія творців та історія месенджера. URL: https://www.karpaty.net.ua/hto-stvoryv-telegram-biografiya-tvorcziv-ta-istoriya-mesendzhera/ (дата звернення: 29.04.2026).

5. Telegram: історія створення месенджера, боти та блокування сервісу. URL: https://24tv.ua/tech/telegram-bot-shho-tse-istoriya-stvorennya-oglyad-na-telegram-kanali_n1343625 (дата звернення: 29.04.2026).

6. Telegram Bot Platform. URL: https://telegram.org/blog/bot-revolution (дата звернення: 29.04.2026).

7. Dealing with information overload: a comprehensive review. URL: https://www.frontiersin.org/journals/psychology/articles/10.3389/fpsyg.2023.1122200/full?utm_source (дата звернення: 29.04.2026).

8. Inferring mechanisms of response prioritization on social media under information overload. URL: https://www.nature.com/articles/s41598-020-79897-5?utm_source (дата звернення: 29.04.2026).

9. The global prevalence of attention deficit hyperactivity disorder in children and adolescents: An umbrella review of meta-analyses. URL: https://pubmed.ncbi.nlm.nih.gov/37495084/ (дата звернення: 29.04.2026).

10. ADHD Prevalence Among U.S. Children and Adolescents in 2022: Diagnosis, Severity, Co-Occurring Disorders, and Treatment. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC11334226/?utm_source (дата звернення: 29.04.2026).

11. Facts About ADHD Throughout the Years. URL: https://www.cdc.gov/adhd/data/adhd-throughout-the-years.html?utm_source (дата звернення: 29.04.2026).

12. The psychology behind task completion: a student's guide. URL: https://www.maketimecount.com/how-completion-helps-students-build-confidence/ (дата звернення: 29.04.2026).

13. Reminder bot for Telegram. URL: https://skeddy.me/ (дата звернення: 29.04.2026).

14. Як використовувати програму «Нагадування» на iPhone, iPad або iPod touch. URL: https://support.apple.com/uk-ua/102484 (дата звернення: 29.04.2026).

15. Habitica. URL: https://habitica.com/static/home (дата звернення: 29.04.2026).

16. Яка в Telegram мова програмування? URL: https://lemon.school/blog/yaka-u-telegram-mova-programuvanya (дата звернення: 29.04.2026).

17. SQLite Documentation. URL: https://www.sqlite.org/docs.html (дата звернення: 15.05.2026).

Декларація
академічної доброчесності
здобувача ступеня вищої освіти ЗНУ

Я, Басс Дарина Євгенівна, студентка 4 курсу, форми навчання заочної, Інженерного навчально-наукового інституту ім. Ю.М. Потебні ЗНУ, спеціальність 121 Інженерія програмного забезпечення освітньої програми "Програмне забезпечення систем", адреса електронної пошти se22ex-02@stu.zsea.edu.ua, — підтверджую, що написана мною кваліфікаційна робота на тему «**Розробка Telegram-бота для організації власної активності користувача**» відповідає вимогам академічної доброчесності та не містить порушень, що визначені у ст.42 Закону України «Про освіту», зі змістом яких ознайомлена.

- заявляю, що надана мною для перевірки електронна версія роботи є ідентичною її друкованій версії;

- згодна на перевірку моєї роботи на відповідність критеріям академічної доброчесності у будь-який спосіб, у тому числі за допомогою інтернет-системи, а також на архівування моєї роботи в базі даних цієї системи.

Дата 18.05.2026 _____ Басс Дарина Євгенівна
(підпис) (прізвище та ініціали) (студент)
Дата 18.05.2026 _____ Скрипник Ірина Анатоліївна
(підпис) (прізвище та ініціали) (керівник)