Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка Telegram-бота для формування самодисципліни в користувачів

Виконав: студент IV курсу, групи СН-42
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

(підпис) Пастух М.В.
(прізвище та ініціали)

Керівник (підпис) Готович В.А.
(прізвище та ініціали)

Нормоконтроль (підпис) Липак Г.І.
(прізвище та ініціали)

Завідувач кафедри (підпис) Боднарчук І.О.
(прізвище та ініціали)

Рецензент (підпис) Жаровський Р.О.
(прізвище та ініціали)

Тернопіль
2026

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«__» _____ 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп’ютерні науки
(шифр і назва спеціальності)

Студенту Пастуху Максиму Віталійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Розробка Telegram-бота для формування самодисципліни в користувачів

Керівник роботи Готович Володимир Анатолійович, кандидат технічних наук доцент кафедри комп'ютерних наук
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 14 » травня 2026 року № 4/9-239

2. Термін подання студентом завершеної роботи 22 червня 2026р.

3. Вихідні дані до роботи Літературні та інтернет-джерела з технологій розробки Telegram-ботів, Python, Telegram bot API, баз даних та вебтехнологій.

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Постановка задачі та формування вимог до програмного комплексу. 1.2. Аналіз предметної області. 1.2. Формування вимог до програмного комплексу. 1.3. Пошук актантів та варіантів використання. 1.4. Вибір середовища розробки. 1.5. Обґрунтування використовуваних технологій. 1.6. Висновок до першого розділу. 2. Проєктування, реалізація та тестування програмного комплексу. 2.1. Формування структури програмного комплексу. 2.2. Проєктування та створення БД для розроблюваного комплексу. 2.3. Проєктування інтерфейсу користувача. 2.4. Розробка програмного комплексу. 2.4.1. Розробка інтерфейсу. 2.4.2. Розробка функцій для обробки даних. 2.5. Тестування розробленого програмного комплексу . 2.6. Перспективи модернізації. 2.7. Висновок до другого розділу. 3. Безпека життєдіяльності, основи охорони праці. 3.1. Працездатність людини-оператора. 3.2. Загальні вимоги безпеки з охорони праці для користувачів ПК. 3.3. Висновок до третього розділу. Висновки. Перелік джерел. Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень, слайдів)
1. Титульна сторінка. 2. Мета кваліфікаційної роботи. 3. Актуальність обраної теми. 4. Аналіз предметної області. 5. Актори програмного комплексу. 6. Використані технології. 7. Структура програмного комплекс. 8. База даних. 9. Вигляд користувацької частини сайту. 10. Вигляд панелі адміністратора. 11. Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Гурик О. Я. кандидат технічних наук, доцент кафедри МТ		

7. Дата видачі завдання 27 січня 2026р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	27.01.2026	
2.	Підбір джерел про технології розробки Telegram-бота для формування самодисципліни в користувачів	27.01.2026-16.02.2026	
3.	Виконання дослідження щодо розробки програмного комплексу для складання розкладу занять	18.02.2026-17.05.2026	
	Розроблення функціоналу програмного комплексу		
4.	Оформлення розділу «Постановка задачі та формування вимог до програмного комплексу»	19.05.2026-22.05.2026	
5.	Оформлення розділу «Загальні підходи та методи розробки telegram-бота»	24.05.2026-28.05.2026	
6.	Оформлення розділу «Реалізація та тестування програмного комплексу»	28.05.2026-31.05.2026	
7.	Виконання завдання до розділу «Безпека життєдіяльності»	01.06.2026-06.06.2026	
8.	Виконання завдання до підрозділу «Основи охорони праці»	01.06.2026-06.06.2026	
9.	Оформлення кваліфікаційної роботи	06.06.2026-09.06.2026	
10.	Нормоконтроль		
11.	Перевірка на плагіат		
12.	Попередній захист кваліфікаційної роботи		
13.	Захист кваліфікаційної роботи	24.06.2026	

Студент _____ Пастух М.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Готович В.А.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка Telegram-бота для формування самодисципліни в користувачів// Кваліфікаційна робота освітнього рівня «Бакалавр» // Пастух Максим Віталійович// Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СН-42 // Тернопіль, 2026 // С. 65, рис. – 17, табл. – 1, додат. – 3, бібліогр. – 35.

Ключові слова: Telegram-бот, програмний комплекс, самодисципліна, планування завдань, нагадування, продуктивність, JavaScript, Node.js, MongoDB, Telegraf

Кваліфікаційна робота присвячена розробці Telegram-бота для формування самодисципліни у користувачів. У роботі досліджено сучасні підходи до планування особистого часу та автоматизації керування повсякденними завданнями за допомогою інформаційних технологій.

У першому розділі проведено аналіз предметної області, сформовано вимоги до програмного комплексу, визначено основних користувачів та сценарії їх взаємодії із системою.

У другому розділі спроєктовано структуру програмного комплексу, базу даних, реалізовано основні функції Telegram-бота та алгоритми його роботи.

У третьому розділі описано реалізацію програмних модулів, взаємодію з базою даних, проведено тестування програмного комплексу та визначено перспективи його подальшої модернізації. У четвертому розділі розглянуто питання безпеки життєдіяльності та охорони праці під час роботи з персональним комп'ютером.

Об'єкт дослідження: процес формування самодисципліни та організації особистого часу користувачів із використанням сучасних інформаційних технологій.

Предмет дослідження: методи, засоби та програмне забезпечення для розробки Telegram-бота, призначеного для планування завдань, автоматичного нагадування про їх виконання та контролю особистої продуктивності.

ANNOTATION

Development of a Telegram Bot for Building User Self-Discipline // Bachelor's Qualification Thesis // Maksym V. Pastukh // Ivan Puluj Ternopil National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Computer Science, Group CH-42 // Ternopil, 2026 // 48 pages, 15 figures, 2 appendices, 25 references.

Keywords: Telegram bot, software system, self-discipline, task planning, reminders, productivity, JavaScript, Node.js, MongoDB, Telegraf.

The qualification thesis is devoted to the development of a Telegram bot for improving users' self-discipline. The research focuses on modern approaches to personal time management and the automation of everyday task planning using information technologies.

The first chapter presents the analysis of the subject area, defines the requirements for the software system, and identifies the main users and use cases.

The second chapter describes the design of the software architecture and database, as well as the implementation of the main Telegram bot functions and its operating algorithms.

The third chapter covers the implementation of the software modules, interaction with the database, testing of the developed system, and prospects for its further modernization. The fourth chapter addresses occupational safety and health issues related to working with personal computers.

Object of research – the process of developing self-discipline and organizing personal time through the use of modern information technologies.

Subject of research – methods, tools, and software for developing a Telegram bot intended for task planning, automatic reminders, and monitoring users' personal productivity.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API (англ. Application Programming Interface) – програмний інтерфейс застосунку, який забезпечує взаємодію між різними програмними системами.

Bot API (англ. Bot Application Programming Interface) – програмний інтерфейс Telegram, що використовується для створення та керування чат-ботами.

CSS (англ. Cascading Style Sheets) – мова стилів, що використовується для опису зовнішнього вигляду вебсторінок.

CRUD (англ. Create, Read, Update, Delete) – базові операції створення, читання, оновлення та видалення даних.

HTML (англ. HyperText Markup Language) – стандартизована мова розмітки документів для створення вебсторінок.

JavaScript (JS) – високорівнева мова програмування, що використовується для розробки вебзастосунків і серверної логіки.

JSON (англ. JavaScript Object Notation) – текстовий формат обміну даними, який використовується для передачі інформації між клієнтом і сервером.

MongoDB – документоорієнтована система керування базами даних класу NoSQL.

Node.js – програмна платформа для виконання JavaScript-коду на серверній стороні.

npm (англ. Node Package Manager) – менеджер пакетів для середовища Node.js.

NoSQL (англ. Not Only SQL) – клас нереляційних баз даних, що використовуються для зберігання неструктурованих або напівструктурованих даних.

REST API (англ. Representational State Transfer Application Programming Interface) – архітектурний стиль взаємодії програмних систем через HTTP-запити.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному світі людина щодня стикається з великою кількістю інформації, завдань та відволікаючих факторів. Через це багатьом важко правильно організувати свій час, дотримуватись режиму та регулярно виконувати поставлені цілі. Недостатній рівень самодисципліни негативно впливає на навчання, роботу та особисту ефективність. Саме тому все більшої популярності набувають цифрові інструменти, які допомагають людям контролювати власні справи, формувати корисні звички та підтримувати мотивацію.

Одним із найзручніших засобів для взаємодії з користувачами сьогодні є месенджери. Особливу популярність має Telegram завдяки простому інтерфейсу, швидкості роботи та можливості створення ботів із різноманітним функціоналом. Telegram-боти дозволяють автоматизувати багато повсякденних процесів: надсилати нагадування, вести облік виконаних завдань, зберігати статистику та допомагати користувачам дотримуватись власного плану дня.

Розробка Telegram-бота для формування самодисципліни є актуальною, оскільки такий програмний продукт може стати зручним помічником для користувачів у навчанні, роботі та повсякденному житті. Бот може допомагати створювати список завдань, нагадувати про важливі справи, відстежувати виконання поставлених цілей та мотивувати користувача до регулярної роботи над собою.

Крім того, використання Telegram-бота не потребує встановлення окремого мобільного застосунку, що робить систему більш доступною та зручною для широкого кола користувачів. Завдяки цьому користувач може швидко отримати доступ до функціоналу бота у будь-який момент.

Мета і задачі дослідження. Метою даної кваліфікаційної роботи освітнього рівня «Бакалавр» є розробка Telegram-бота для формування самодисципліни у користувачів.

Для досягнення поставленої мети необхідно виконати такі **задачі**:

– проаналізувати предметну область;

- дослідити сучасні технології створення Telegram-ботів;
- визначити основні функціональні можливості системи
- спроєктувати структуру програмного комплексу;
- розробити базу даних для зберігання інформації;
- реалізувати функціонал створення та управління завданнями;
- реалізувати систему нагадувань;
- створити модуль статистики та відстеження прогресу користувача;
- розробити зручний інтерфейс взаємодії з ботом;
- провести тестування програмного продукту.

Практичне значення одержаних результатів. Результатом виконання кваліфікаційної роботи є Telegram-бот для формування самодисципліни, який дозволяє користувачам ефективніше організовувати свій час та контролювати виконання поставлених завдань. Система забезпечує можливість створення списків справ, отримання нагадувань, перегляду статистики активності та відстеження власного прогресу.

Розроблений програмний продукт може бути корисним для студентів, працівників та всіх користувачів, які прагнуть покращити власну продуктивність і навички самоорганізації. Використання бота дозволяє спростити процес планування повсякденних справ та підвищити ефективність виконання завдань.

.

РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ ТА ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОГО КОМПЛЕКСУ

1.1 Аналіз предметної області

Сучасний розвиток інформаційних технологій та електронної комерції суттєво вплинув на сферу продажу товарів і надання послуг. Одним із найбільш популярних напрямів стала організація онлайн замовлення та доставки харчових продуктів через вебплатформи та мобільні застосунки. Такий підхід дозволяє користувачам швидко оформлювати замовлення, переглядати асортимент продукції, оплачувати покупки онлайн та отримувати доставку у зручне місце й час.

Предметною областю даного дослідження є процес автоматизації замовлення та доставки харчових продуктів із використанням вебтехнологій. Основною метою таких систем є забезпечення ефективної взаємодії між клієнтом, адміністратором і службою доставки. Вебплатформа повинна забезпечувати зручний інтерфейс, високу швидкість роботи, безпечне зберігання даних та можливість обробки великої кількості замовлень.

На сьогодні існує значна кількість сервісів онлайн доставки продуктів та готової їжі, серед яких Glovo, Bolt Foodта Uber Eats. Дані системи надають користувачам можливість швидкого пошуку товарів, оформлення замовлень та відстеження доставки в режимі реального часу. Проте багато існуючих платформ мають недоліки, серед яких складність адміністрування, перевантажений інтерфейс, високе навантаження на систему або недостатня адаптація під потреби окремих закладів чи магазинів.

Таким чином, аналіз предметної області показує, що розробка вебплатформи для онлайн замовлення та доставки харчових продуктів є актуальним та перспективним напрямом. Створення такої системи дозволить автоматизувати процеси продажу, покращити якість обслуговування клієнтів та забезпечити ефективне управління замовленнями [1].

1.2 Формування вимог до Telegram-бота

Під час розробки Telegram-бота для формування самодисципліни важливо чітко визначити, що саме повинна вміти система та як вона має поводитися в різних ситуаціях. Це дозволяє ще на етапі проєктування уникнути помилок і зробити бот зрозумілим та зручним для користувача.

Основна ідея цього проєкту полягає в тому, щоб користувач міг організовувати свої щоденні завдання без складних застосунків — прямо в Telegram. Тому головна вимога до системи — простота взаємодії та швидкий доступ до всіх функцій.

Функціональні вимоги описують те, що саме повинен уміти бот. Насамперед користувач повинен мати можливість легко почати роботу з ботом без складної реєстрації — достатньо просто запустити його в Telegram. Після цього система повинна “запам’ятати” користувача та надалі працювати з його даними.

Користувач повинен мати змогу створювати нові завдання, вказувати їх назву, опис і, за потреби, час або дату виконання. Важливо, щоб завдання можна було не лише додавати, але й редагувати або видаляти, якщо вони втратили актуальність.

Окремою важливою функцією є система нагадувань. Бот повинен автоматично надсилати повідомлення користувачу у визначений час, щоб допомогти не забувати про заплановані справи. Саме ця функція є однією з ключових, оскільки вона напряму впливає на формування дисципліни.

Також користувач повинен мати можливість відмічати виконані завдання. На основі цього бот може вести просту статистику: скільки завдань виконано, скільки пропущено, та як змінюється активність користувача з часом. Це дозволяє людині бачити власний прогрес, що додатково мотивує.

Нефункціональні вимоги стосуються того, як система повинна працювати. У першу чергу бот має бути максимально простим у використанні, оскільки він розрахований на звичайних користувачів без технічних знань. Уся взаємодія повинна відбуватись через зрозумілі команди або кнопки в Telegram.

Також важливо, щоб бот працював швидко і не “зависав” при обробці запитів. Навіть при великій кількості користувачів система повинна стабільно виконувати свої функції без втрати даних.

Окремо варто відзначити вимогу до надійності збереження інформації. Усі завдання та дані користувача повинні зберігатися так, щоб вони не втрачались при перезапуску бота або сервера.

Крім цього, система повинна бути гнучкою для подальшого розвитку. У майбутньому можна буде додати нові функції, наприклад систему досягнень, рівнів користувача або більш складну аналітику продуктивності.

Таким чином, сформовані вимоги дозволяють створити зрозумілий, зручний і корисний Telegram-бот, який реально допомагає користувачам організовувати свій час і поступово формувати навички самодисципліни.

1.3 Пошук актантів та варіантів використання

Під час проєктування будь-якого програмного комплексу важливо визначити, хто саме буде взаємодіяти із системою та які функції виконуватиме кожен учасник. Такий аналіз дозволяє правильно сформувати функціональні можливості програмного продукту та забезпечити зручну взаємодію користувачів із системою.

У розробленому Telegram-боті можна виділити декілька акторів, які беруть участь у роботі програмного комплексу. До них належать користувач, адміністратор, Telegram Bot API та система сповіщень. Кожен із них виконує власні функції та взаємодіє із системою відповідно до визначених прав і можливостей [2-4].

На рисунку 1.1 наведено основних акторів програмного комплексу та їхню роль у роботі Telegram-бота.

Головним актором системи є користувач. Саме він безпосередньо взаємодіє з Telegram-ботом та використовує його функціональні можливості для організації власного часу. Користувач може створювати нові завдання, переглядати їх список, редагувати або видаляти їх, встановлювати нагадування,

позначати виконані справи та переглядати статистику власної активності. Основною метою використання бота є підвищення рівня самоорганізації та формування звички регулярно виконувати заплановані завдання.

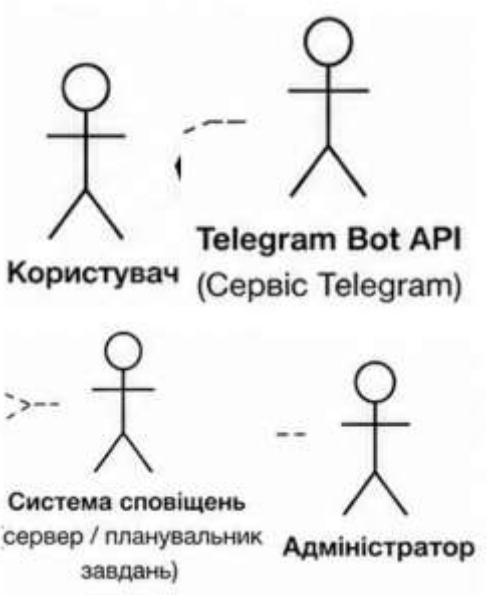


Рисунок1.1 – Основні актори програмного комплексу

Другим важливим актором є адміністратор. Його функції пов'язані із забезпеченням стабільної роботи програмного комплексу та його підтримкою. Адміністратор здійснює контроль роботи Telegram-бота, переглядає журнали помилок, контролює працездатність системи, оновлює програмне забезпечення та за необхідності змінює налаштування роботи окремих модулів. На відміну від звичайного користувача, адміністратор не використовує бот для планування власних справ, а відповідає за технічне обслуговування та розвиток системи.

Ще одним учасником роботи програмного комплексу є Telegram Bot API, який забезпечує обмін повідомленнями між Telegram та серверною частиною застосунку. Саме через цей сервіс користувач отримує відповіді бота, меню, повідомлення та іншу інформацію.

Крім цього, у роботі системи використовується система сповіщень, яка відповідає за автоматичне надсилання нагадувань користувачам у визначений час. Її використання дозволяє своєчасно повідомляти про заплановані справи та підтримувати регулярність виконання завдань.

Після визначення основних акторів доцільно проаналізувати їх взаємодію із системою. Для цього будується діаграма варіантів використання (Use Case Diagram), яка відображає всі основні сценарії роботи Telegram-бота.

На рисунку 1.2 представлено діаграму варіантів використання, що демонструє взаємодію користувачів та інших акторів із програмним комплексом.

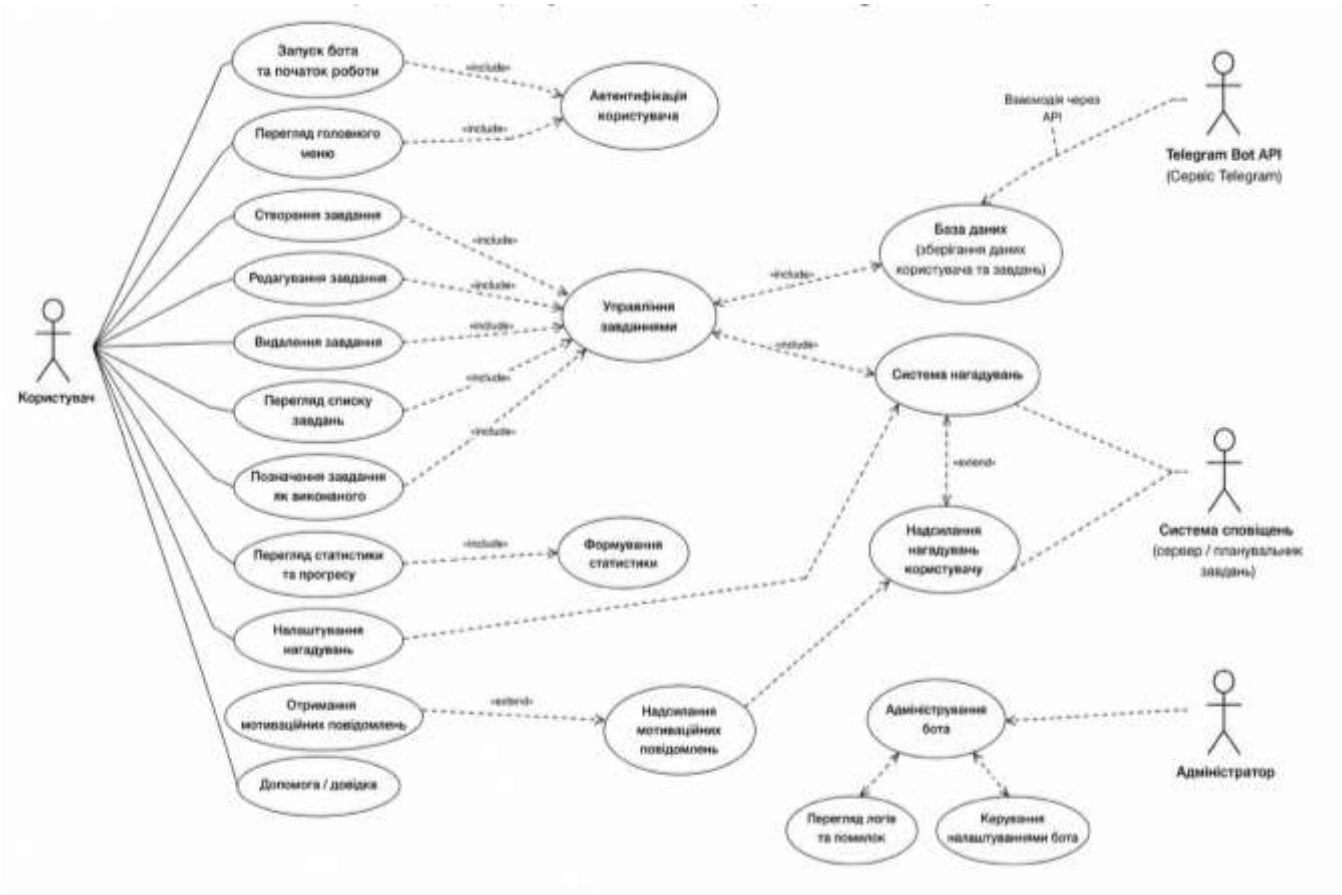


Рисунок 1.2 – Діаграма варіантів використання Telegram-бота для формування самодисципліни

З наведеної діаграми видно, що після запуску бота користувач отримує доступ до головного меню та може скористатися основними функціями системи. До них належать створення нового завдання, його редагування або видалення, перегляд списку завдань, налаштування нагадувань, позначення виконаних справ як виконаних та перегляд статистики продуктивності.

Однією з найважливіших функцій Telegram-бота є система автоматичних нагадувань. Після створення завдання інформація зберігається у базі даних, а

система сповіщень у зазначений час формує повідомлення та надсилає його користувачу через Telegram Bot API. Завдяки цьому користувач не пропускає важливі події та поступово виробляє звичку виконувати заплановані справи вчасно.

Після завершення виконання завдання користувач може позначити його як виконане. На основі накопичених даних система формує статистику, яка відображає кількість виконаних завдань, активність користувача та його прогрес за певний проміжок часу [5-6].

Таким чином, аналіз акторів та побудова діаграми варіантів використання дозволяють визначити основні сценарії взаємодії із Telegram-ботом та є важливим етапом проєктування програмного комплексу. Це забезпечує чітке розуміння структури системи та спрощує подальшу реалізацію її функціональних можливостей.

1.4 Вибір середовища розробки

Вибір середовища розробки є одним із важливих етапів створення будь-якого програмного продукту. Від правильно підібраних інструментів залежить швидкість розробки, якість програмного коду, зручність тестування та подальша підтримка проєкту.

Для розробки Telegram-бота було обрано сучасне інтегроване середовище розробки Visual Studio Code. Воно є безкоштовним, має простий та зрозумілий інтерфейс, підтримує велику кількість мов програмування та дозволяє встановлювати додаткові розширення для підвищення ефективності роботи.

Однією з головних переваг Visual Studio Code є підтримка технології IntelliSense, яка автоматично пропонує варіанти завершення коду, допомагає швидше знаходити помилки та значно прискорює процес програмування. Крім цього, середовище підтримує інтеграцію з системою контролю версій Git, що дозволяє зберігати історію змін та працювати над проєктом у команді.

Для написання серверної частини Telegram-бота використовувалася платформа Node.js, яка забезпечує швидке виконання JavaScript-коду на сервері

та дозволяє створювати масштабовані мережеві застосунки. Використання Node.js значно спрощує інтеграцію з Telegram Bot API та роботу із зовнішніми сервісами.

Як менеджер пакетів було використано npm, за допомогою якого встановлювалися всі необхідні бібліотеки та залежності проєкту. Це дозволило швидко додавати новий функціонал та підтримувати актуальні версії програмних модулів.

Для тестування роботи Telegram-бота використовувався офіційний застосунок Telegram, через який перевірялася коректність виконання команд, надсилання повідомлень, створення завдань та робота системи нагадувань.

Зберігання програмного коду здійснювалося із використанням системи контролю версій Git, а також віддаленого репозиторію GitHub. Це дозволяє зберігати резервні копії проєкту, контролювати внесені зміни та забезпечує можливість подальшого розвитку програмного комплексу.

Обрані інструменти повністю відповідають сучасним вимогам до створення серверних застосунків та забезпечують зручність написання, тестування й підтримки програмного коду. Їх використання дозволило реалізувати всі необхідні функції Telegram-бота та створити стабільний програмний продукт.

Отже, вибране середовище розробки та програмні засоби є оптимальними для реалізації Telegram-бота, забезпечують високу продуктивність роботи розробника та створюють умови для подальшої

1.5 Обґрунтування вибору використаних технологій

Під час розробки Telegram-бота для формування самодисципліни особливу увагу було приділено вибору технологій, оскільки саме вони визначають продуктивність, швидкодію, надійність та можливість подальшого розвитку програмного комплексу. Обрані програмні засоби є сучасними, широко використовуються у веброзробці та дозволяють реалізувати необхідний функціонал із мінімальними витратами часу.

Основною мовою програмування для реалізації серверної частини було обрано JavaScript. Ця мова є однією з найпоширеніших у сфері веброзробки, має простий синтаксис та підтримує асинхронне виконання операцій, що особливо важливо при обробці великої кількості запитів від користувачів Telegram-бота.

Для виконання JavaScript-коду на сервері використовується платформа Node.js. Її головною перевагою є висока швидкість роботи та можливість одночасно обробляти багато запитів без значного навантаження на систему. Це робить Node.js одним із найкращих рішень для створення чат-ботів та серверних застосунків.

Для взаємодії із сервісом Telegram використовується Telegram Bot API, який надає програмний інтерфейс для отримання повідомлень від користувачів та надсилання відповідей. Завдяки цьому бот може реагувати на команди, обробляти введені дані, надсилати повідомлення, кнопки та нагадування.

Для роботи з Telegram Bot API доцільно використовувати бібліотеку Telegraf, яка значно спрощує процес створення бота. Вона містить готові інструменти для обробки команд, повідомлень, callback-кнопок та інших можливостей Telegram, що дозволяє скоротити час розробки та зробити програмний код більш зрозумілим.[7]

Для зберігання інформації про користувачів, завдання та нагадування використовується база даних MongoDB. Вона належить до нереляційних баз даних і дозволяє зберігати інформацію у вигляді документів, що є зручним для роботи з динамічними структурами даних. MongoDB забезпечує швидкий доступ до інформації, добре масштабується та широко застосовується при створенні сучасних вебзастосунків.

Для роботи з базою даних використовується бібліотека Mongoose, яка спрощує створення моделей даних, їх валідацію та виконання запитів до бази. Використання цього інструмента робить код більш структурованим і полегшує подальшу підтримку програмного комплексу.

Для керування залежностями проєкту використовується npm (Node Package Manager). Він дозволяє швидко встановлювати необхідні бібліотеки, оновлювати їх та підтримувати стабільну роботу програмного комплексу.

Під час розробки також використовувалася система контролю версій Git, яка забезпечує збереження історії змін програмного коду та дозволяє відновити попередні версії проєкту у разі виникнення помилок. Для віддаленого зберігання репозиторію застосовується сервіс GitHub, що спрощує резервне копіювання та спільну роботу над проєктом.

Обраний стек технологій забезпечує стабільну та швидку роботу програмного комплексу, дозволяє легко реалізувати необхідний функціонал і створює можливість для подальшого вдосконалення системи. Використання сучасних інструментів розробки також сприяє підвищенню якості програмного коду та спрощує його супровід.[8-10]

Таким чином, застосування JavaScript, Node.js, Telegram Bot API, Telegraf, MongoDB, Mongoose, npm та Git є обґрунтованим рішенням для створення Telegram-бота, оскільки ці технології повністю відповідають вимогам проєкту та забезпечують ефективну реалізацію поставлених

1.6 Висновок до першого розділу

У першому розділі кваліфікаційної роботи було проведено аналіз предметної області та визначено основні аспекти розробки Telegram-бота для формування самодисципліни у користувачів. Дослідження показало, що проблема ефективного планування часу та організації повсякденної діяльності є актуальною, а використання сучасних цифрових технологій може значно полегшити процес формування корисних звичок і підвищення особистої продуктивності.

У процесі виконання розділу було сформовано основні вимоги до програмного комплексу, визначено його функціональні можливості та описано основні принципи роботи. Особливу увагу приділено функціям створення завдань, налаштування нагадувань, ведення статистики виконання та відстеження особистого прогресу користувача.

Також було проаналізовано основних акторів системи та їх взаємодію з Telegram-ботом. Побудова діаграми варіантів використання дозволила

визначити основні сценарії роботи програмного комплексу та сформувати загальне уявлення про його структуру і функціонування. Крім цього, було обрано середовище розробки та обґрунтовано використання сучасних технологій і програмних засобів, серед яких JavaScript, Node.js, Telegram Bot API, Telegraf, MongoDB та Git. Використання цього технологічного стеку забезпечує стабільну роботу системи, швидку обробку запитів користувачів і можливість подальшого розширення функціоналу.

Отримані результати першого розділу створюють теоретичну основу для наступного етапу дослідження, який присвячений проєктуванню архітектури програмного комплексу, створенню бази даних, реалізації Telegram-бота та перевірці його працездатності. Це дозволить розробити сучасний програмний продукт, який сприятиме розвитку самодисципліни та ефективнішій організації особистого часу користувачів.

РОЗДІЛ 2. ЗАГАЛЬНІ ПІДХОДИ ТА МЕТОДИ РОЗРОБКИ TELEGRAM-БОТА

2.1 Формування структури програмного комплексу

Після аналізу предметної області та визначення основних вимог до системи наступним етапом розробки є формування структури програмного комплексу. Від правильно спроєктованої архітектури залежить стабільність роботи Telegram-бота, зручність його підтримки та можливість подальшого розширення функціональних можливостей.

Розроблений програмний комплекс побудований за модульним принципом, що дозволяє розділити функціонал на окремі незалежні компоненти. Такий підхід спрощує внесення змін у програмний код, підвищує його читабельність та полегшує процес тестування окремих частин системи.

Архітектура Telegram-бота складається з декількох взаємопов'язаних модулів. Центральним елементом є серверна частина, яка обробляє запити користувачів, взаємодіє з Telegram Bot API та виконує необхідну бізнес-логіку.

Після надсилання користувачем повідомлення воно передається через Telegram Bot API на сервер, де обробляється відповідним модулем. Залежно від отриманої команди система виконує необхідну операцію: створення нового завдання, перегляд списку справ, встановлення нагадування, позначення завдання як виконаного або відображення статистики.

Для зберігання інформації використовується база даних, у якій містяться відомості про користувачів, створені ними завдання, дати виконання та історія активності. Це забезпечує можливість збереження інформації навіть після завершення роботи користувача з ботом.

Окремим модулем є система нагадувань, яка автоматично перевіряє час виконання запланованих завдань та у визначений момент надсилає повідомлення користувачу. Завдяки цьому бот виконує свою головну функцію – допомагає підтримувати дисципліну та не забувати про важливі справи [11].

Також у структурі програмного комплексу передбачено модуль формування статистики. Він аналізує інформацію про виконані та невиконані завдання, після чого формує звіт про активність користувача та його прогрес.

На рисунку 2.1 наведено загальну структуру розробленого програмного комплексу.

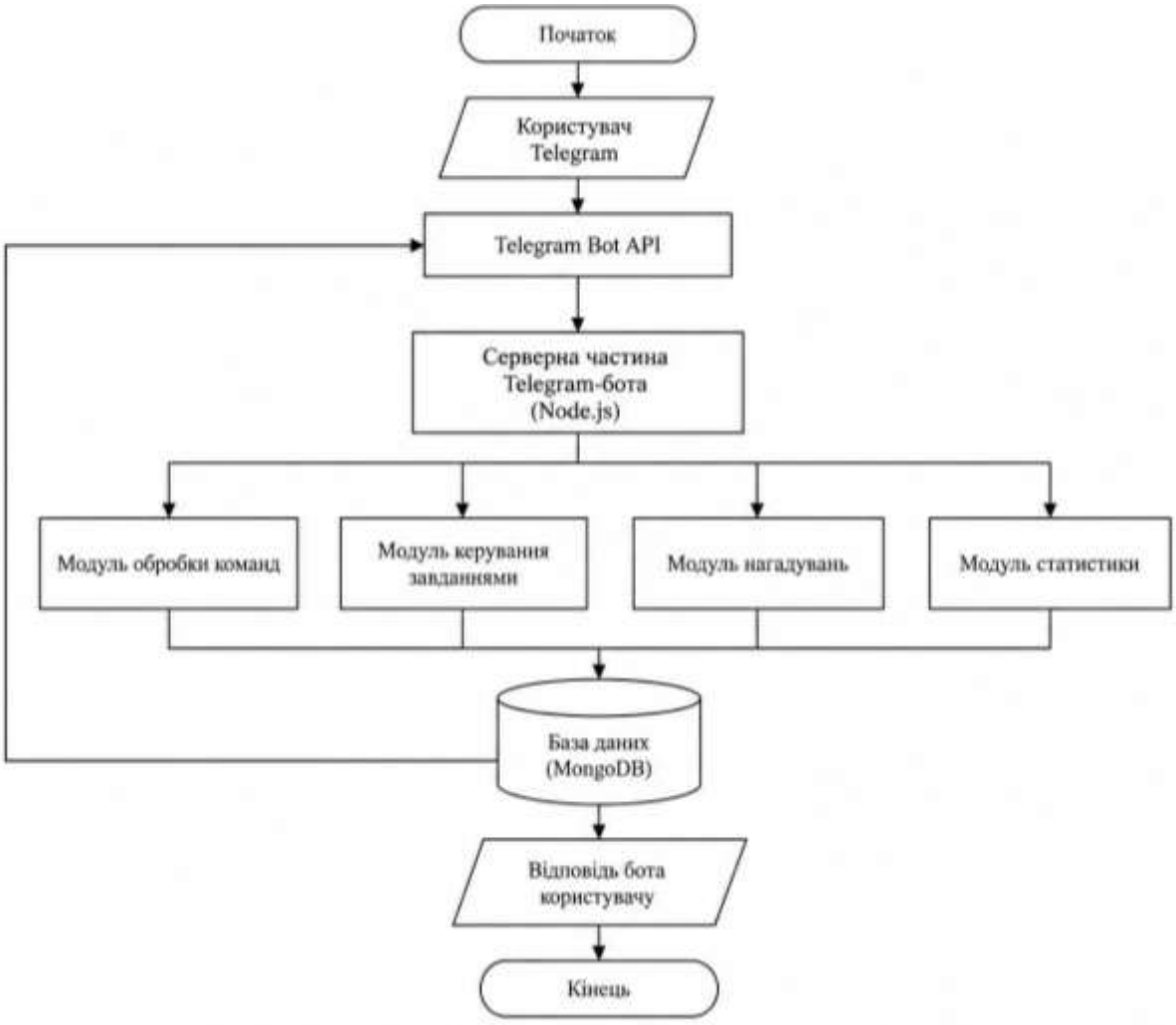


Рисунок 2.1 – Структура Telegram-бота для формування самодисципліни

Структура програмного комплексу включає такі основні компоненти:

– користувач Telegram;

– Telegram Bot API;

– серверну частину програмного комплексу;

– модуль обробки команд;

– модуль керування завданнями;

– модуль нагадувань;

– модуль статистики;

– базу даних.

Усі модулі взаємодіють між собою та забезпечують повноцінну роботу системи. Така архітектура дозволяє легко модернізувати окремі компоненти без внесення суттєвих змін до інших частин програмного комплексу.

Використання модульної структури також створює можливість для подальшого розширення функціоналу. У майбутньому до системи можна додати нові можливості, зокрема систему досягнень, рейтинги користувачів, інтеграцію з календарями або використання штучного інтелекту для формування персоналізованих рекомендацій.

Отже, сформована структура програмного комплексу є логічною, зрозумілою та відповідає поставленим вимогам. Вона забезпечує ефективну взаємодію всіх компонентів системи та створює надійну основу для реалізації Telegram-бота для формування самодисципліни.

2.2 Проєктування архітектури програмного комплексу

Архітектура програмного комплексу є одним із найважливіших етапів розробки Telegram-бота, оскільки саме вона визначає взаємодію між усіма компонентами системи, порядок обробки запитів користувачів та механізми зберігання інформації. Грамотно спроєктована архітектура забезпечує стабільну роботу програмного комплексу, можливість його подальшого розширення та підтримки.

Розроблений Telegram-бот побудований за клієнт-серверною архітектурою. У ролі клієнта виступає застосунок Telegram, через який користувач взаємодіє із системою. Серверна частина відповідає за приймання повідомлень, їх обробку, виконання необхідних бізнес-процесів та формування відповіді користувачу.

Основними компонентами архітектури є Telegram API, сервер застосунку, база даних та користувач. Користувач надсилає повідомлення або виконує певну

команду в Telegram, після чого запит через Telegram Bot API передається на сервер. Сервер аналізує отримані дані, виконує необхідну логіку роботи, звертається до бази даних у разі потреби та формує відповідь, яка повертається користувачу через Telegram.

Важливим елементом архітектури є серверна частина, реалізована на платформі Node.js. Вона забезпечує швидку обробку великої кількості одночасних запитів, підтримує асинхронну модель роботи та дозволяє ефективно взаємодіяти із зовнішніми сервісами й базою даних.

Для зберігання інформації використовується база даних, у якій містяться дані про користувачів, історію звернень, налаштування та інші службові відомості. Централізоване зберігання інформації забезпечує швидкий доступ до необхідних даних та спрощує адміністрування програмного комплексу.

Архітектура програмного комплексу побудована за модульним принципом. Кожен функціональний модуль відповідає за окремий напрям роботи: обробку команд, взаємодію з базою даних, авторизацію користувачів, формування повідомлень та адміністрування системи. Такий підхід значно спрощує супровід програмного забезпечення та дозволяє додавати нові функції без зміни вже реалізованих компонентів.

Для детального відображення структури програмного комплексу та взаємозв'язків між його основними компонентами було розроблено діаграму класів. Вона демонструє основні сутності системи, їх атрибути, методи та взаємозв'язки між окремими модулями Telegram-бота. Діаграму класів наведено на рисунку 2.2.

Взаємодія між модулями здійснюється через внутрішні програмні інтерфейси, що забезпечує незалежність окремих компонентів та підвищує надійність роботи системи. У випадку модернізації одного з модулів інші компоненти програмного комплексу можуть працювати без змін.

Схему архітектури Telegram-бота наведено у Додатку А, де відображено взаємодію між користувачем, Telegram Bot API, серверною частиною та базою даних. Запропонована архітектура забезпечує високу продуктивність, масштабованість і можливість подальшого розвитку програмного комплексу.

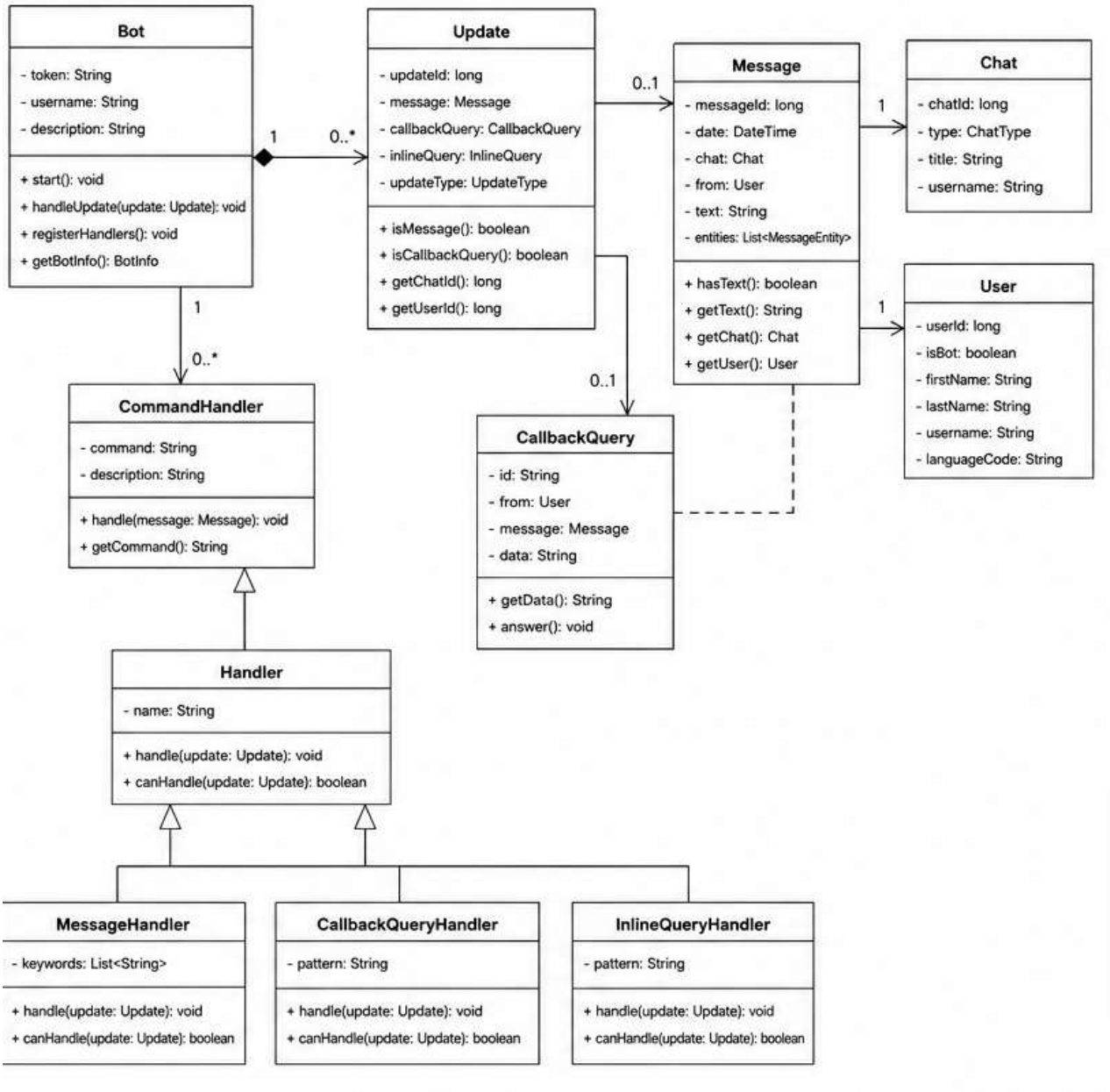


Рисунок 2.2 – Діаграма класів чат-бота

Отже, проєктування архітектури програмного комплексу дозволило визначити основні компоненти системи та організувати їх взаємодію таким чином, щоб забезпечити швидку обробку запитів користувачів, стабільну роботу Telegram-бота та можливість його подальшого вдосконалення.

2.3 Використання Telegram Bot API

Telegram Bot API є програмним інтерфейсом, який надає розробникам можливість створювати чат-ботів та організовувати їх взаємодію з

користувачами через месенджер Telegram. Завдяки використанню API програмний комплекс може отримувати повідомлення від користувачів, обробляти їх та надсилати відповіді в автоматичному режимі.

Основною перевагою Telegram Bot API є простота інтеграції, висока швидкість обробки запитів та підтримка широкого набору функцій, серед яких текстові повідомлення, інтерактивні кнопки, меню, команди, файли, зображення та повідомлення-нагадування. Це дозволяє створювати повноцінні інформаційні системи без необхідності розробки окремого мобільного застосунку.

У розробленому програмному комплексі Telegram Bot API використовується для організації взаємодії між користувачем та серверною частиною системи. Після надсилання повідомлення користувачем запит передається до серверного застосунку, де відбувається його аналіз та виконання необхідних операцій. Сервер взаємодіє з базою даних, отримує або змінює інформацію про завдання, після чого формує відповідь та надсилає її користувачу через Telegram Bot API.

Для реалізації бота використовується технологія Node.js, яка забезпечує швидку обробку великої кількості одночасних запитів та ефективну роботу із зовнішніми API. Збереження інформації про користувачів, завдання та статистику виконання здійснюється у базі даних MongoDB, що дозволяє забезпечити швидкий доступ до даних та їх подальшу обробку.

Важливою особливістю Telegram Bot API є підтримка механізму команд та інтерактивних кнопок. Використання команд /start, /menu, /tasks, /statistics та інших забезпечує швидкий доступ до функціональних можливостей програмного комплексу. Інтерактивні кнопки дозволяють значно спростити взаємодію користувача із системою та мінімізувати кількість помилок під час введення інформації.

На рисунку 2.5 наведено схему взаємодії користувача з програмним комплексом через Telegram Bot API. Запити користувача надходять до серверної частини, де відбувається їх обробка, взаємодія з базою даних та формування відповіді, яка повертається користувачу через месенджер Telegram. Такий підхід забезпечує швидку та надійну роботу системи.

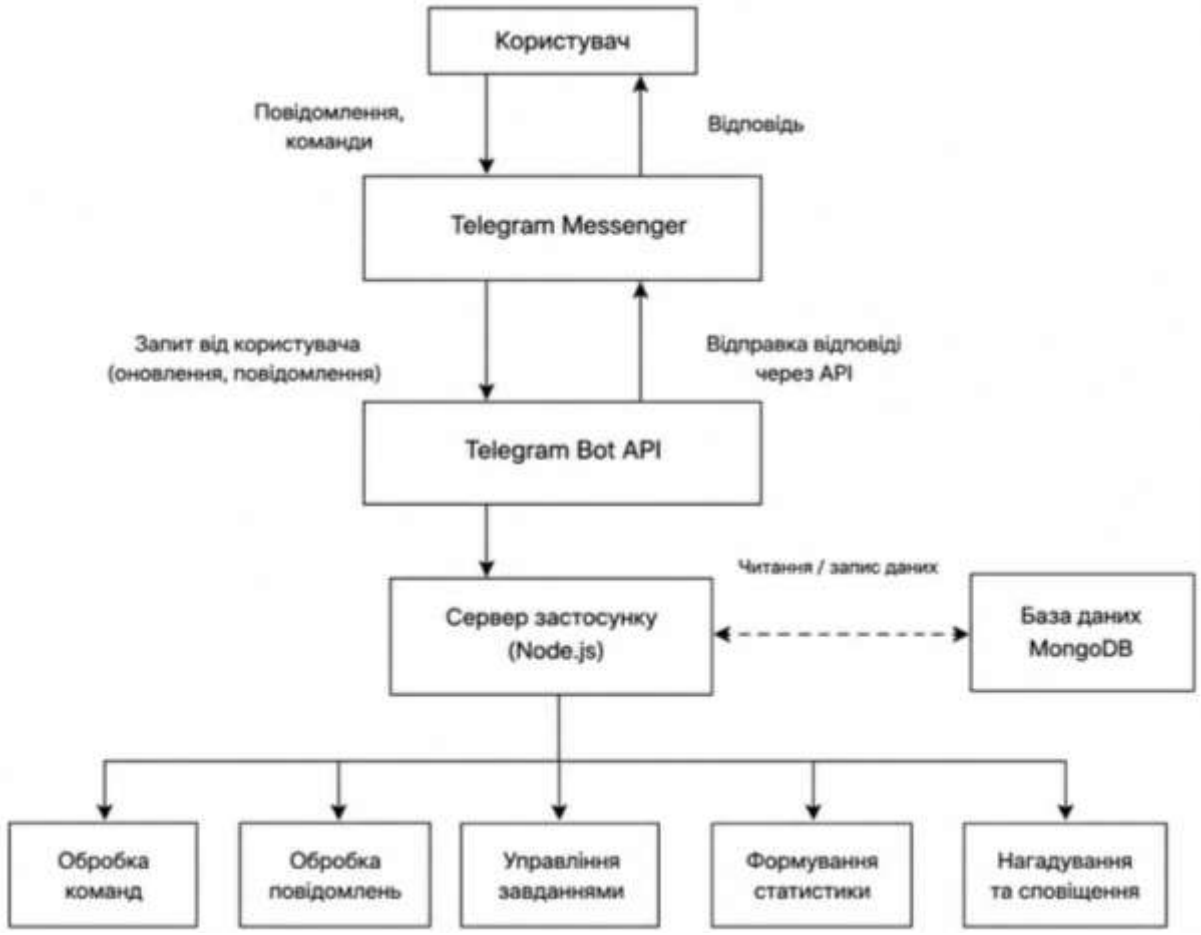


Рисунок 2.3 – Схема взаємодії Telegram Bot API з програмним комплексом

Таким чином, використання Telegram Bot API дозволило реалізувати сучасний, зручний та масштабований програмний комплекс, який забезпечує автоматизацію процесу планування завдань, контролю їх виконання та формування самодисципліни користувачів

2.4 Проєктування моделі взаємодії компонентів системи

Одним із важливих етапів проєктування Telegram-бота є визначення моделі взаємодії між його основними компонентами. Від правильно організованого процесу обміну інформацією залежить швидкодія системи, стабільність її роботи та зручність використання для кінцевого користувача.

Модель взаємодії програмного комплексу побудована за принципом клієнт-серверної архітектури, у якій користувач взаємодіє із системою через застосунок Telegram. Усі повідомлення та команди спочатку надходять до

Telegram Bot API, який виступає проміжною ланкою між клієнтом і серверною частиною програмного комплексу.

Після отримання повідомлення серверна частина виконує його аналіз та визначає необхідний сценарій обробки. Залежно від типу команди відбувається звернення до відповідного програмного модуля, який реалізує необхідну бізнес-логіку системи. Якщо для виконання запиту необхідно отримати або змінити інформацію, сервер звертається до бази даних. Після виконання відповідної операції результати повертаються до модуля формування відповіді, де створюється повідомлення для користувача.

Сформована відповідь передається через Telegram Bot API до застосунку Telegram, після чого відображається користувачу. Таким чином забезпечується двосторонній обмін інформацією між користувачем і програмним комплексом у режимі реального часу.

Для забезпечення стабільності роботи всі програмні модулі функціонують незалежно один від одного, що дозволяє модернізувати окремі компоненти без внесення змін до всієї системи. Модульний підхід також спрощує супровід програмного забезпечення та дає можливість розширювати його функціональні можливості шляхом додавання нових сервісів.

Особливістю розробленої моделі є централізована обробка запитів, яка забезпечує контроль за всіма діями користувачів, ведення журналу подій та можливість інтеграції із зовнішніми сервісами через API.

Схему взаємодії основних компонентів програмного комплексу наведено на рисуноку 2.4 де відображено послідовність проходження запиту від користувача до серверної частини та формування відповіді.

Отже, запропонована модель взаємодії забезпечує ефективний обмін інформацією між усіма компонентами Telegram-бота, підвищує надійність роботи системи та створює основу для її подальшого розвитку й масштабування.

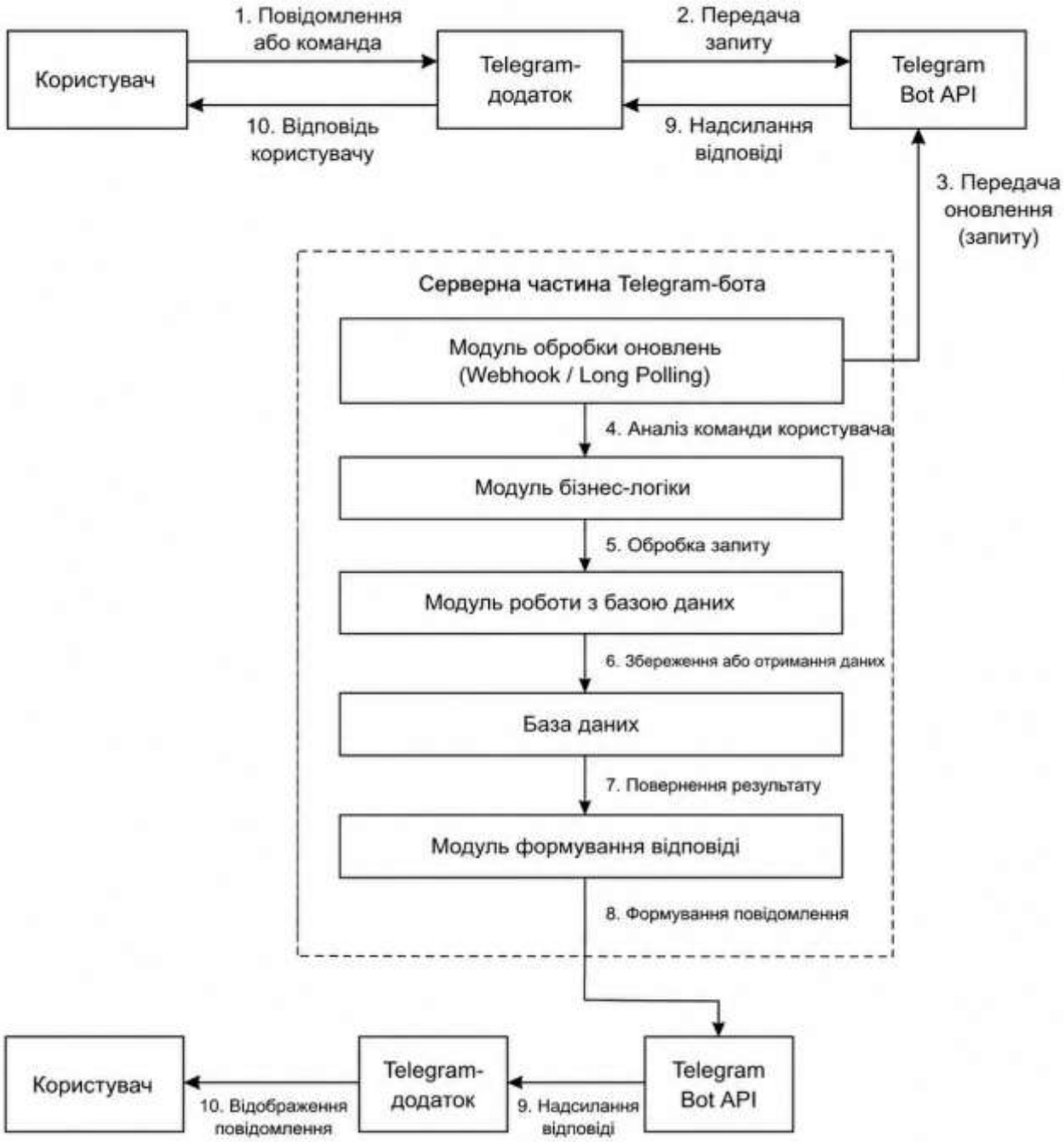


Рисунок 2.4 – Схема взаємодії основних компонентів

2.5 Проєктування та створення БД для розроблюваного комплексу

Одним із найважливіших етапів розробки Telegram-бота є проєктування бази даних. Саме вона забезпечує збереження інформації про користувачів, їхні завдання, нагадування та статистику виконання. Правильно спроєктована

структура бази даних дозволяє швидко обробляти запити, уникати дублювання інформації та забезпечувати стабільну роботу програмного комплексу.

Під час розробки було проаналізовано функціональні вимоги системи та визначено основні сутності, необхідні для реалізації роботи Telegram-бота. Враховуючи специфіку проєкту, база даних повинна забезпечувати можливість зберігання персональної інформації користувачів, списку їхніх завдань, часу нагадувань, статусу виконання та статистичних показників.[12-14]

Основною сутністю бази даних є «Користувач», яка містить інформацію про зареєстрованих користувачів Telegram-бота. Для кожного користувача зберігається його унікальний ідентифікатор Telegram, ім'я, дата реєстрації та службова інформація, необхідна для роботи системи.

Другою важливою сутністю є «Завдання». Вона містить назву завдання, його опис, дату створення, дату або час виконання, поточний статус та посилання на користувача, якому належить це завдання. Один користувач може створити необмежену кількість завдань, тому між сутностями «Користувач» і «Завдання» існує зв'язок «один до багатьох».

Для реалізації функції автоматичних повідомлень передбачено сутність «Нагадування», у якій зберігається інформація про час надсилання повідомлення та його стан. Це дозволяє системі автоматично перевіряти заплановані нагадування та надсилати їх користувачам у визначений момент.

Крім цього, у системі може використовуватися сутність «Статистика», яка накопичує інформацію про кількість виконаних завдань, відсоток їх виконання та активність користувача. На основі цих даних Telegram-бот формує статистику та відображає прогрес користувача.

На рисунку 2.5 наведено схему бази даних розробленого програмного комплексу.

З наведеної схеми видно, що центральною таблицею є таблиця Users, яка пов'язана з таблицею Tasks через ідентифікатор користувача. Таблиця Tasks, у свою чергу, містить інформацію про створені завдання та може бути пов'язана з таблицею Reminders, яка відповідає за збереження параметрів нагадувань. Дані

про виконання завдань використовуються для формування статистики активності користувача.

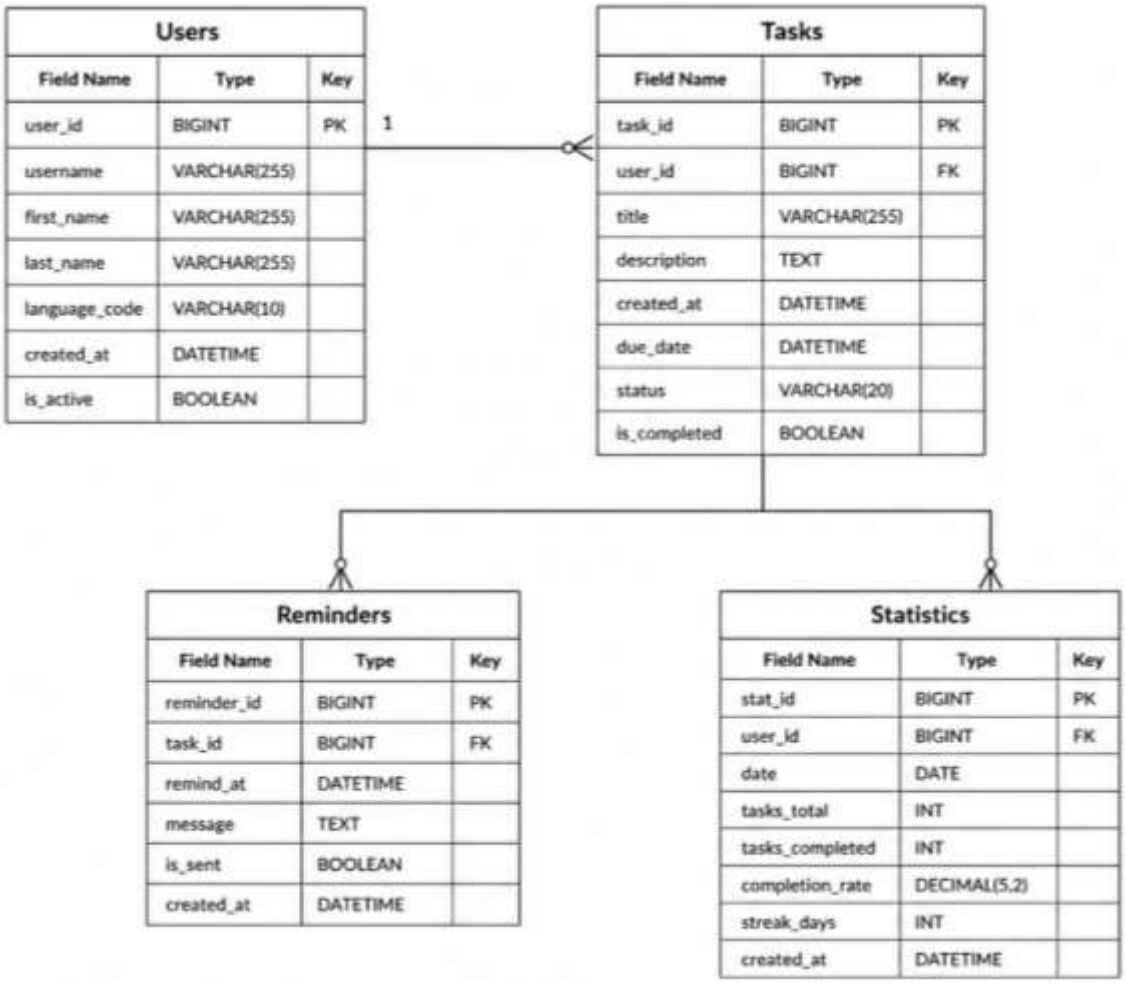


Рисунок 2.5 – Схема бази даних Telegram-бота для формування самодисципліни

Така структура бази даних забезпечує цілісність інформації, дозволяє швидко отримувати необхідні дані та легко масштабувати систему при додаванні нового функціоналу. У майбутньому структура може бути доповнена новими таблицями для реалізації системи досягнень, категорій завдань, мотиваційних повідомлень або персональних налаштувань користувачів.

Отже, спроєктована база даних повністю відповідає вимогам програмного комплексу та забезпечує ефективне зберігання й обробку інформації, необхідної для роботи Telegram-бота.

2.6 Розробка алгоритму функціонування Telegram-бота

Алгоритм функціонування Telegram-бота визначає послідовність виконання операцій, які забезпечують взаємодію користувача із програмним комплексом, обробку отриманих повідомлень та формування відповідей. Правильно побудований алгоритм дозволяє забезпечити швидку роботу системи, мінімізувати кількість помилок та підвищити ефективність використання бота.

Робота програмного комплексу починається із запуску Telegram-бота користувачем. Після надсилання команди /start система ініціалізує процес взаємодії, перевіряє наявність користувача у базі даних та, за необхідності, створює новий запис із його основними даними.

Після успішної авторизації користувачу відображається головне меню, яке містить перелік доступних функцій. Подальша робота системи залежить від вибору користувача. Бот аналізує отриману команду, визначає відповідний сценарій виконання та передає її до модуля бізнес-логіки.

На етапі обробки запиту система перевіряє правильність введених даних та визначає необхідність звернення до бази даних. Якщо для виконання операції потрібно отримати інформацію, сервер надсилає відповідний запит до бази даних, після чого отримані результати передаються до модуля формування відповіді.

Після обробки інформації бот автоматично генерує повідомлення та надсилає його користувачу через Telegram Bot API. Якщо введена команда є некоректною або не підтримується системою, користувач отримує повідомлення про помилку та рекомендації щодо подальших дій.

Для наочного відображення послідовності виконання основних операцій було побудовано діаграму діяльності зображену на рисунку 2.6. Вона демонструє логіку роботи програмного комплексу від моменту запуску бота до виконання користувацької команди та формування відповіді.

Алгоритм роботи програмного комплексу побудований циклічно. Після завершення виконання кожної операції бот повертається до режиму очікування

нових повідомлень, що забезпечує безперервну взаємодію з користувачами та можливість одночасного обслуговування великої кількості запитів.

Рисунок 2.6 – Діаграма діяльності телеграм-бота

Важливою особливістю алгоритму є модульність його структури. Кожна команда обробляється окремим програмним модулем, що спрощує модернізацію системи та дозволяє без труднощів додавати нові функції без зміни загальної логіки роботи програмного комплексу.

Схему алгоритму функціонування Telegram-бота наведено у Додатку В , де графічно представлено послідовність виконання основних операцій від моменту отримання повідомлення користувача до формування та надсилання відповіді.

Отже, розроблений алгоритм забезпечує стабільне функціонування Telegram-бота, ефективну обробку запитів користувачів та швидке виконання всіх передбачених функцій. Використання такого алгоритму підвищує

надійність програмного комплексу та створює можливість його подальшого розвитку й розширення функціональних можливостей.

2.7 Висновок до другого розділу

У другому розділі кваліфікаційної роботи було розглянуто основні підходи до проєктування Telegram-бота та визначено методи побудови його програмної архітектури. На основі проведеного аналізу сформовано структуру програмного комплексу, яка забезпечує логічний розподіл функцій між окремими модулями та створює основу для подальшої реалізації програмного продукту.

У процесі роботи було спроєктовано архітектуру програмного комплексу, визначено модель взаємодії між користувачем, Telegram Bot API, серверною частиною та базою даних. Запропонована структура забезпечує ефективний обмін інформацією між усіма компонентами системи та дозволяє швидко обробляти запити користувачів.

Також було розроблено інформаційну модель бази даних, яка забезпечує централізоване зберігання інформації про користувачів, їхні запити та результати роботи Telegram-бота. Запропонована структура є гнучкою та може бути розширена відповідно до подальших функціональних потреб програмного комплексу.

Особливу увагу приділено розробці алгоритму функціонування Telegram-бота, який визначає послідовність обробки повідомлень, виконання команд, взаємодії з базою даних та формування відповідей користувачам. Використання модульного підходу до побудови алгоритму спрощує супровід програмного забезпечення та забезпечує можливість швидкого додавання нових функцій.

Результати проєктування підтверджують доцільність використання клієнт-серверної архітектури, сучасних засобів програмування та централізованої системи зберігання даних для реалізації Telegram-бота. Запропоновані рішення забезпечують високу продуктивність, масштабованість та надійність роботи програмного комплексу.

Отже, виконане проєктування створило необхідну теоретичну та практичну основу для подальшої реалізації програмного комплексу, його тестування та впровадження. Розроблені моделі, алгоритми та структура системи повністю відповідають поставленим вимогам і забезпечують ефективне функціонування Telegram-бота в умовах його практичного використання.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМНОГО КОМПЛЕКСУ

3.1 Проєктування інтерфейсу користувача

Проєктування інтерфейсу користувача є одним із важливих етапів розробки Telegram-бота, оскільки саме від зручності взаємодії із системою залежить комфорт використання програмного комплексу. Основною метою було створення простого та інтуїтивно зрозумілого інтерфейсу, який дозволить користувачу швидко отримувати доступ до необхідних функцій без зайвих дій.

На відміну від традиційних вебзастосунків, інтерфейс Telegram-бота реалізований у вигляді діалогу між користувачем та системою. Усі дії виконуються за допомогою текстових команд або інтерактивних кнопок, що значно спрощує процес роботи та не потребує додаткового навчання [14-17].

Після першого запуску бот вітає користувача та пропонує перейти до головного меню. Саме головне меню є центральним елементом інтерфейсу, через який здійснюється доступ до всіх функцій програмного комплексу.

Основними елементами інтерфейсу є:

– кнопка створення нового завдання;

– перегляд списку завдань;

– перегляд виконаних завдань;

– налаштування нагадувань;

– перегляд статистики та прогресу;

– отримання мотиваційних повідомлень;

– довідка про роботу бота.

Під час створення нового завдання користувач послідовно вводить його назву, за необхідності опис та дату або час виконання. Такий підхід дозволяє уникнути перевантаження інформацією та робить процес додавання нових справ максимально простим.

Для перегляду списку завдань бот формує повідомлення, у якому відображаються всі актуальні справи користувача із зазначенням їхнього статусу. Після виконання завдання користувач може одним натисканням кнопки змінити його статус на «Виконано».

Окремий розділ інтерфейсу присвячений статистиці. У ньому користувач може переглянути кількість виконаних завдань, загальний прогрес та власну активність за певний проміжок часу. Це допомагає оцінити ефективність планування та мотивує до подальшого розвитку самодисципліни.

Для забезпечення максимальної зручності всі повідомлення мають лаконічний вигляд, а кнопки згруповані відповідно до їхнього призначення. Завдяки цьому користувач може швидко знаходити необхідні функції та взаємодіяти із системою без складних налаштувань.

На рисунку 3.1 наведено структуру головного меню Telegram-бота.

Рисунок 3.1 – Структура головного меню Telegram-бота

В додатку Б представлено приклад взаємодії користувача із Telegram-ботом під час створення нового завдання.

Таким чином, розроблений інтерфейс користувача є простим, зрозумілим та зручним у використанні. Його структура забезпечує швидкий доступ до всіх основних функцій Telegram-бота та створює комфортні умови для щоденного використання програмного комплексу.

3.2 Розробка програмного комплексу

3.2.1 Загальна характеристика

Після завершення етапів аналізу предметної області, проєктування структури програмного комплексу та створення бази даних було розпочато безпосередню розробку Telegram-бота для формування самодисципліни користувачів. Основною метою цього етапу стало створення програмного продукту, який забезпечує швидку та зручну взаємодію користувача із системою, автоматизує процес планування завдань і допомагає контролювати їх виконання.

Розробка програмного комплексу здійснювалася з використанням мови програмування JavaScript та середовища виконання Node.js. Для взаємодії із сервісом Telegram використовувався Telegram Bot API, який забезпечує передачу повідомлень між користувачем та серверною частиною програми.

Архітектура Telegram-бота побудована за модульним принципом. Кожен модуль відповідає за виконання окремого функціонального завдання, що робить програмний код більш структурованим та спрощує його подальше супроводження. Такий підхід також дозволяє легко додавати нові функції без зміни основної логіки роботи системи.

Після запуску бота користувач проходить початкову взаємодію із системою та отримує доступ до головного меню. Усі команди, які надсилає користувач, обробляються серверною частиною програми, після чого виконується відповідна бізнес-логіка та формується відповідь.

Під час розробки особлива увага була приділена реалізації функціоналу створення та керування завданнями. Користувач має можливість додавати нові завдання, переглядати їх список, редагувати інформацію або позначати завдання

як виконані. Усі зміни автоматично зберігаються у базі даних, що забезпечує постійне збереження інформації.

Однією з ключових функцій програмного комплексу є система автоматичних нагадувань. Після встановлення дати та часу виконання завдання бот у потрібний момент надсилає користувачу повідомлення, що допомагає не пропустити важливі справи та сприяє розвитку самодисципліни.

Крім основного функціоналу, у програмному комплексі реалізовано модуль статистики, який аналізує активність користувача та формує інформацію про кількість виконаних завдань і рівень продуктивності. Це дозволяє користувачу оцінювати власний прогрес та мотивує його до регулярного виконання поставлених цілей [18].

Для відображення процесу обміну повідомленнями між компонентами системи було побудовано діаграму послідовності зображена на рисунку 3.2. Вона демонструє порядок взаємодії користувача, Telegram Bot API, серверної частини та бази даних під час виконання запиту.

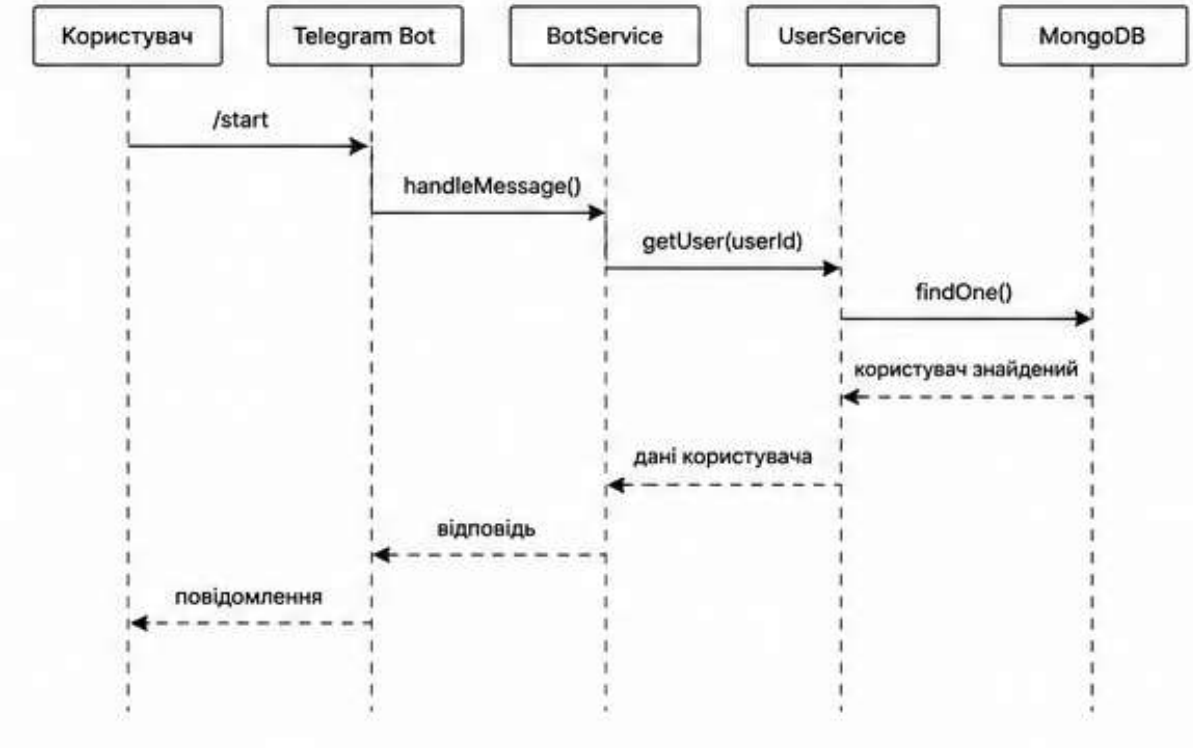


Рисунок 3.2 – Діаграма послідовності взаємодії телеграм-бота

Таким чином, у результаті розробки було створено функціональний Telegram-бот, який забезпечує ефективне управління особистими завданнями, автоматичне надсилання нагадувань та ведення статистики виконання. Реалізований програмний комплекс є зручним інструментом для підвищення рівня самоорганізації користувачів і може бути розширений новими можливостями у майбутньому.

У наступних підрозділах буде детально розглянуто реалізацію інтерфейсу користувача та основних функцій обробки даних, які забезпечують повноцінну роботу Telegram-бота.

3.2.2 Розробка інтерфейсу

Одним із головних етапів створення Telegram-бота є розробка інтерфейсу користувача. Саме від його простоти та зрозумілості залежить зручність використання програмного комплексу та швидкість виконання необхідних дій. Оскільки взаємодія із системою відбувається через месенджер Telegram, інтерфейс реалізований у вигляді текстових повідомлень та інтерактивних кнопок.

Під час проєктування інтерфейсу основна увага приділялася мінімізації кількості дій, які необхідно виконати користувачу для досягнення поставленої мети. Усі основні функції згруповані в головному меню, що дозволяє швидко перейти до потрібного розділу без введення складних команд.

Після першого запуску Telegram-бот вітає користувача та пропонує ознайомитися з його можливостями. Далі на екрані відображається головне меню, яке містить основні кнопки для роботи із системою. Через це меню користувач може створити нове завдання, переглянути список активних або виконаних завдань, перейти до статистики, налаштувати нагадування або отримати довідкову інформацію.

При натисканні кнопки «Створити завдання» бот послідовно пропонує користувачу ввести назву завдання, його опис та дату або час виконання. Після

введення всієї необхідної інформації дані автоматично зберігаються у базі даних, а користувач отримує повідомлення про успішне створення нового запису.

Для перегляду завдань передбачено окремий розділ, у якому відображається перелік усіх створених справ. Біля кожного завдання може міститися інформація про його статус та дату виконання. Користувач має можливість переглядати детальну інформацію, редагувати завдання або позначати його як виконане.

Окремий інтерфейс реалізовано для роботи зі статистикою. У цьому розділі відображаються основні показники активності користувача, кількість виконаних завдань, відсоток їх виконання та інші статистичні дані, що дозволяють оцінити особисту продуктивність.

Для підвищення зручності використання програмного комплексу більшість функцій реалізовано за допомогою інтерактивних кнопок Telegram. Це дозволяє уникнути помилок при введенні команд та значно спрощує взаємодію користувача із системою [19-21]. На рисунку 3.2 наведено головне меню.

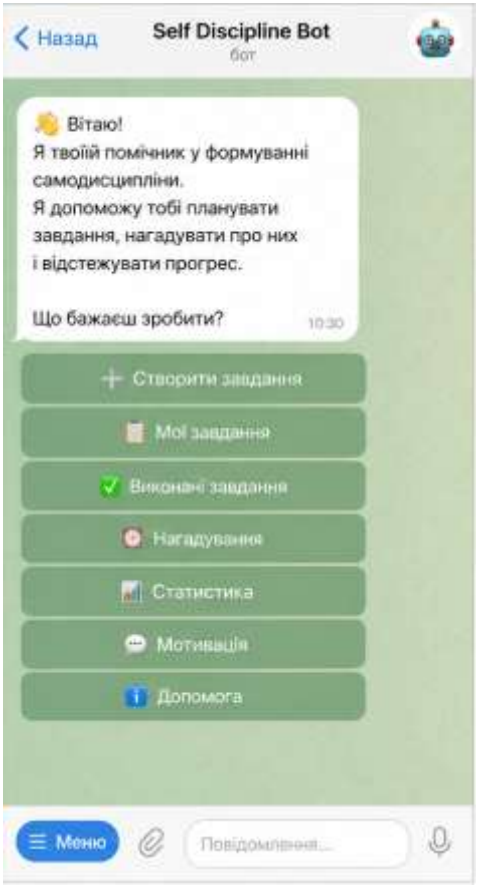


Рисунок 3.2 – Головне меню Telegram-бота

На рисунку 3.3 представлено інтерфейс створення нового завдання. Цей елемент є одним із ключових у роботі Telegram-бота, оскільки саме через нього користувач формує перелік своїх майбутніх справ та планує їх виконання.

Під час створення нового завдання бот послідовно запитує необхідну інформацію, зокрема назву завдання, його короткий опис та дату або час виконання. Такий покроковий підхід спрощує процес введення даних і мінімізує ймовірність помилок з боку користувача.

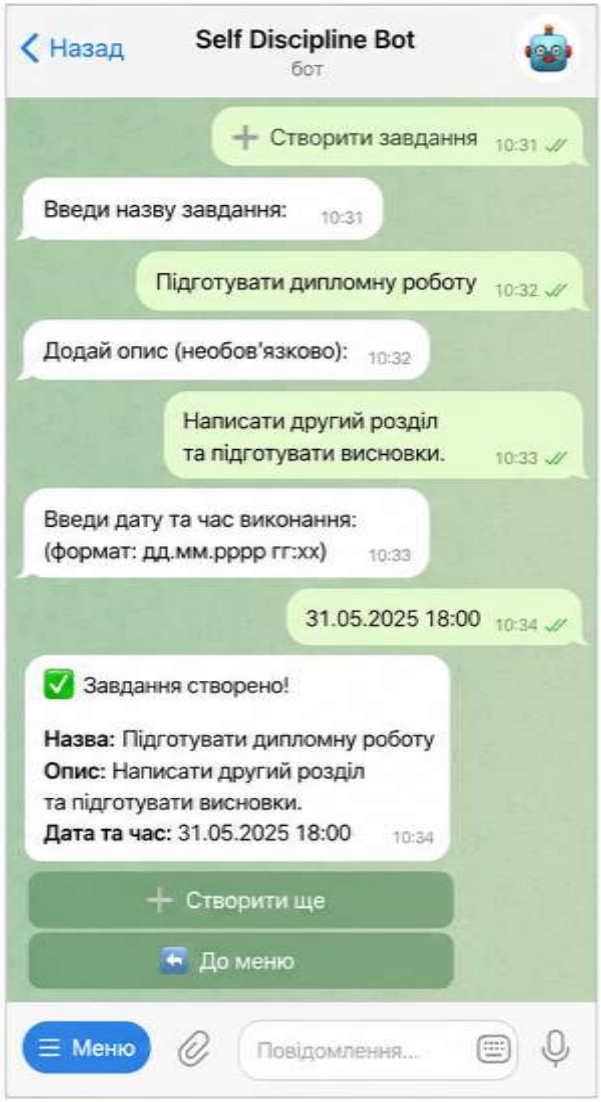


Рисунок 3.3 – Вікно створення нового завдання

На рисунку 3.3 показано приклад перегляду списку завдань користувача. У цьому розділі Telegram-бот відображає всі створені користувачем завдання із зазначенням їх назви, дати виконання та поточного статусу. Завдяки такому

поданню інформації користувач може швидко переглянути перелік запланованих справ, оцінити їх актуальність та визначити пріоритетність виконання.

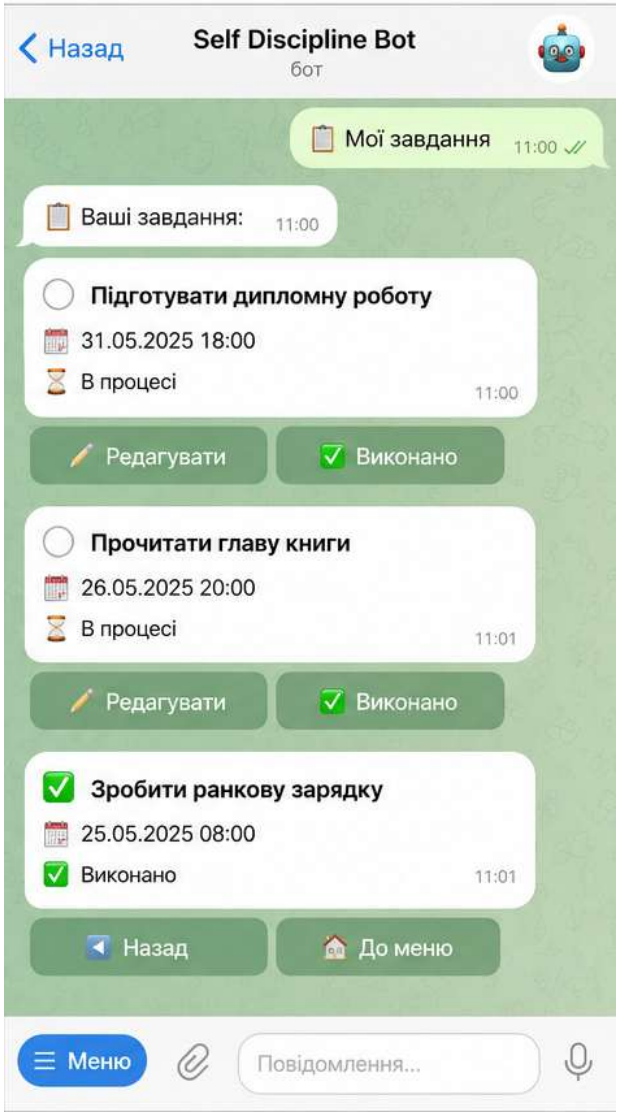


Рисунок 2.8 – Інтерфейс перегляду списку завдань

Таким чином, розроблений інтерфейс є простим, зрозумілим та інтуїтивно доступним для користувача. Використання кнопок, логічної структури меню та послідовного введення інформації забезпечує комфортну роботу із Telegram-ботом та сприяє його ефективному використанню у повсякденному житті.

3.2.3 Розробка функцій для обробки даних

Одним із найважливіших етапів створення Telegram-бота є реалізація функцій, що забезпечують обробку даних користувачів та взаємодію із базою даних. Саме ці програмні модулі відповідають за створення, редагування, видалення та отримання інформації про завдання, а також за формування статистики й роботу системи нагадувань.

Після надсилання повідомлення користувачем Telegram Bot API передає його серверній частині програмного комплексу. Сервер аналізує отриману команду, визначає необхідну дію та викликає відповідну функцію. Залежно від типу запиту відбувається звернення до бази даних, після чого формується відповідь і надсилається користувачу.

Для реалізації основного функціоналу було створено набір окремих функцій, кожна з яких виконує визначене завдання. Такий підхід дозволяє зробити програмний код більш зрозумілим, структурованим та зручним для подальшого супроводу.

Нижче наведено фрагмент коду, який відповідає за створення нового завдання та його збереження у базі даних.

Лістинг 3.1 – Функція створення нового завдання

```
const task = await Task.create({
  userId,
  title,
  description,
  dueDate,
  status: "active",
});
return task;
```

Після отримання команди на створення нового завдання бот перевіряє введені користувачем дані, створює новий запис у базі даних та повертає інформацію про успішне виконання операції.

Для отримання списку всіх завдань користувача реалізовано окрему функцію, яка здійснює пошук записів за ідентифікатором користувача.

Лістинг 3.2 – Отримання списку завдань користувача

```
const tasks = await Task.find({
  userId: ctx.from.id,
});
return tasks;
```

Після виконання запиту користувач отримує актуальний перелік усіх своїх завдань, які зберігаються у базі даних.

Окрему увагу під час розробки було приділено реалізації функції оновлення статусу завдання після його виконання.

Лістинг 3.3 – Позначення завдання як виконаного

```
await Task.findByIdAndUpdate(id, {
  status: "completed",
});
```

Використання цієї функції дозволяє автоматично змінити стан завдання та врахувати його під час формування статистики.

Не менш важливою складовою роботи Telegram-бота є система автоматичних нагадувань. Для цього сервер періодично перевіряє всі заплановані завдання та визначає, чи настав час надсилання повідомлення.

Лістинг 3.4 – Перевірка часу нагадування

```
if (task.dueDate <= new Date()) {
  bot.telegram.sendMessage(
    task.userId,
    task.title
  );
}
```

Завдяки цьому користувач своєчасно отримує повідомлення про необхідність виконання запланованої справи.

Для аналізу активності користувача реалізовано функцію підрахунку виконаних завдань.

Лістинг 3.5 – Формування статистики

```
const completed = await Task.countDocuments({
  userId,
  status: "completed",
});
```

Отримані результати використовуються для формування статистики продуктивності та відображення особистого прогресу користувача.

Однією з основних функцій програмного комплексу є отримання інформації про завдання користувача з бази даних. Для цього використовується запит пошуку документів за ідентифікатором користувача та сортування їх за датою створення.

Лістинг 3.5 – Отримання списку завдань користувача

```
const tasks = await Task.find({
    userId: chatId
}).sort({
    createdAt: -1
});
```

Наведений запит здійснює пошук усіх завдань, що належать конкретному користувачу, після чого сортує їх у порядку спадання дати створення. Використання методу find() дозволяє швидко отримати необхідні записи, а функція sort() забезпечує відображення найновіших завдань першими.

Під час створення нового завдання використовується операція додавання документа до колекції.

Лістинг 3.6 – Додавання нового завдання

```
await Task.create({
    userId: chatId,
    title: taskTitle,
    deadline: deadline,
    status: "active"
});
```

Даний фрагмент коду створює новий запис у базі даних та автоматично зберігає інформацію про користувача, текст завдання, дату виконання та його поточний статус. Для зміни стану завдання використовується операція оновлення документа.

Лістинг 3.7 – Оновлення статусу завдання

```
await Task.updateOne(
    { _id: taskId },
    {
        $set: {
            status: "completed"
        }     } );
```

Запит знаходить завдання за його ідентифікатором та змінює значення поля status на completed, що свідчить про його успішне виконання.

Для підвищення швидкодії роботи програмного комплексу у базі даних використовується індексування.

Лістинг 3.8 – Створення індексу

```
taskSchema.index({
    userId: 1,
    deadline: 1
});
```

Індекс дозволяє значно прискорити пошук завдань користувача та їх сортування за датою виконання, що особливо важливо при роботі з великою кількістю записів.

Для автоматичного оновлення інформації може використовуватися тригер (middleware Mongoose), який виконується перед збереженням документа.

Лістинг 3.9 – Автоматичне оновлення дати редагування

```
taskSchema.pre("save", function(next) {
    this.updatedAt = new Date();
    next();
});
```

Наведений middleware автоматично встановлює поточну дату та час у поле updatedAt перед збереженням документа. Такий механізм дозволяє вести історію змін без додаткових дій з боку програміста та підтримує актуальність інформації у базі даних.

Таким чином, реалізовані функції забезпечують повний цикл роботи Telegram-бота: від отримання запиту користувача до збереження інформації, її обробки та формування відповіді. Основна частра коду відображена Додатку Г. Використання окремих програмних модулів робить код більш структурованим, полегшує його підтримку та створює можливість для подальшого розширення функціональних можливостей програмного комплексу.

3.3 Тестування розробленого програмного комплексу

Після завершення розробки Telegram-бота було проведено його тестування з метою перевірки коректності роботи всіх реалізованих функцій. Основним завданням тестування було виявлення можливих помилок, оцінка стабільності роботи системи та перевірка відповідності отриманих результатів поставленим вимогам.

Тестування проводилося шляхом послідовної перевірки кожної функції програмного комплексу окремо, а також комплексної перевірки роботи всієї системи. Особлива увага приділялася перевірці взаємодії користувача з Telegram-ботом, роботі із базою даних та системі автоматичних нагадувань.

На першому етапі було перевірено запуск Telegram-бота та його підключення до Telegram Bot API. Після надсилання команди /start бот коректно відображав вітальне повідомлення та головне меню, що свідчить про правильне налаштування системи. Результат відображено на рисунку 3.1

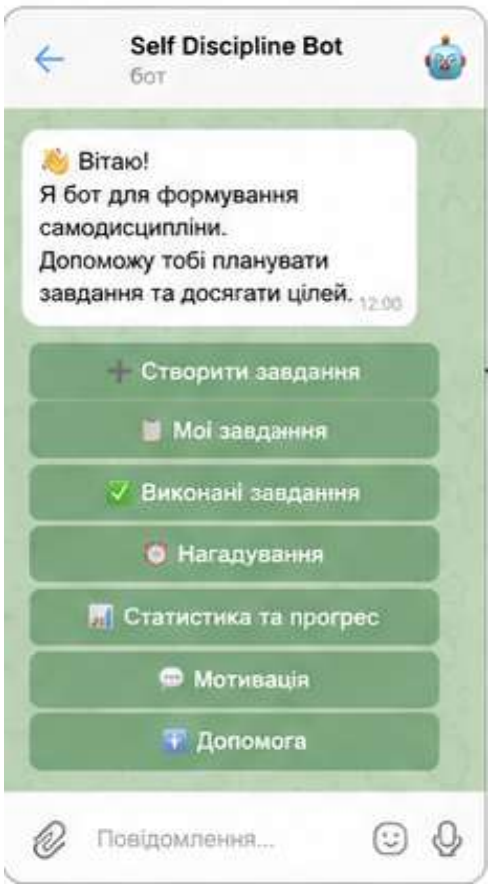


Рисунок 3.1 – Початок роботи з ботом

Наступним етапом стало тестування функції створення нового завдання(рис3.2). Було перевірено можливість введення назви, опису та часу виконання завдання. Після завершення введення інформації запис успішно зберігався у базі даних та відображався у списку завдань користувача.

Рисунок 3.2 – Створення нового завдання

Також було протестовано функцію перегляду списку завдань(рис 3.3). Система коректно отримувала інформацію з бази даних та відображала всі створені записи відповідно до користувача, який виконував запит.

Окрему увагу приділено перевірці роботи системи нагадувань. Після встановлення часу виконання завдання бот автоматично надсилав повідомлення у зазначений момент, що підтвердило правильність роботи механізму планування повідомлень.

Під час тестування було перевірено можливість редагування та видалення завдань. Усі внесені зміни коректно зберігалися у базі даних та одразу відображалися користувачу після оновлення списку.

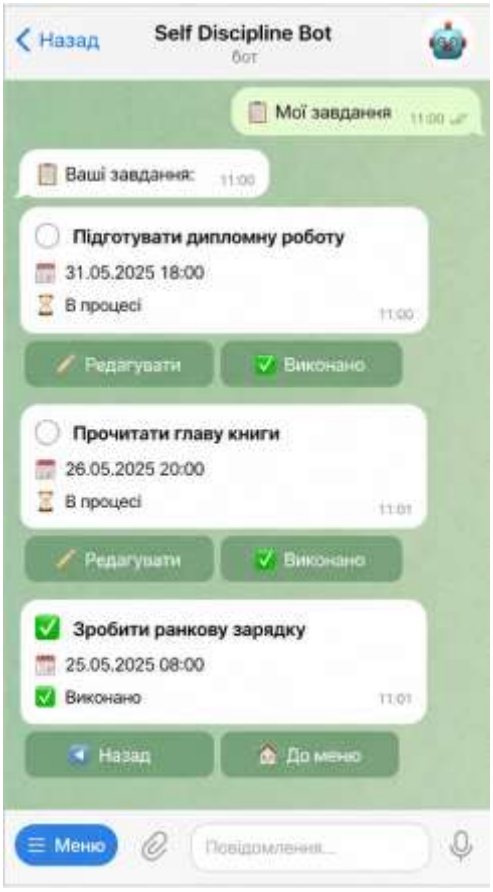


Рисунок 3.3 – Список завданнь

Також було протестовано функцію формування статистики. Після позначення завдань як виконаних система правильно підраховувала їх кількість та відображала актуальну інформацію про активність користувача.

Результати проведеного тестування наведено у таблиці 3.1.

Під час проведення тестування критичних помилок у роботі програмного комплексу виявлено не було. Усі реалізовані функції працюють відповідно до поставлених вимог та забезпечують коректну взаємодію користувача із системою.

Отже, результати тестування підтверджують працездатність розробленого Telegram-бота та його готовність до практичного використання. Реалізований програмний комплекс забезпечує стабільну роботу, швидку обробку запитів та коректне виконання всіх основних функцій, необхідних для формування самодисципліни користувачів.

Таблиця 3.1 – Результати тестування Telegram-бота

Функція	**Очікуваний результат**	**Отриманий результат**	**Статус**
Запуск бота	Відображення головного меню	Функція працює коректно	Успішно
Створення завдання	Збереження нового запису	Завдання створено	Успішно
Перегляд завдань	Відображення списку	Дані відображаються	Успішно
Редагування завдання	Оновлення інформації	Дані оновлено	Успішно
Видалення завдання	Видалення запису	Завдання видалено	Успішно
Надсилання нагадувань	Повідомлення у потрібний час	Повідомлення надіслано	Успішно
Формування статистики	Відображення прогресу	Статистика сформована	Успішно

Перейдемо до перспектив подальшої модернізації програмного продукту.

3.4 Перспективи модернізації

Розроблений Telegram-бот для формування самодисципліни є повноцінним програмним продуктом, який реалізує основні функції планування завдань, надсилання нагадувань та ведення статистики активності користувачів. Проте сучасні інформаційні технології постійно розвиваються, тому програмний комплекс має значний потенціал для подальшого вдосконалення та розширення функціональних можливостей.

Одним із перспективних напрямів розвитку є впровадження системи гейміфікації. Користувачі можуть отримувати бали, досягнення або рівні за

регулярне виконання завдань, що стане додатковою мотивацією для підтримання дисципліни та досягнення поставлених цілей.

Ще одним напрямом модернізації може бути реалізація категорій та пріоритетів для завдань. Це дозволить користувачам розподіляти справи за важливістю, створювати окремі списки для навчання, роботи чи особистих потреб та ефективніше організовувати свій час.

Доцільним є також впровадження календаря, який дозволить переглядати всі заплановані завдання у вигляді щоденного, тижневого або місячного розкладу. Така функція зробить використання Telegram-бота ще більш зручним та інформативним.

У майбутньому програмний комплекс можна інтегрувати із сервісами Google Calendar або Microsoft Outlook, що забезпечить автоматичну синхронізацію подій та нагадувань між різними платформами.

Перспективним напрямом розвитку є використання технологій штучного інтелекту. На основі аналізу поведінки користувача система зможе формувати персональні рекомендації, визначати найбільш продуктивний час для виконання завдань та пропонувати оптимальний розподіл навантаження протягом дня.

Також можна реалізувати модуль аналітики, який буде будувати графіки продуктивності, визначати кількість виконаних завдань за різні періоди та формувати детальні звіти про активність користувача. Це допоможе краще оцінювати власні результати та планувати подальшу діяльність.

Для підвищення рівня персоналізації доцільно додати можливість зміни мови інтерфейсу, вибору тем оформлення та налаштування індивідуального графіка надсилання мотиваційних повідомлень.

Крім цього, перспективним є створення вебпанелі адміністратора, яка дозволить контролювати роботу Telegram-бота, переглядати статистику використання системи, керувати налаштуваннями та здійснювати моніторинг працездатності програмного комплексу.

Отже, розроблений Telegram-бот має широкі можливості для подальшої модернізації та вдосконалення. Реалізація запропонованих напрямів розвитку дозволить розширити функціональні можливості програмного комплексу,

підвищити його ефективність та зробити його ще більш корисним інструментом для формування самодисципліни й організації особистого часу користувачів.

3.5 Висновок до третього розділу

У третьому розділі кваліфікаційної роботи розглянуто основні етапи проєктування, реалізації та тестування Telegram-бота для формування самодисципліни користувачів. На основі сформованих у першому розділі вимог було розроблено структуру програмного комплексу, яка забезпечує логічну взаємодію всіх його компонентів та ефективну обробку запитів користувачів.

У процесі виконання роботи було спроєктовано базу даних для зберігання інформації про користувачів, завдання, нагадування та результати їх виконання. Запропонована структура дозволяє забезпечити швидкий доступ до даних, їх цілісність та можливість подальшого розширення функціоналу програмного комплексу.

Особливу увагу було приділено проєктуванню інтерфейсу користувача. Завдяки використанню можливостей месенджера Telegram вдалося створити простий, зрозумілий та інтуїтивний інтерфейс, який не потребує спеціальних навичок для роботи із системою. Використання інтерактивних кнопок і послідовних діалогів робить процес взаємодії максимально зручним.

Під час реалізації програмного комплексу було створено основні функціональні модулі, що відповідають за обробку команд користувачів, керування завданнями, автоматичне надсилання нагадувань та формування статистики активності. Усі дані зберігаються у базі даних та обробляються серверною частиною програмного комплексу, що забезпечує стабільну та безперебійну роботу Telegram-бота.

Проведене тестування підтвердило коректність роботи всіх реалізованих функцій. Було перевірено створення, редагування та видалення завдань, роботу системи нагадувань, перегляд статистики та взаємодію з базою даних. У результаті тестування встановлено, що програмний комплекс відповідає поставленим вимогам і забезпечує правильне виконання всіх основних операцій.

Також у розділі було розглянуто перспективи подальшої модернізації системи. Запропоновано можливості впровадження гейміфікації, інтеграції із зовнішніми сервісами, розширеної аналітики, персоналізованих рекомендацій та інших функцій, які можуть зробити Telegram-бот ще більш ефективним інструментом для організації особистого часу та розвитку самодисципліни.

Отже, результати другого розділу свідчать про успішну реалізацію програмного комплексу та підтверджують можливість його практичного використання. Розроблений Telegram-бот є сучасним, функціональним та зручним програмним продуктом, який допомагає користувачам планувати свою діяльність, контролювати виконання завдань і підвищувати рівень особистої продуктивності.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ХОРОНИ ПРАЦІ

4.1 Працездатність людини-оператора

У процесі роботи людина переживає різні функціональні стани, які зумовлюють різні рівні її працездатності.

Під працездатністю людини розуміють можливість її виконувати певну роботу з необхідною якістю та у встановлений час. Працездатність людини залежить як від зовнішніх факторів, так і від внутрішнього стану (внутрішні фактори).

До зовнішніх факторів належать:

– кількість та форма отриманої інформації;
– зручність робочого місця;
– характер взаємовідносин в колективі;
– вплив факторів середовища існування.

До внутрішніх факторів належать:

– рівень підготовки;
– тренованість людини;
– емоційна стійкість.

Розглядаючи зміни функціонального стану та якості роботи людини у процесі одного трудового циклу (зміни), виділяють 4 фази працездатності: пристосування до праці. стійкої працездатності, субкомпенсації, втоми. Тривалість усіх фаз та усього циклу роботи залежить від рівня підготовки людини до роботи (рис. 4.1) [18].

Фаза пристосування до праці (0 – t1) – це час, протягом якого людина адаптується до умов праці. Основний показник ефективності праці поступово досягає свого встановленого значення. Тривалість періоду пристосування організму до умов праці залежить від багатьох факторів, серед яких основними є інтенсивність роботи (чим інтенсивніша робота, тим цей період коротший) та рівень готовності людини до майбутньої роботи.

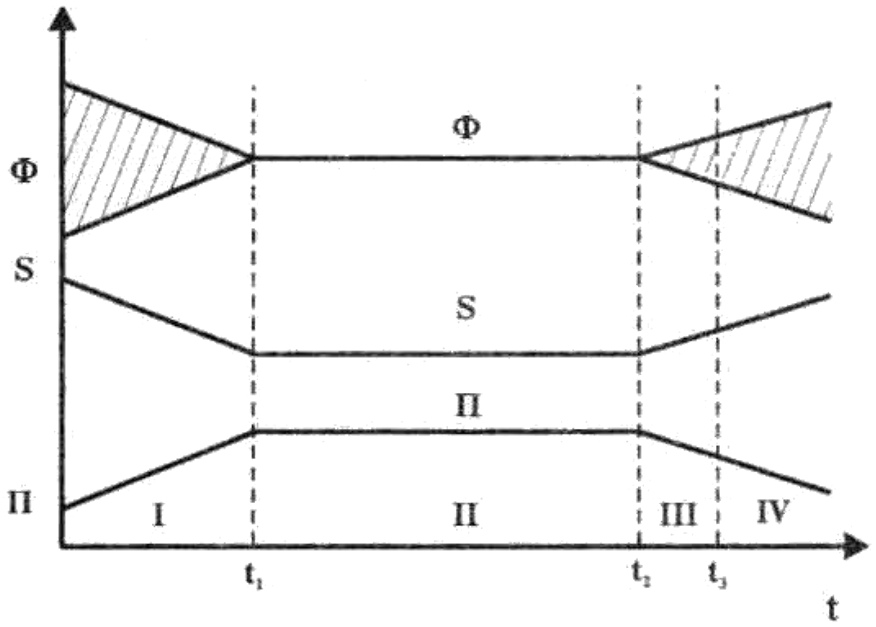


Рисунок 4.1 – Фази працездатності (Ф – показник функціонального стану; S – помилки роботи; П – продуктивність праці)

Фаза стійкої працездатності (t1 – t2) характеризується найвищою якістю праці при оптимальних рівнях функціонування фізіологічних систем організму. Тривалість цього періоду залежить від інтенсивності роботи. Чим інтенсивніша праця, тим коротший цей період. Найоптимальніша динамічна робота, коли цей період може бути в десятки разів довшим, ніж при статичній діяльності.

На процес стійкої працездатності також впливають емоції. Негативні знижують працездатність, а позитивні значно продовжують період стійкої працездатності.

Продовження періоду стійкої працездатності можна забезпечити:

– оптимальним рівнем напруги психофізіологічних функцій;

– комфортними умовами праці;

– правильним поєднанням режимів праці та відпочинку;

– емоційним розвантаженням;

– використанням тонізуючих напоїв (кава, чай), фармакологічних засобів, зокрема препаратів рослинного походження (вітаміни, препарати, які впливають на енергетичні та метаболічні процеси);

– інформуванням людини про наслідки її діяльності, наглядом та контролем її роботи.

Фаза субкомпенсації (t2 – t3) розглядається як початок розвитку втоми. В цей період якість праці ще зберігається на високому рівні, але тільки за рахунок перенапруги відповідних функцій організму.

Фаза втоми (з моменту t3) характеризується чітко вираженим зниженням якості роботи при подальшому погіршенні функціонального стану людини. Об'єктивними показниками втоми є зміна частоти пульсу, дихання, зорової та слухової чутливості.

Наступною фазою життєдіяльності людини повинна бути фаза відновлення працездатності (відпочинку), яка може тривати від декількох хвилин до декількох годин і навіть декілька діб [20].

Підсумовуючи вище описане можна вивести такі рекомендації:

– запровадити невеликі перерви для відпочинку;

– облаштувати зону з різними напоями;

– влаштовувати різні заходи для згуртування колективу та емоційного розвантаження.

4.2 Загальні вимоги безпеки з охорони праці для користувачів ПК

До роботи на персональному комп’ютері допускають осіб, які пройшли інструктажі з питань охорони праці та пожежної безпеки. До самостійної роботи з персональним комп'ютером допускаються особи, які досягли 18 річного віку, пройшли медичний огляд, ознайомлені з інструкцією з охорони праці при роботі з оргтехнікою, не мають протипоказань за станом здоров'я [21].

Вимоги до приміщення включають в себе освітлення, захист від сонця, розташування предметів в кабінеті. Також є деякі вимоги до предметів інтер’єру.

Освітлення повинно бути змішаним (природним та штучним). Освітлювальні установки повинні забезпечувати рівномірне освітлення і не повинні утворювати засліплюючих відблисків на клавіатурі, а також на екрані монітора за напрямом очей.

Для захисту від прямих сонячних променів повинні передбачатися сонцезахисні пристрої (плівка з металізованим покриттям, регульовані жалюзі з вертикальними панелями тощо) [20].

При роботі з ПК не допускається розташування робочого місця в приміщеннях без наявності природної або штучної вентиляції. Робоче місце з комп'ютером повинно розміщуватися на відстані не менше 1м від стіни, а від стіни з віконними отворами – на відстані не менше 1,5 м.

У приміщенні кабінету і на робочому місці необхідно підтримувати чистоту і порядок, проводити систематичне провітрювання.

При розміщенні елементів робочого місця слід враховувати:

– робочу позу користувача;

– простір для розміщення користувача;

– можливість огляду елементів робочого місця;

– можливість огляду простору поза межами робочого місця;

– можливість робити записи, розміщувати на робочому столі документацію та матеріали, які використовує користувач.

Конструкція робочого столу має бути такою, щоб оптимально розмістити на робочій поверхні обладнання, що використовують у роботі. Крісло має забезпечувати підтримування раціональної робочої пози під час виконання основних виробничих операцій та можливість зміни пози. Тип робочого крісла обирають залежно від характеру та тривалості роботи.[20-21]

Раціональна поза користувача:

– ступні розташовані на підлозі або на підставці для ніг;

– стегна зорієнтовані у горизонтальній площині;

– верхні ділянки рук вертикальні;

– кут ліктьового суглоба у межах 70-90°;

– зап’ястя зігнуті під кутом не більше ніж 20°;

– нахил голови у межах 15-20°, а часті її повороти виключені.

Вимоги до ПК включають в себе його розташування на столі, розташування розеток та периферійних пристроїв.

ПК встановлювати на рівній твердій поверхні (столі). Не дозволено встановлювати ПК та іншу техніку на хитких підставках чи на похилій поверхні. Також не встановлювати впритул до стіни, перегородки тощо. Не допускати загородження вентиляційних отворів ПК сторонніми предметами.

Розетка біля ПК має бути в доступному місці, щоб в аварійних випадках можна було своєчасно його відімкнути. Не рекомендовано використовувати подовжувачі. ПК під'єднувати до електромережі лише за допомогою справних штепсельних з'єднань та електророзеток заводського виробництва.

Під час переміщення ПК, периферійних пристроїв витягти вилку живлення з розетки. Не допускати ушкодження чи модифікування шнура живлення. Заборонено ставити важкі речі на шнур живлення, тягнути чи надмірно перегинати його, скручувати та перев'язувати шнур живлення вузлом.

Також є ряд вимог щодо монітора і клавіатури (кут нахилу, відстань до користувача).

Кут нахилу екрана монітора або ноутбука по відношенню до вертикалі повинен складати 10-15 градусів. Кут зору екрана повинен бути прямим і становити 90 градусів. Монітор встановлюють так, щоб відстань від поверхні екрана до очей користувача була 600-700 мм залежно від розміру екрана.

Клавіатуру розміщують на робочому або окремому столі на відстані 100-300 мм від краю з боку користувача. Положення клавіатури та кут її нахилу залежить від побажання користувача (як правило, в межах 5-15°). Не допускається хитання клавіатури [22-25].

Про всі виявлені під час роботи несправності обладнання необхідно доповісти керівнику, у випадку поломки необхідно припинити роботу до усунення аварійних обставин. При виявленні можливої небезпеки, попередити оточуючих та негайно повідомити керівнику .

Про нещасний випадок очевидець, працівник, який його виявив, або сам потерпілий повинні доповісти безпосередньо керівникові установи і вжити заходів з надання медичної допомоги.

Отже, необхідно дотримуватись усіх стандартів ДСТУ щодо робочих приміщень та розміщених у них предметах. Регулятно потрібно проводити інструктажі та медичні огляди.

4.3 Висновок до четвертого розділу

У четвертому розділі розглянуто працездатність людини-оператора. Вона залежить як від зовнішніх факторів, так і від внутрішнього стану. В процесі однієї трудової зміни можна виділити 4 фази працездатності. Під час першої фази працівник адаптується до умов праці. Друга фаза характеризується найвищою продуктивністю. На третій фазі починається розвиватися втома. А четверта – характеризується чітко вираженим зниженням якості роботи.

Також описано загальні вимоги безпеки з охорони праці для користувачів ПК. Серед основних вимог це проведення інструктажів, встановлення правильного освітлення та робочого місця. Також є ряд вимог щодо розташування самого ПК та периферійних пристроїв.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи розроблено Telegram-бот для формування самодисципліни користувачів, який забезпечує організацію повсякденних завдань, контроль їх виконання та підвищення особистої продуктивності. Проведене дослідження підтвердило актуальність використання сучасних інформаційних технологій для автоматизації процесів планування та розвитку корисних звичок.

У першому розділі роботи проведено аналіз предметної області, досліджено особливості використання Telegram-ботів у сфері самоорганізації та визначено основні проблеми, які виникають під час планування особистої діяльності. Сформовано функціональні та нефункціональні вимоги до програмного комплексу, визначено основних акторів і сценарії використання системи, а також обґрунтовано вибір середовища розробки та сучасних програмних технологій, зокрема Node.js, JavaScript, Telegram Bot API та MongoDB.

У другому розділі розглянуто проєктну частину розробки програмного комплексу. Сформовано структуру Telegram-бота, спроєктовано його архітектуру, модель взаємодії компонентів та структуру бази даних. Також розроблено алгоритм функціонування програмного комплексу, який визначає послідовність обробки запитів користувачів та формування відповідей. Запропоновані проєктні рішення забезпечують модульність, масштабованість і можливість подальшого розширення функціональних можливостей системи.

У третьому розділі реалізовано програмний комплекс відповідно до сформованих вимог. Розроблено інтерфейс взаємодії користувача з Telegram-ботом, реалізовано функції створення, редагування та видалення завдань, механізм автоматичних нагадувань, збереження інформації у базі даних та перегляд статистики виконання поставлених цілей. Проведене тестування підтвердило працездатність програмного комплексу, правильність роботи його основних функцій та відповідність поставленим вимогам. Також визначено перспективи подальшої модернізації системи, серед яких інтеграція з

календарями, розширення аналітичних можливостей, впровадження системи досягнень та використання елементів штучного інтелекту для персоналізації рекомендацій.

У четвертому розділі було розглянуто питання безпеки життєдіяльності та охорони праці під час роботи користувачів із комп'ютерною технікою. Проаналізовано фактори, що впливають на працездатність людини-оператора, наведено основні вимоги безпеки при роботі з персональним комп'ютером та рекомендації щодо організації безпечного робочого місця.

Отже, поставлена мета кваліфікаційної роботи була повністю досягнута, а всі визначені завдання успішно виконані. Розроблений Telegram-бот є сучасним програмним продуктом, який може використовуватися як ефективний інструмент для формування самодисципліни, покращення навичок планування часу та підвищення особистої ефективності користувачів. Отримані результати можуть бути використані для подальшого розвитку програмного комплексу та його практичного впровадження.

ПЕРЕЛІК ДЖЕРЕЛ

1 Database, SQL Analytics, SQL Certification, SQL Data Analysis. Independently Published, 2019.

2 Flanagan D. JavaScript: The Definitive Guide. 7th ed. Sebastopol : O'Reilly Media, 2020. 704 p.

3 Node.js Official Documentation. URL: https://nodejs.org/en/docs/

4 Telegram Bot API Documentation. URL: https://core.telegram.org/bots/api

5 Telegraf Documentation. URL: https://telegraf.js.org/

6 Mongoose Documentation. URL: https://mongoosejs.com/docs/

7 MongoDB Documentation. URL: https://www.mongodb.com/docs/

8 Tilkov S., Vinoski S. Node.js in Action. 2nd ed. Manning Publications, 2018. 432 p.

9 Freeman E., Robson E. Learning JavaScript Programming. O'Reilly Media, 2023. 690 p.

10 MDN Web Docs. JavaScript Guide. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript

11 Express.js Documentation. URL: https://expressjs.com/

12 REST API Design Guidelines. Microsoft Learn. URL: https://learn.microsoft.com/en-us/azure/architecture/best-practices/api-design

13 Git Documentation. URL: https://git-scm.com/doc

14 GitHub Documentation. URL: https://docs.github.com/

15 Visual Studio Code Documentation. URL: https://code.visualstudio.com/docs

16 npm Documentation. URL: https://docs.npmjs.com/

17 Бедрій Я. І. Безпека життєдіяльності : навчальний посібник. Київ : Кондор, 2009. 286 с.

18 Грибан В. Г., Негодченко О. В. Охорона праці. Київ : Центр учбової літератури, 2009. 280 с.

19 ДСТУ 7234:2011. Дизайн і ергономіка. Обладнання виробниче. Загальні вимоги дизайну та ергономіки.

20 ДСТУ 7299:2013. Дизайн і ергономіка. Робоче місце оператора. Загальні вимоги ергономіки.

21 Інструкція з охорони праці при роботі з комп'ютером та оргтехнікою. URL: https://osvita-docs.com/node/41

22 Мартін Р. К. Чистий код. Створення, аналіз і рефакторинг : пер. з англ. Київ : Фабула, 2019. 448 с.

23 Фаулер М. Рефакторинг. Поліпшення наявного коду : пер. з англ. Київ: Фабула, 2020. 448 с.

24 Шилдт Г. JavaScript. Повний довідник : пер. з англ. Київ : Діалектика, 2022. 800 с.

25 Фрімен Е., Робсон Е. Head First. Патерни проєктування : пер. з англ. Київ : Фабула, 2021. 672 с.

26 Сілбершац А., Корт Г., Сударшан С. Системи баз даних. Повний курс : пер. з англ. Київ : Вільямс, 2021. 1376 с.

27 Грибан В.Г., Негодченко О.В. Охорона праці. – К.: Центр учбової літератури, 2009, 209 с.

28 ДСТУ 7234:2011. Дизайн і ергономіка. Обладнання виробниче. Загальні вимоги дизайну та ергономіки. Чинний від 2011-08-01. Вид. офіц. Укр. НДІ дизайну та ергономіки НАУ.

29 ДСТУ 7299:2013. Дизайн і ергономіка. Робоче місце оператора. Взаємне розташування елементів робочого місця. Загальні вимоги ергономіки. Чининй від 2014-01-01. Вид. офіц.

30 Інструкція з охорони праці при роботі з комп'ютером, принтером, ксероксом та іншою оргтехнікою | Інструкції для навчальних закладів України. Інструкції для навчальних закладів України | Інструкції з охорони праці, техніки безпеки і пожежної безпеки. URL: https://osvita-docs.com/node/41

31 Готович В., Попович В. Дослідження та розробка AI-асистента на основі моделі Mistral для середовища університету. Матеріали XIII науково-технічної конференції «Інформаційні моделі, системи та технології», 17-18 грудня 2025 року. – Т. : ТНТУ, 2025. – С. 40. – (Математичне моделювання)

32 Гайдар А., Готович В. Розробка платформи для перевірки знань шляхом тестування // Матеріали IX науково-технічної конференції „Інформаційні моделі, системи та технології“. – ТНТУ, 2021. – С. 37

33 Козак В. І., Готович В. А. Дослідження варіантів проектування інтерфейсу користувача в інформаційних інтерактивних аналітичних панелях // Матеріали XII Міжнародної науково-практичної конференції молодих учених та студентів „Актуальні задачі сучасних технологій “. – ФОП Паляниця В. А., 2023. – С. 385–386

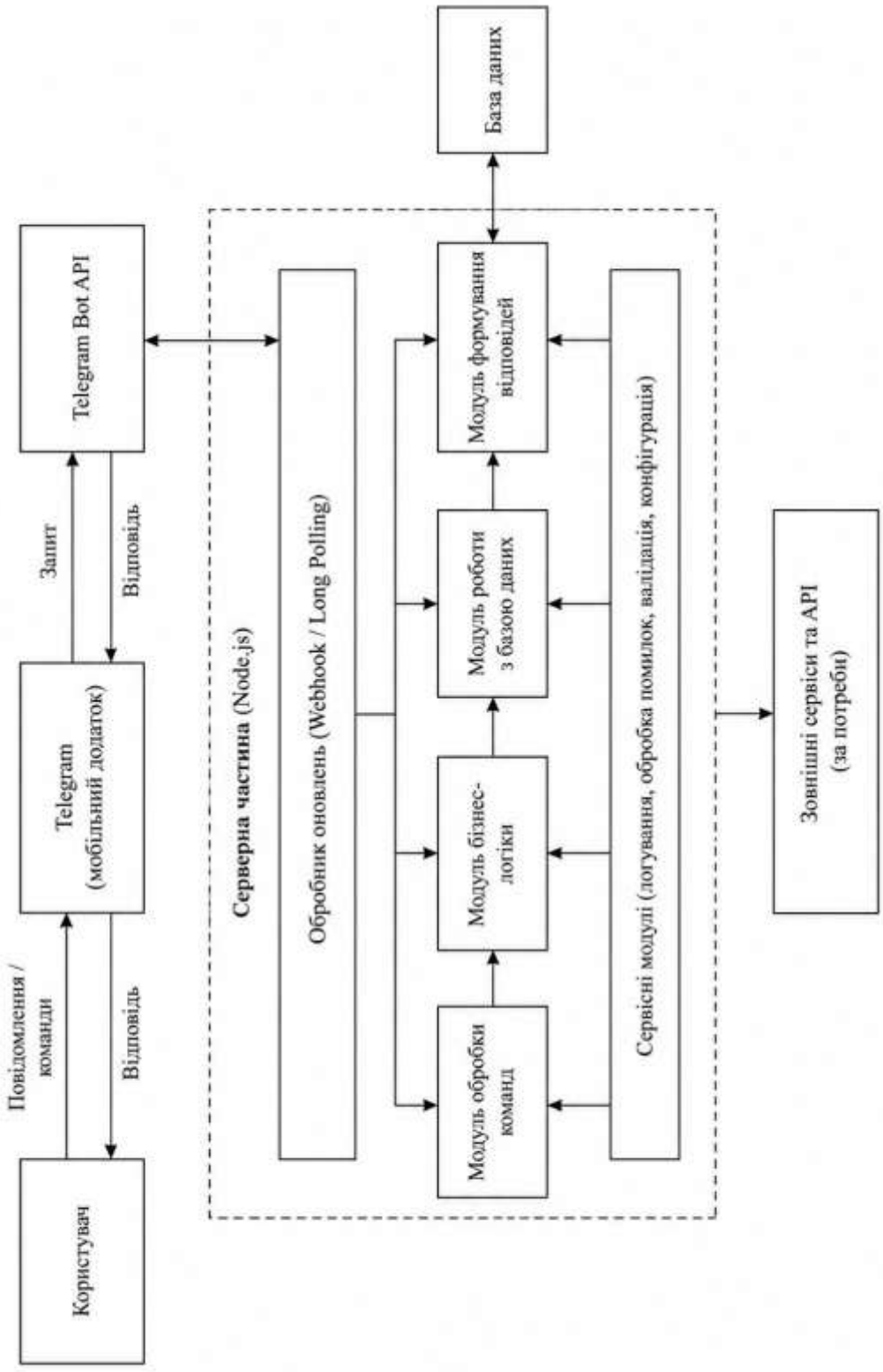
34 Методичні вказівки до виконання лабораторних робіт з дисципліни «Програмування для мобільних пристроїв» для студентів денної форми навчання спеціальності 126 «Інформаційні системи та технології» / Укладачі: Готович В. А., Михайлович Т. В. – Тернопіль : Тернопільський національний технічний університет імені Івана Пулюя, 2020. – 100 с.

35 Конспект лекцій з дисципліни «Програмування для мобільних пристроїв» для студентів денної форми навчання спеціальності 126 «Інформаційні системи та технології / Укладачі: Готович В. А., Михайлович Т. В. – Тернопіль : Тернопільський національний технічний університет імені Івана Пулюя, 2020. – 216 с.

ДОДАТКИ

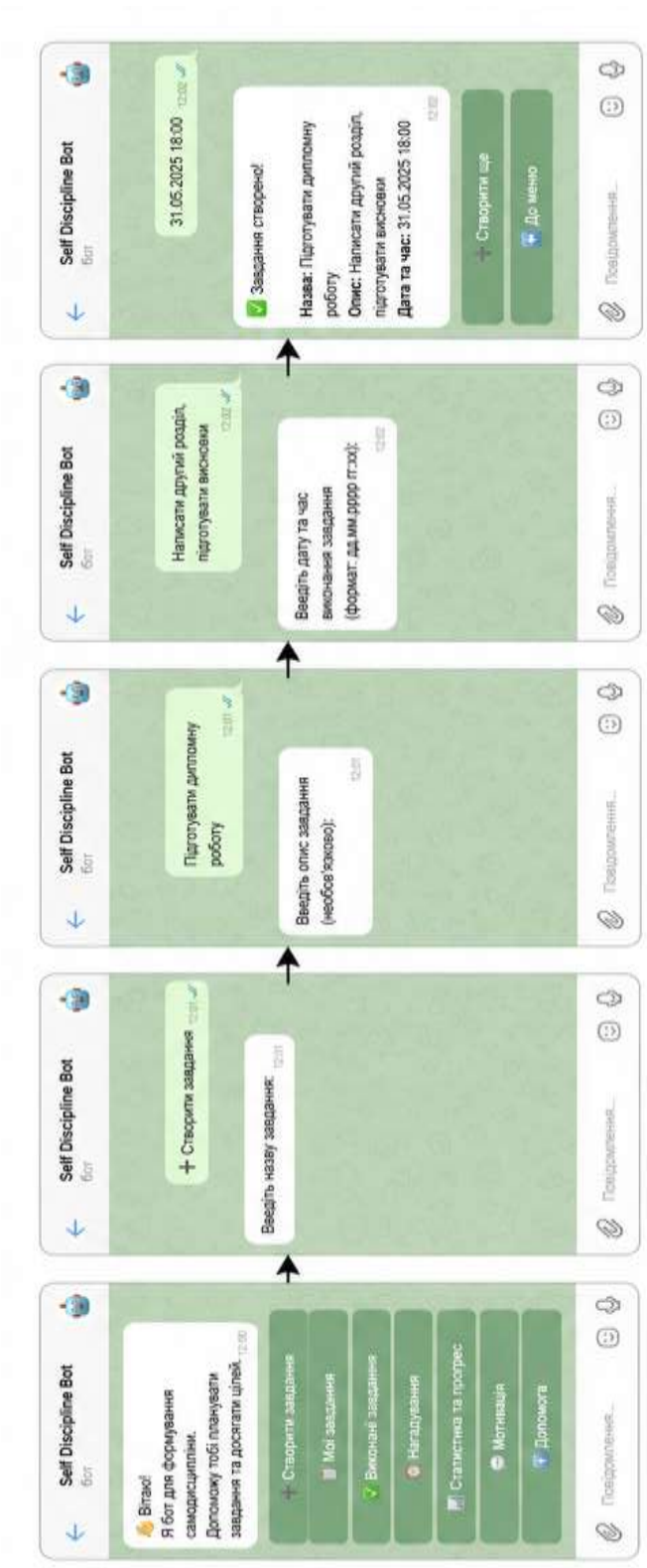
Додаток А

Блок модель завдання



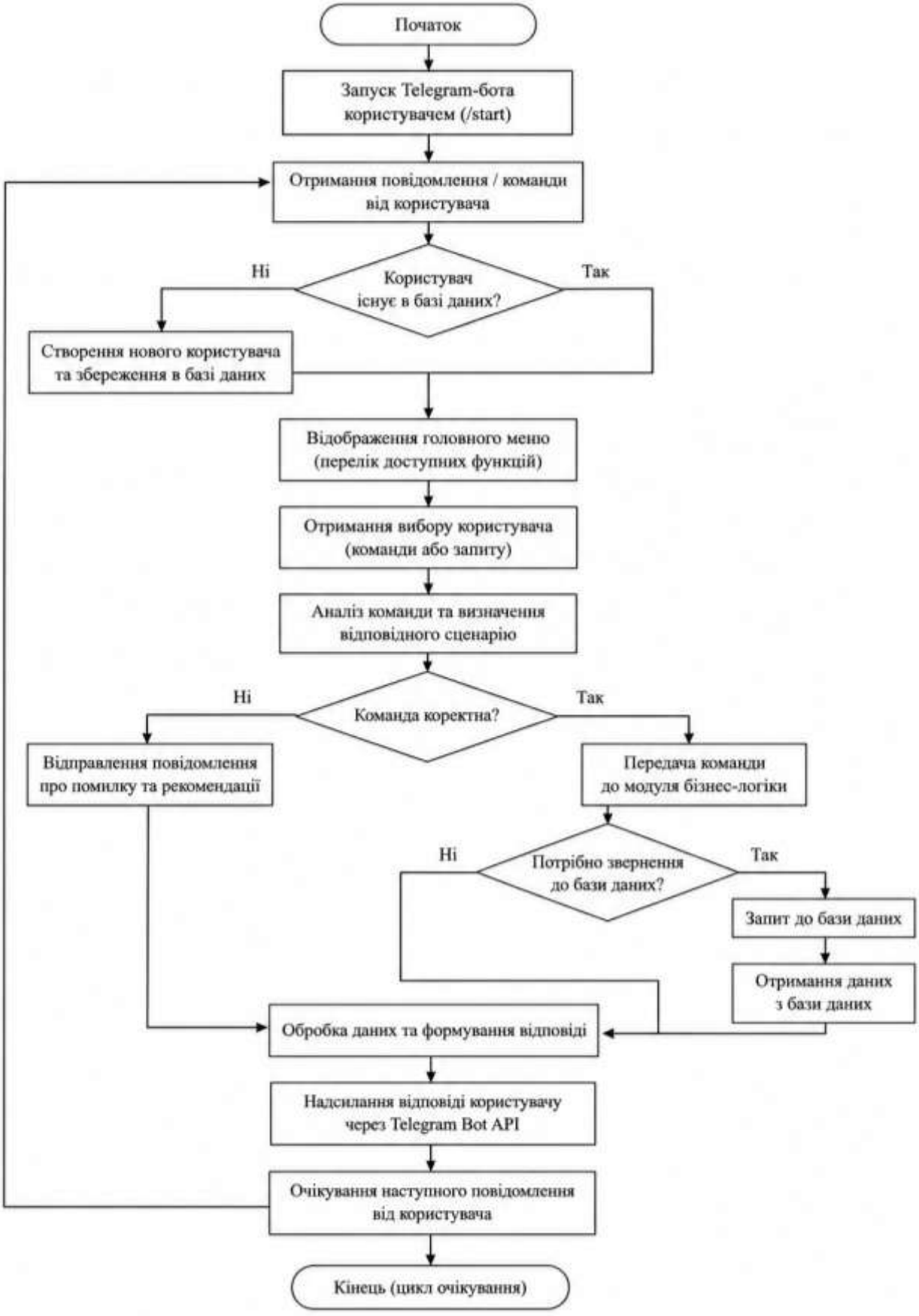
Додаток Б

Приклад взаємодії користувача з Телегерам-ботом



Додаток В

Схема алгоритму функціонування Telegram-бота



Додаток Д

Програмний код Telegram-бота

```
const { Telegraf, Markup, session } = require('telegraf');
const mongoose = require('mongoose');
const cron = require('node-cron');
require('dotenv').config();

const User = require('./models/User');
const Task = require('./models/Task');

if (!process.env.BOT_TOKEN) {
  console.error('Помилка: додайте BOT_TOKEN у файл .env');
  process.exit(1);
}

const bot = new Telegraf(process.env.BOT_TOKEN);

const mainMenu = Markup.keyboard([
  ['➕ Створити завдання'],
  ['📋 Мої завдання', '✅ Виконані завдання'],
  ['📊 Статистика', '❓ Допомога'],
]).resize();

bot.use(session());

async function registerUser(ctx) {
  await User.findOneAndUpdate(
    { telegramId: ctx.from.id },
    {
      telegramId: ctx.from.id,
      firstName: ctx.from.first_name || '',
      username: ctx.from.username || '',
    },
    { upsert: true, new: true }
  );
}

function formatDate(date) {
  return new Date(date).toLocaleString('uk-UA', {
    day: '2-digit',
    month: '2-digit',
    year: 'numeric',
    hour: '2-digit',
    minute: '2-digit',
  });
}

function parseDate(text) {
  const match =
text.trim().match(/^(\d{2})\.(\d{2})\.(\d{4})\s+(\d{2}):(\d{2})$/)
;
```

```
  if (!match) {
    return null;
  }

  const [, day, month, year, hour, minute] = match;
  const date = new Date(
    Number(year),
    Number(month) - 1,
    Number(day),
    Number(hour),
    Number(minute)
  );

  if (Number.isNaN(date.getTime())) {
    return null;
  }

  return date;
}

bot.start(async (ctx) => {
  await registerUser(ctx);
  ctx.session = {};

  await ctx.reply(
    'Вітаю! Я Telegram-бот для формування самодисципліни.\n\n' +
      'Я допоможу створювати завдання, отримувати нагадування та
контролювати власну продуктивність.',
    mainMenu
  );
});

bot.hears('☐ Допомога', async (ctx) => {
  await ctx.reply(
    'Як користуватись ботом:\n\n' +
      '✚ Створити завдання — додати нову справу.\n' +
      '☐ Мої завдання — переглянути активні завдання.\n' +
      '✓ Виконані завдання — переглянути завершені завдання.\n' +
      '☐ Статистика — переглянути прогрес.\n\n' +
      'Дата вводиться у форматі: 10.06.2026 18:00',
    mainMenu
  );
});

bot.hears('✚ Створити завдання', async (ctx) => {
  await registerUser(ctx);

  ctx.session = {
    action: 'create_task',
    step: 'title',
  };

  await ctx.reply('Введіть назву завдання:');
```

```
});

bot.hears('☐ Мої завдання', async (ctx) => {
  const tasks = await Task.find({
    userId: ctx.from.id,
    status: 'active',
  }).sort({ dueDate: 1 });

  if (tasks.length === 0) {
    return ctx.reply('У вас поки немає активних завдань.',
mainMenu);
  }

  for (const task of tasks) {
    await ctx.reply(
      `☐ ${task.title}\n\n` +
        `Опис: ${task.description || 'без опису'}\n` +
        `Дата: ${formatDate(task.dueDate)}\n` +
        `Статус: активне`,
      Markup.inlineKeyboard([
        Markup.button.callback('✓ Виконано',
`complete_${task._id}`),
        Markup.button.callback('☐ Видалити',
`delete_${task._id}`),
      ])
    );
  }
});

bot.hears('✓ Виконані завдання', async (ctx) => {
  const tasks = await Task.find({
    userId: ctx.from.id,
    status: 'completed',
  }).sort({ updatedAt: -1 });

  if (tasks.length === 0) {
    return ctx.reply('У вас поки немає виконаних завдань.',
mainMenu);
  }

  let message = '✓ Виконані завдання:\n\n';

  tasks.forEach((task, index) => {
    message += `${index + 1}. ${task.title}\n`;
  });

  await ctx.reply(message, mainMenu);
});

bot.hears('☐ Статистика', async (ctx) => {
  const userId = ctx.from.id;

  const total = await Task.countDocuments({ userId });
```

```
  const active = await Task.countDocuments({ userId, status:
'active' });
  const completed = await Task.countDocuments({ userId, status:
'completed' });

  const progress = total === 0 ? 0 : Math.round((completed / total)
* 100);
  await ctx.reply(
    '☐ Ваша статистика:\n\n' +
      `Усього завдань: ${total}\n` +
      `Активні: ${active}\n` +
      `Виконані: ${completed}\n` +
      `Прогрес: ${progress}%`,
    mainMenu
  );
});

bot.action(/complete_(.+)/, async (ctx) => {
  const taskId = ctx.match[1];

  const task = await Task.findOneAndUpdate(
    {
      _id: taskId,
      userId: ctx.from.id,
    },
    {
      status: 'completed',
    },
    { new: true }
  );

  if (!task) {
    return ctx.answerCbQuery('Завдання не знайдено');
  }

  await ctx.answerCbQuery('Завдання виконано');
  await ctx.editMessageText(`✅ Завдання виконано: ${task.title}`);
});

bot.action(/delete_(.+)/, async (ctx) => {
  const taskId = ctx.match[1];

  const task = await Task.findOneAndDelete({
    _id: taskId,
    userId: ctx.from.id,
  });

  if (!task) {
    return ctx.answerCbQuery('Завдання не знайдено');
  }
  await ctx.answerCbQuery('Завдання видалено');
  await ctx.editMessageText(`☐ Завдання видалено: ${task.title}`);
});
```

```
bot.on('text', async (ctx) => {
  if (!ctx.session || ctx.session.action !== 'create_task') {
    return ctx.reply('Оберіть дію з меню.', mainMenu);
  }

  const text = ctx.message.text.trim();

  if (ctx.session.step === 'title') {
    ctx.session.title = text;
    ctx.session.step = 'description';

    return ctx.reply('Введіть опис завдання або напишіть "-", якщо
опис не потрібен:');
  }

  if (ctx.session.step === 'description') {
    ctx.session.description = text === '-' ? '' : text;
    ctx.session.step = 'date';

    return ctx.reply('Введіть дату і час виконання у форматі:
10.06.2026 18:00');
  }

  if (ctx.session.step === 'date') {
    const dueDate = parseDate(text);

    if (!dueDate) {
      return ctx.reply('Неправильний формат дати. Введіть так:
10.06.2026 18:00');
    }
    await Task.create({
      userId: ctx.from.id,
      title: ctx.session.title,
      description: ctx.session.description,
      dueDate,
      status: 'active',
      reminded: false,
    });
    ctx.session = {};
    return ctx.reply('✅ Завдання успішно створено!', mainMenu);
  }
});
cron.schedule('* * * * *', async () => {
  const now = new Date();

  const tasks = await Task.find({
    status: 'active',
    reminded: false,
    dueDate: { $lte: now },
  });

  for (const task of tasks) {
    try {
      await bot.telegram.sendMessage(
```

```
        task.userId,
        `□ Нагадування!\n\nПотрібно виконати завдання:
${task.title}`
      );
      task.reminded = true;
      await task.save();
    } catch (error) {
      console.log('Помилка надсилання нагадування:',
error.message);
    }
  }
});
async function startBot() {
  try {
    await mongoose.connect(
      process.env.MONGO_URL ||
'mongodb://127.0.0.1:27017/self_discipline_bot'
    );

    console.log('Базу даних підключено');
    await bot.launch();
    console.log('Telegram-бот для формування самодисципліни
запущено');
  } catch (error) {
    console.log('Помилка запуску:', error.message);
  }
}

startBot();

process.once('SIGINT', () => bot.stop('SIGINT'));
process.once('SIGTERM', () => bot.stop('SIGTERM'));
```