Форма № 24

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій
(повне найменування факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

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

на тему РОЗРОБЛЕННЯ ТА ДОСЛІДЖЕННЯ АЛГОРИТМІВ ВИЗНАЧЕННЯ ЕМОЦІЙНОГО СТАНУ КОРИСТУВАЧА
DEVELOPMENT AND RESEARCH OF ALGORITHMS FOR HUMAN EMOTION RECOGNITION

Виконав(ла): студент(ка) 4 курсу, групи КНТ-222
Спеціальності 122 Комп'ютерні науки
(код і найменування спеціальності)
Освітня програма (спеціалізація)
Комп'ютерні науки

ШАМРАЙ П.О.
(ПРІЗВИЩЕ та ініціали)
Керівник ГЛАДКОВА О.М.
(ПРІЗВИЩЕ та ініціали)
Рецензент ГОЛУБ Т.В.
(ПРІЗВИЩЕ та ініціали)

2026

Форма № 25

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 122 Комп'ютерні науки
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
_____Сергій СУББОТІН
"____" _____2026 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ШАМРАЙ Поліни Олегівни
(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розроблення та дослідження алгоритмів визначення емоційного стану користувача. Development and Research of Algorithms for Human Emotion Recognition
керівник проєкту (роботи) к.т.н., доцент, ГЛАДКОВА Ольга Миколаївна ,
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)
затверджені наказом закладу вищої освіти від " 07 " квітня 2026 року № 139
2. Строк подання студентом проєкту (роботи) 10 червня 2026 року
3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали, методи та технології розроблення. 3. Проєктування та розроблення Telegram-бота Mind Balance. 4. Тестування та дослідження системи.

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

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ГЛАДКОВА О.М., доцент		
Нормоконтроль	ДЕЙНЕГА Л.Ю., ст. викладач		

7. Дата видачі завдання « 20 » квітня 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій	2 тиждень	Розділ 2
	розробки.		
4	Розробка архітектури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділ 3, 4
6	Тестування та експериментальне дослідження	5 тиждень	Розділ 4
	програмного забезпечення.		
7	Оформлення пояснювальної записки та документів	6 тиждень	Додатки
	до неї.		
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка) _____ Поліна ШАМРАЙ
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи) _____ Ольга ГЛАДКОВА
(підпис) (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи бакалавра: 59 с., 6 табл., 23 рис., 2 додатки, 13 джерела.

TELEGRAM-БОТ, PYTHON, AIOGRAM, АНАЛІЗ ЕМОЦІЙ, ОБРОБКА ПРИРОДНОЇ МОВИ, МОНІТОРИНГ ЕМОЦІЙНОГО СТАНУ.

Об’єкт дослідження – моделі, методи та алгоритми аналізу текстових даних і побудови програмних систем моніторингу емоційного стану користувача.

Предмет дослідження – методи та програмні засоби визначення емоційного стану користувача на основі текстових повідомлень у Telegram-боті.

Мета роботи – підвищення швидкості та точності визначення емоційного стану користувача шляхом розроблення Telegram-бота з використанням алгоритмів аналізу тексту та автоматизованої обробки повідомлень.

Матеріали, методи та технічні засоби: мова програмування Python, бібліотека Aiogram, технології асинхронного програмування asyncio, Telegram Bot API, бібліотека Matplotlib, методи sentiment analysis та словникового аналізу.

Результати. Розроблено Telegram-бота Mind Balance для аналізу емоційного стану користувача. Реалізовано модулі аналізу тексту, опитування, статистики та побудови графіків. Проведене тестування підтвердило коректність роботи системи.

Висновки. У роботі розроблено Telegram-бота для моніторингу емоційного стану користувача. Використання алгоритмів sentiment analysis дозволило скоротити час аналізу повідомлень та підвищити точність визначення емоційного стану.

Галузь використання – системи цифрового моніторингу емоційного стану, Telegram-сервіси, інформаційні системи підтримки користувачів.

ABSTRACT

Explanatory note to the bachelor thesis: 59 pages, 6 tables, 23 figures, 2 appendixes, 13 references.

TELEGRAM BOT, PYTHON, AIOGRAM, EMOTION ANALYSIS, NATURAL LANGUAGE PROCESSING, EMOTIONAL STATE MONITORING.

The object of the research is models, methods, and algorithms for text data analysis and the development of software systems for monitoring the user's emotional.

The subject of the research is methods and software tools for determining the user's emotional state based on text messages in a Telegram bot.

The purpose of the work is to increase the speed and accuracy of determining the user's emotional state by developing a Telegram bot using text analysis algorithms and automated message processing.

Materials, methods, and technical tools: Python programming language, Aiogram library, asyncio asynchronous programming technologies, Telegram Bot API, Matplotlib library, sentiment analysis methods, and dictionary-based text analysis.

Results. The Mind Balance Telegram bot for analyzing the user's emotional state was developed. Modules for text analysis, surveys, statistics, and graph generation were implemented. The conducted testing confirmed the correctness of the system operation.

Conclusions. In this work, a Telegram bot for monitoring the user's emotional state was developed. The use of sentiment analysis algorithms made it possible to reduce message analysis time and improve the accuracy of emotional state determination.

Field of application – digital emotional state monitoring systems, Telegram services, and user support information systems.

ЗМІСТ

С.

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API – Application Programming Interface;

Bot API – інтерфейс прикладного програмування Telegram для створення ботів;

FSM – Finite State Machine, скінченний автомат станів;

NLP – Natural Language Processing, обробка природної мови;

Sentiment Analysis – автоматизоване визначення емоційної тональності тексту;

UML – Unified Modeling Language, уніфікована мова моделювання;

БД – база даних;

ПЗ – програмне забезпечення.

ВСТУП

У сучасному інформаційному суспільстві значна частина повсякденної комунікації відбувається у цифровому середовищі. Користувачі взаємодіють із месенджерами, освітніми платформами та соціальними сервісами, створюючи текстові повідомлення, які можуть відображати їхній поточний настрій, рівень енергії та загальний емоційний фон. Тому автоматизований аналіз емоційного стану користувача є актуальним напрямом сучасних комп'ютерних наук.

Особливого значення набувають прості інструменти самоспостереження. На відміну від складних інформаційних систем, чат-боти працюють у звичному для користувача середовищі та не потребують встановлення окремого клієнтського програмного забезпечення. Telegram-бот, який веде діалог із користувачем, фіксує оцінку настрою, зберігає нотатки та формує статистику, є зручною моделлю цифрового щоденника.

Метою дипломної роботи є розроблення та дослідження Telegram-бота Mind Balance для аналізу емоційного стану користувача на основі числового самооцінювання, текстових нотаток і результатів опитування. Для досягнення мети необхідно проаналізувати предметну область, дослідити методи sentiment analysis, спроєктувати архітектуру системи, реалізувати програмні модулі та виконати тестування.

Об'єктом дослідження є процес автоматизованого аналізу емоційного стану користувача. Предметом дослідження є алгоритми та програмні засоби оцінювання емоційного стану на основі Telegram-бота. Практична цінність полягає у створенні навчального прототипу, який демонструє поєднання чат-інтерфейсу, обробки тексту, статистичних розрахунків і візуалізації даних.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальність цифрового моніторингу емоційного стану

Емоційний стан людини є складною характеристикою, яка формується під впливом зовнішніх подій, інформаційного навантаження, соціальної взаємодії, навчального або професійного середовища. У цифрову епоху значна частина цих факторів відображається у взаємодії користувача з інформаційними системами.

Моніторинг емоційного стану може мати різні рівні складності. Найпростішим підходом є самооцінювання, коли користувач самостійно обирає числове значення настрою за шкалою. Такий метод є прозорим і зрозумілим, але залежить від суб’єктивного сприйняття [1].

Іншим підходом є аналіз текстових повідомлень. Повсякденні нотатки можуть містити слова, що характеризують позитивні або негативні емоції. У межах навчального проєкту доцільно використовувати словниковий метод, який порівнює текст із визначеними списками слів.

Важливим компонентом є візуалізація. Графік настрою, середнє значення, найкращий і найважчий день, а також порівняння першої і другої половини записів дають змогу сформувати цілісне уявлення про динаміку стану.

Емоційний стан людини є складною характеристикою, яка формується під впливом зовнішніх подій, інформаційного навантаження, соціальної взаємодії, навчального або професійного середовища. У цифрову епоху значна частина цих факторів відображається у взаємодії користувача з інформаційними системами.

1.2 Аналіз сучасних програмних рішень

Сучасні програмні рішення для моніторингу емоцій можна поділити на щоденники настрою, чат-боти підтримки, сервіси медитації та системи аналітики

тексту. Кожен тип має власні переваги та обмеження. У таблиці 1.1 наведено порівняння типів систем.

Мобільні щоденники настрою орієнтовані на регулярне введення коротких оцінок стану. Їхньою перевагою є простий інтерфейс і календарна структура, проте вони часто не аналізують зміст текстових нотаток.

Чат-боти використовують діалогову форму спілкування. Вони можуть ставити уточнювальні питання та формувати короткі відповіді. Для бакалаврської роботи важливо зосередитися не на терапевтичному ефекті, а на алгоритмічній частині: фіксації відповідей, розрахунку показників і візуалізації.

Системи автоматизованої аналітики тексту застосовують словникові методи, статистичні алгоритми або моделі машинного навчання. Для прототипу Mind Balance обрано словниковий підхід, оскільки він прозорий, швидкий і не потребує зовнішніх сервісів [2].

Таблиця 1.1 – Порівняння типів систем

Тип системи	Основні функції	Переваги	Недоліки
Щоденник настрою	Оцінка стану, календар, нотатки	Простота	Обмежений аналіз тексту
Чат-бот	Діалог, питання, підказки	Інтерактивність	Потрібна перевірка якості
Аналітика тексту	Sentiment analysis	Автоматизація	Потреба у словниках або моделях

1.3 Методи обробки природної мови та sentiment analysis

Обробка природної мови вивчає методи автоматизованого аналізу тексту. У контексті оцінювання емоційного стану особливе значення має sentiment analysis - визначення емоційної тональності повідомлення [3].

Словниковий метод передбачає створення наборів слів, які відповідають певним емоційним категоріям. Наприклад, слова «чудово», «спокійно», «радість» відносяться до позитивних, а «сумно», «тривога», «втома» - до негативних [4].

Перевага словникового методу полягає у пояснюваності. Користувач і розробник можуть зрозуміти, чому система віднесла повідомлення до позитивного, негативного або змішаного стану.

Недоліком є обмежена здатність враховувати контекст, сарказм, заперечення та складні мовні конструкції. Це обмеження враховано у висновках і запропоновано як напрям подальшого розвитку. У таблиці 1.2 наведено порівняння методів sentiment analysis. і запропоновано як напрям подальшого розвитку.

Таблиця 1.2 – Методи sentiment analysis

Підхід	Суть	Переваги	Доцільність
Словниковий	Підрахунок слів	Прозорість	Висока
Правила	Шаблони й умови	Контрольована логіка	Середня
ML	Навчання на даних	Вища точність	Обмежена
Нейронні мережі	Глибокі моделі	Гнучкість	Надмірна складність

1.4 Постановка задачі дослідження

На основі аналізу предметної області сформульовано задачу розроблення Telegram-бота Mind Balance, який забезпечує інтерактивне введення даних користувачем, їх первинну обробку, збереження у внутрішній структурі та подання результатів.

Функціональні вимоги включають запуск бота командою /start, головне меню, введення настрою від 1 до 10, збереження нотаток, проведення опитування з десяти питань, розрахунок показників, формування статистики, побудову графіка і визначення тренду.

Нефункціональні вимоги передбачають простоту інтерфейсу, швидку реакцію, зрозумілі відповіді, можливість розширення словників і модулів аналізу. Важливим обмеженням є те, що система не повинна позиціонуватися як медичний діагностичний інструмент.

Отже, задача дослідження полягає у створенні програмної системи, що поєднує чат-інтерфейс, алгоритми простого аналізу тексту, обробку результатів опитування, статистичні розрахунки та візуалізацію.

2 МАТЕРІАЛИ, МЕТОДИ ТА ТЕХНОЛОГІЇ РОЗРОБЛЕННЯ

2.1 Вибір мови програмування та бібліотек

Для реалізації Telegram-бота використано Python, оскільки ця мова має високу читабельність, підтримує асинхронне програмування та має розвинену екосистему бібліотек. Python зручний для створення прототипів, обробки тексту та візуалізації даних [5].

Основною бібліотекою є Aiogram [6]. Вона підтримує створення об’єктів Bot і Dispatcher, обробку команд, фільтрацію повідомлень та асинхронну взаємодію з Telegram Bot API [7].

Для побудови графіків використано Matplotlib. Бібліотека дозволяє створювати лінійні графіки, підписувати осі, задавати межі шкали та зберігати результат у PNG-файл [8].

Модуль datetime застосовано для збереження дати та часу записів, а asyncio - для запуску асинхронного циклу роботи бота. У таблиці 2.1 наведено основні технології системи.

Таблиця 2.1 – Технології системи

Технологія	Призначення	Переваги
Python	Основна мова	Простота та екосистема
Aiogram	Telegram-бот	Асинхронність
Matplotlib	Графіки	Гнучка візуалізація
datetime	Дата і час	Коректна фіксація записів

2.2 Telegram Bot API та асинхронна модель Aiogram

Telegram Bot API надає інтерфейс для створення ботів, які отримують повідомлення та надсилають відповіді.

У розробленій системі використано механізм polling, коли застосунок періодично отримує нові події [5].

Асинхронна модель важлива для чат-ботів, оскільки користувачі можуть надсилати повідомлення у довільний момент.

Використання async/await дозволяє не блокувати програму під час очікування мережевих операцій.

Dispatcher отримує повідомлення, перевіряє фільтри та передає керування відповідному обробнику. Натискання кнопки меню активує потрібний сценарій роботи.

Меню створено за допомогою ReplyKeyboardMarkup і KeyboardButton. Це зменшує кількість помилок і робить взаємодію зручною.

2.3 Алгоритм аналізу нотаток користувача

Модуль аналізу нотаток реалізує словниковий алгоритм визначення емоційної тональності тексту. Після вибору пункту «Нотатки дня» бот переводить користувача у режим note.

Текст нотатки переводиться до нижнього регістру. Далі програма перевіряє входження кожного слова з позитивного та негативного словників у повідомлення користувача [9].

Якщо негативних збігів більше, бот формує коментар про переважання негативних емоцій.

Якщо позитивних збігів більше, формується позитивний коментар. Якщо кількість однакова, стан описується як змішаний.

Алгоритм легко розширити: можна додати ваги словам, врахувати заперечення, збільшити словники або використати модель машинного навчання.

2.4 Алгоритм опитування та розрахунку показників

Опитування складається з десяти питань, на які користувач відповідає «так» або «ні». Питання пов’язані з енергією, концентрацією, мотивацією, настроєм, сном і соціальною активністю.

Початкове значення кожного показника дорівнює 10. Відповіді «так» на окремі питання зменшують відповідні показники на задану кількість балів. Мінімальне значення обмежується одиницею [10].

У структурі користувача зберігається номер поточного питання та масив відповідей. Після кожної відповіді крок збільшується, а після останнього питання виконується розрахунок.

Перевагою підходу є простота та прозорість. Кожен показник формується за зрозумілими правилами, а структура опитування може бути змінена у майбутніх версіях. У таблиці 2.2 наведено показники опитування.

Таблиця 2.2 – Показники опитування

Показник	Початкове значення	Фактори зменшення
Енергія	10	втома, сон, важко вставати
Настрій	10	сум, тривога, емоційна важкість
Мотивація	10	концентрація, мотивація
Соціальність	10	уникання людей

2.5 Висновки за розділом 2

У другому розділі було обґрунтовано вибір технологій та методів реалізації програмної системи. Для створення Telegram-бота використано Python, Aiogram та Matplotlib, які забезпечують асинхронну обробку повідомлень, взаємодію з Telegram Bot API та побудову графіків.

Описано алгоритм аналізу текстових нотаток і механізм проведення опитування користувача. Отримані результати підтвердили доцільність використання обраних технологій для реалізації системи моніторингу емоційного стану.

3 ПРОЄКТУВАННЯ ТА РОЗРОБЛЕННЯ TELEGRAM-БОТА MIND BALANCE

3.1 Загальна архітектура програмної системи

Telegram-бот Mind Balance реалізовано як асинхронний застосунок, що складається з інтерфейсного рівня, обробників повідомлень, модуля аналізу, модуля статистики та модуля візуалізації.

Внутрішні дані зберігаються у словнику user_data. Для кожного користувача створюється структура з полями moods, notes, mode, survey_step, survey_answers та analysis.

Основні функції розділено між обробниками. Обробник стартової команди створює профіль, обробник настрою приймає оцінку, обробник нотаток аналізує текст, обробник опитування проводить послідовність питань.

Подієва архітектура дозволяє легко додавати нові функції без повної перебудови програми. На рисунку 3.1 наведено компонентну діаграму системи. перебудови програми.

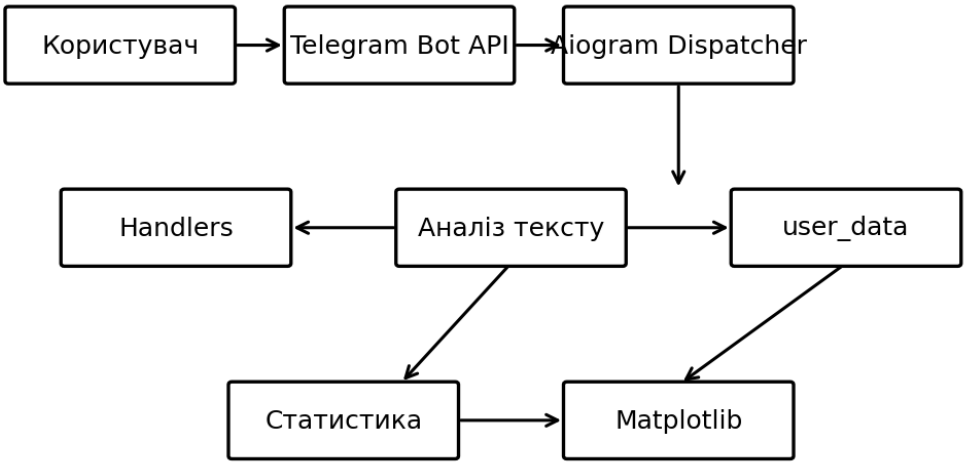

Рисунок 3.1 – Компонентна діаграма

3.2 UML-моделювання системи

Для опису системи використано UML-моделювання. Побудовано діаграму варіантів використання, компонентну діаграму, діаграму активності та діаграму послідовності [11].

Діаграма варіантів використання (рис. 3.2) показує, що користувач може записувати настрій, вести нотатку, проходити опитування, переглядати статистику, формувати графік і аналізувати тренд.

Компонентна діаграма відображає зв’язки між Telegram Bot API, Aiogram Dispatcher, обробниками, модулем аналізу, сховищем даних і Matplotlib.

Діаграма активності описує типовий сценарій: запуск, вибір меню, встановлення режиму, перевірка введення, збереження результату та надсилання відповіді.

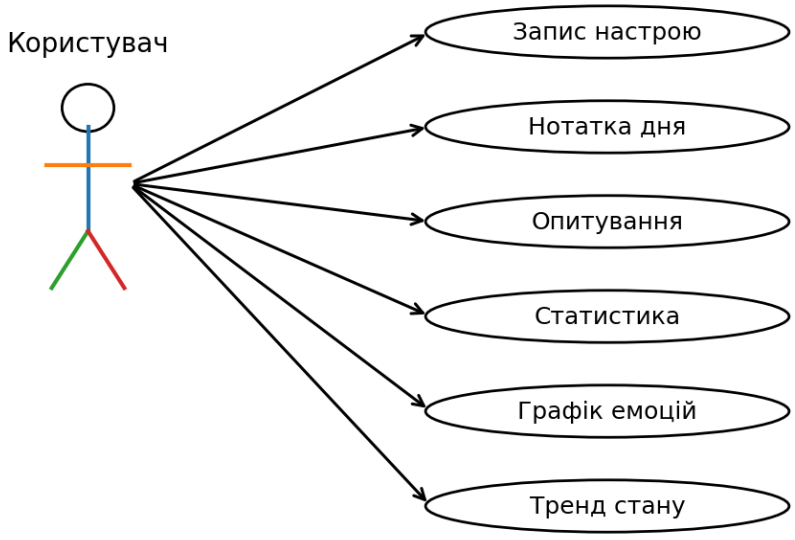

Рисунок 3.2 – UML-діаграма варіантів використання

На рисунку 3.3 представлено діаграму активності Telegram-бота Mind Balance, яка демонструє послідовність виконання основних операцій системи під час взаємодії з користувачем.

Рисунок 3.3 – Діаграма активності

Робота системи починається із запуску Telegram-бота та ініціалізації основних програмних модулів. Після цього користувач отримує доступ до головного меню, у якому може обрати один із доступних режимів роботи: запис емоційного стану, проходження опитування, створення текстової нотатки, перегляд статистики або формування графіка емоцій.

Після вибору певної функції система переходить до етапу введення даних. Якщо користувач обирає фіксацію настрою, бот перевіряє правильність введеної оцінки. У випадку коректного значення інформація зберігається у внутрішньому сховищі даних разом із поточною датою та часом. Якщо введені дані не відповідають встановленому формату, система надсилає повідомлення про помилку та пропонує повторити введення.

Під час проходження опитування бот послідовно відображає запитання та зберігає відповіді користувача. Для текстових нотаток виконується аналіз повідомлення з пошуком емоційно забарвлених слів і визначенням загального емоційного контексту.

Після завершення обробки введеної інформації система формує результат аналізу, оновлює статистичні показники та надсилає користувачу відповідь у вигляді текстового повідомлення або графічної візуалізації. Завершальним етапом є повернення до головного меню для подальшої роботи з ботом.

На рисунку 3.4 наведено діаграму послідовності, яка відображає процес обміну повідомленнями між користувачем, Telegram API, модулем обробки повідомлень та внутрішніми компонентами системи.

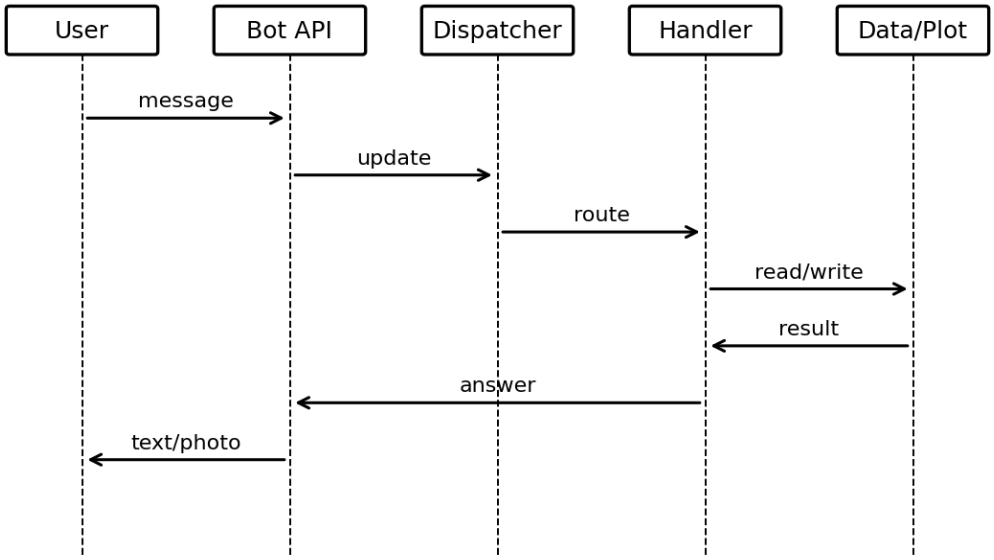

Рисунок 3.4 – Діаграма послідовності

Взаємодія починається з того, що користувач надсилає команду або текстове повідомлення Telegram-боту. Telegram API передає отримані дані до модуля Dispatcher, який визначає тип запиту та активує відповідний обробник.

Після цього обробник аналізує отримане повідомлення та визначає необхідний сценарій роботи. Якщо користувач вводить оцінку настрою, система перевіряє допустимість значення та передає дані до модуля збереження інформації. Якщо користувач надсилає текстову нотатку, повідомлення передається до модуля аналізу емоційного стану, де виконується пошук позитивних, негативних або нейтральних маркерів.

Після завершення аналізу результати передаються до модуля статистики, який обчислює середні показники, визначає зміни емоційного стану та формує узагальнену інформацію. У разі необхідності модуль візуалізації створює графік емоційної активності користувача.

На завершальному етапі сформована відповідь повертається через Telegram API до користувача у вигляді повідомлення, статистики або графіка. Діаграма демонструє логічну взаємодію всіх програмних компонентів та послідовність обробки даних у системі.

На рисунку 3.5 наведено діаграму алгоритму аналізу емоційного стану користувача, яка узагальнює процеси, описані у підрозділах 2.2–2.4. Діаграма демонструє повний цикл обробки інформації — від моменту отримання повідомлення користувача до формування результатів аналізу та статистичних висновків.

На початковому етапі користувач надсилає текстове повідомлення або оцінку емоційного стану через Telegram-бот. Отримані дані передаються до модуля обробки повідомлень, де виконується перевірка коректності введення та визначення типу даних.

Далі система переходить до етапу попередньої обробки тексту. На цьому етапі виконується очищення повідомлення від зайвих символів, перетворення тексту до єдиного формату та підготовка до аналізу. Після цього запускається модуль емоційного аналізу, який порівнює слова повідомлення зі словниками позитивних і негативних емоційних маркерів.

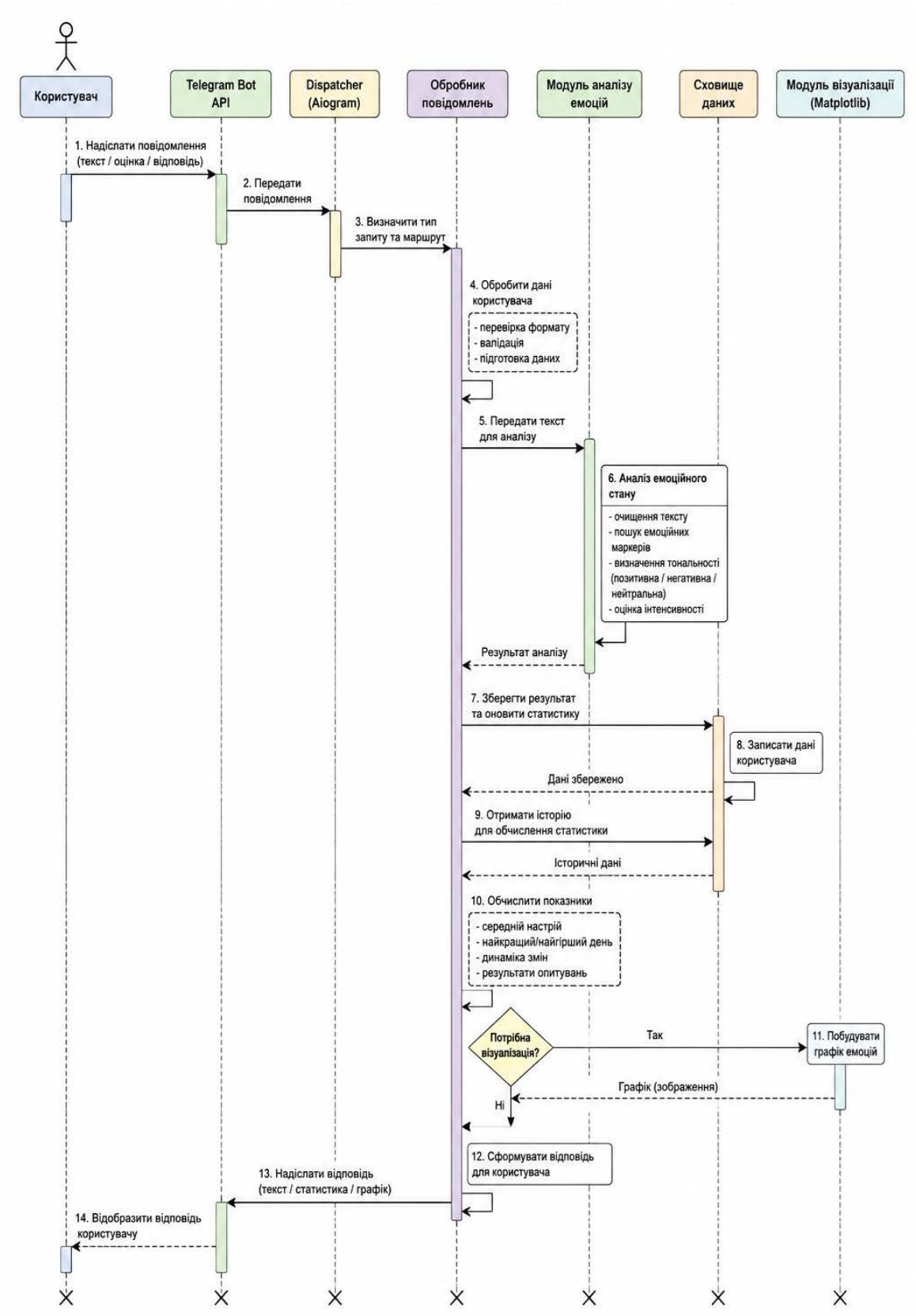

Рисунок 3.5 – Діаграма алгоритму аналізу емоційного стану

У результаті аналізу система визначає емоційне забарвлення повідомлення: позитивне, негативне або змішане. Додатково обчислюється рівень емоційної активності та формуються статистичні показники на основі попередніх записів користувача.

Після завершення аналізу результати зберігаються у внутрішньому сховищі даних, а користувачу надсилається відповідь із коротким описом поточного емоційного стану, статистикою або графічною візуалізацією змін настрою. Завершальним етапом алгоритму є оновлення історії спостережень для подальшого аналізу динаміки емоційного стану.

3.3 Реалізація основних функціональних модулів

Модуль запуску створює об’єкт Bot, ініціалізує Dispatcher і запускає polling. Асинхронна функція main забезпечує постійне очікування нових повідомлень.

Модуль меню містить кнопки для всіх основних функцій: емоційний стан дня, опитування, нотатки дня, загальна статистика, графік емоцій і тренд стану.

Модуль фіксації настрою перевіряє, чи є введення числом від 1 до 10. Коректна оцінка зберігається разом із поточною датою.

Модуль статистики визначає кількість записів, середній настрій, найкращий день, найважчий день та останні показники опитування.

Для узагальнення реалізованих функціональних можливостей системи у таблиці 3.1 наведено основні модулі Telegram-бота, вхідні дані та результати їх обробки.

Модуль запуску створює об’єкт Bot, ініціалізує Dispatcher і запускає polling. Асинхронна функція main забезпечує постійне очікування нових повідомлень.

Модуль меню містить кнопки для всіх основних функцій: емоційний стан дня, опитування, нотатки дня, загальна статистика, графік емоцій і тренд стану.

Модуль фіксації настрою перевіряє, чи є введення числом від 1 до 10. Коректна оцінка зберігається разом із поточною датою.

Модуль статистики визначає кількість записів, середній настрій, найкращий день, найважчий день та останні показники опитування.

Таблиця 3.1 – Функціональні модулі

Функція	Вхідні дані	Результат
Запис настрою	Число 1-10	Збережена оцінка
Нотатка	Текст	Емоційний коментар
Опитування	так/ні	4 показники
Графік	2+ записи	PNG-графік
Тренд	3+ записи	Висновок про динаміку

3.4 Організація збереження даних і статистики

У прототипі дані зберігаються в оперативній пам’яті. Такий підхід достатній для демонстрації алгоритмів, але для промислової системи доцільно використати SQLite або PostgreSQL [12].

Список moods містить записи настрою з полями score та date. Ці дані використовуються для розрахунку середнього настрою та побудови графіка.

Список notes містить текстові нотатки користувача. У майбутньому ці дані можна використовувати для аналізу частоти емоційних слів у часі.

Поле mode виконує роль простого механізму станів і показує, як інтерпретувати наступне повідомлення користувача.

3.5 Висновки за розділом 3

У третьому розділі було виконано проєктування та реалізацію Telegram-бота Mind Balance. Розроблено загальну архітектуру програмної системи, яка складається з модулів обробки повідомлень, аналізу тексту, статистики та візуалізації даних. Побудовано UML-діаграми, що відображають структуру системи та взаємодію між її компонентами. Реалізовано основні функціональні модулі: запис емоційного стану, аналіз нотаток, проведення опитування, формування статистики та побудову графіків. Також описано механізм збереження даних користувача та принципи організації внутрішньої структури системи.

4 ТЕСТУВАННЯ ТА ДОСЛІДЖЕННЯ СИСТЕМИ

4.1 Методика тестування

Тестування виконувалося для перевірки функціональних вимог, стабільності сценаріїв взаємодії та правильності розрахунків. Перевірено введення настрою, нотатки, опитування, статистику, графік і тренд.

Методика базується на функціональних тест-кейсах. Для кожної функції визначено вхідні дані, очікуваний результат і фактичний результат [13].

Додатково перевірено некоректні введення: текст замість числа, число поза діапазоном, неправильну відповідь в опитуванні та недостатню кількість записів для графіка.

Критерієм успішності є відповідність фактичної поведінки очікуваній.

4.2 Функціональне тестування

Під час тестування запису настрою підтверджено, що система коректно обробляє лише числові значення від 1 до 10. Некоректні дані не зберігаються.

Тестування нотаток показало, що словниковий аналіз працює за закладеними правилами. Повідомлення з позитивними словами отримують позитивний коментар, а з негативними - негативний.

Опитування послідовно надсилає всі питання та приймає лише відповіді «так» або «ні». Після завершення система розраховує чотири показники стану.

Функції статистики, графіка і тренду працюють після накопичення достатньої кількості записів.

У таблиці 4.1 наведено результати функціонального тестування основних можливостей Telegram-бота Mind Balance.

Таблиця 4.1 – Результати функціонального тестування

№	Сценарій	Вхідні дані	Очікуваний результат	Статус
1	Запуск	/start	Меню	Успішно
2	Настрій	8	Запис збережено	Успішно
3	Некоректний настрій	15	Помилка діапазону	Успішно
4	Позитивна нотатка	сьогодні чудово	Позитивний коментар	Успішно
5	Негативна нотатка	мені сумно	Негативний коментар	Успішно
6	Опитування	так/ні	Розрахунок показників	Успішно

4.3 Аналіз результатів та обмеження системи

Результати тестування підтвердили працездатність Telegram-бота Mind Balance. Система виконує основні функції, передбачені постановкою задачі.

Головною перевагою рішення є простота використання. Користувач працює з ботом через меню Telegram і отримує відповіді у вигляді повідомлень або зображення.

Система має обмеження: дані зберігаються лише в оперативній пам’яті, словниковий аналіз не враховує контекст, а опитування не є діагностичним інструментом.

Подальший розвиток може включати базу даних, розширення словників, лематизацію української мови, моделі машинного навчання, експорт статистики та нагадування.

4.4 Робота з чат-ботом та взаємодія з інтерфейсом

Для демонстрації роботи Telegram-бота Mind Balance було виконано тестові сценарії взаємодії користувача з інтерфейсом системи. У цьому підрозділі наведено скріншоти основних етапів роботи з ботом: запуск, введення настрою, проходження опитування, створення нотаток, перегляд статистики, побудову графіка та визначення тренду. Кожен приклад ілюструє конкретний запит користувача та відповідь чат-бота. Скріншоти підтверджують, що взаємодія відбувається через стандартний інтерфейс Telegram із кнопковим меню. Це дозволяє оцінити не лише працездатність алгоритмів, а й зручність користування розробленою системою.

На рисунку 4.1 показано головне меню Telegram-бота Mind Balance після запуску команди /start. Бот вітає користувача та пропонує обрати одну з функцій щоденника.

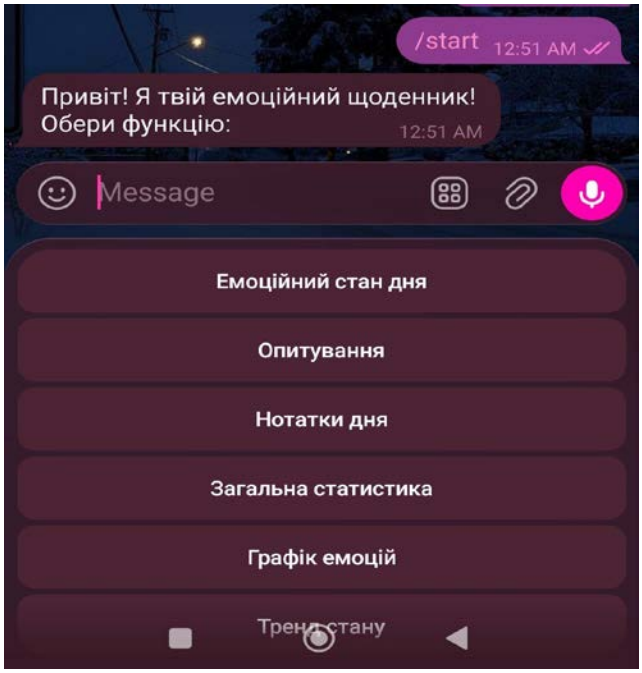

Рисунок 4.1 – Головне меню Telegram-бота Mind Balance

На екрані відображені кнопки для фіксації емоційного стану, проходження опитування, створення нотатки, перегляду статистики, побудови графіка та аналізу тренду. Користувач може натискати готові кнопки замість ручного введення команд. Такий інтерфейс робить взаємодію з ботом зрозумілою та зручною.

На рисунку 4.2 наведено приклад роботи функції фіксації емоційного стану дня.

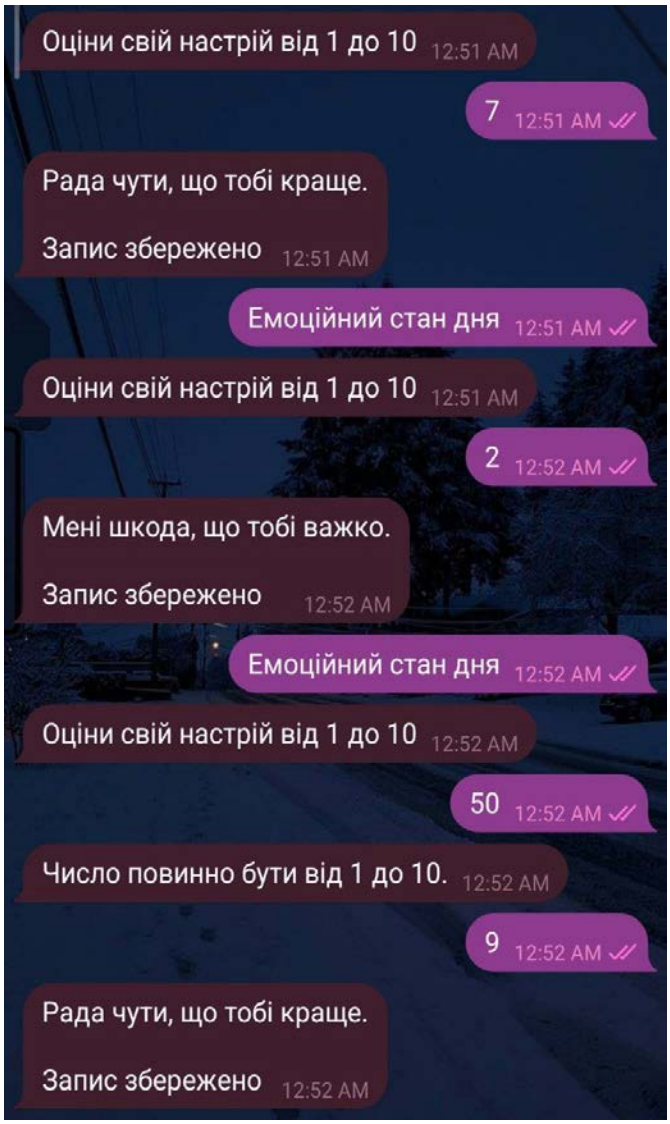

Рисунок 4.2 – Приклад введення емоційного стану користувача

Користувач обирає відповідний пункт меню та вводить оцінку настрою за шкалою від 1 до 10. Бот перевіряє введене значення та приймає лише числа в заданому діапазоні. Після коректного введення система надсилає короткий коментар і повідомляє про збереження запису. У прикладі показано кілька оцінок, які додаються до історії настрою користувача.

На рисунку 4.3 показано початковий етап проходження опитування.

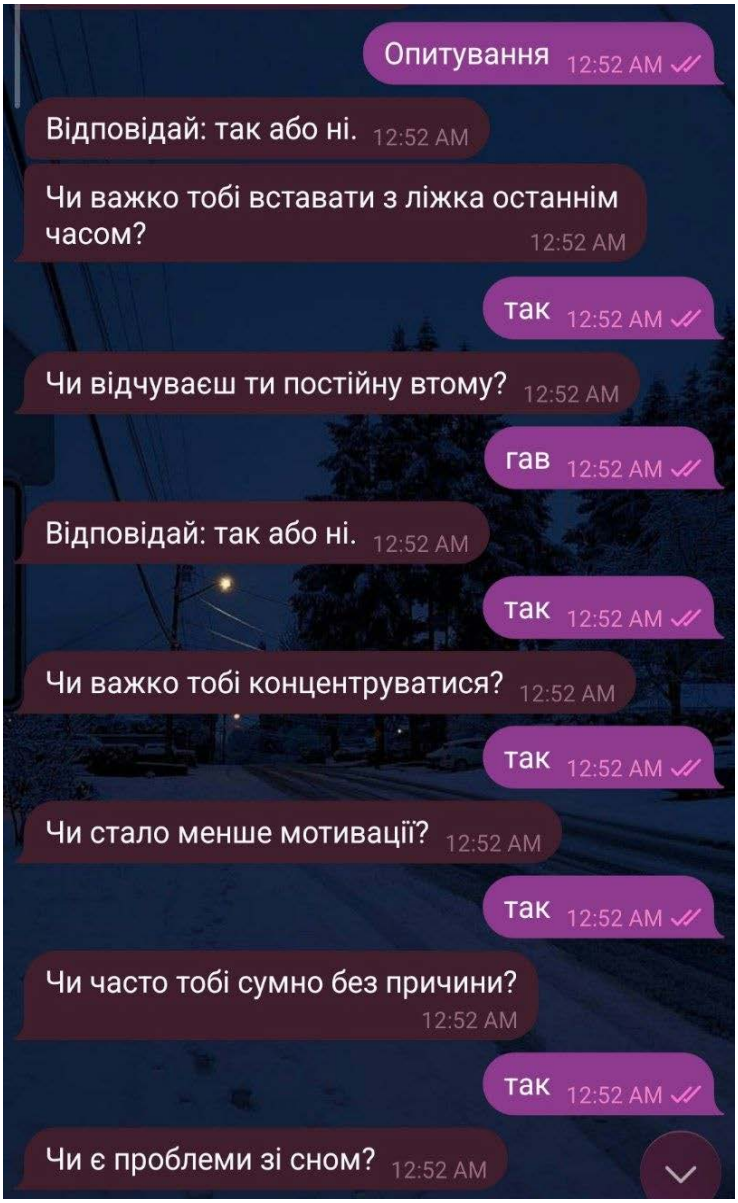

Рисунок 4.3 – Початок проходження опитування у чат-боті

Користувач натискає кнопку «Опитування», після чого бот просить відповідати лише «так» або «ні». Далі система послідовно виводить питання щодо сну, втоми, концентрації, мотивації та емоційного стану. Кожна відповідь зберігається у внутрішній структурі даних для подальшого розрахунку показників. Такий формат дозволяє зібрати інформацію без складного введення тексту.

На рисунку 4.4 наведено завершальну частину опитування користувача.

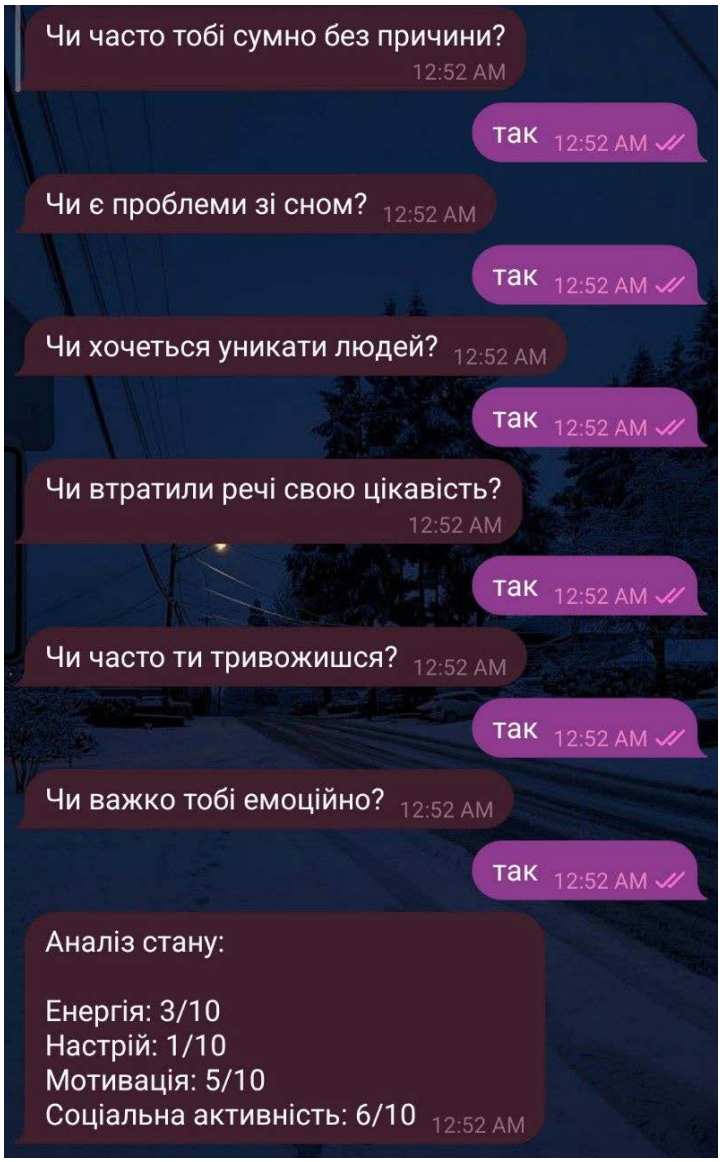

Рисунок 4.4 – Завершення опитування та результат аналізу стану

Бот продовжує ставити питання та фіксує відповіді «так» або «ні». Після останнього питання система автоматично розраховує показники енергії, настрою, мотивації та соціальної активності. Результат аналізу подається у вигляді короткого текстового повідомлення. У прикладі видно, що кожен показник має числову оцінку за шкалою від 1 до 10.

На рисунку 4.5 продемонстровано роботу функції «Нотатки дня».

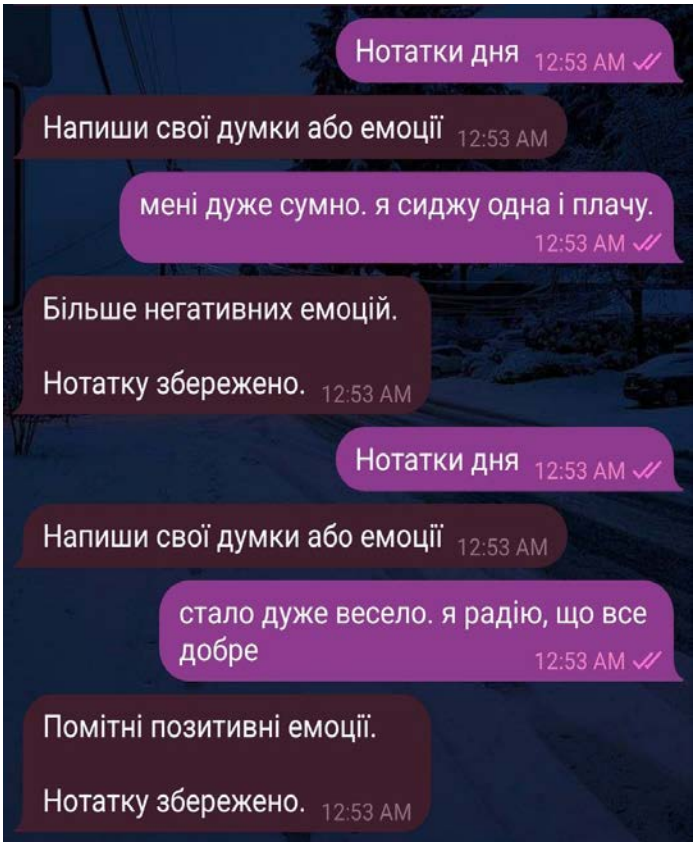

Рисунок 4.5 – Приклад роботи модуля нотаток дня

Користувач вводить текстове повідомлення, у якому описує власні емоції або події дня. Система аналізує слова в повідомленні та порівнює їх зі словниками позитивних і негативних маркерів. Після аналізу бот формує короткий висновок про переважання негативних або позитивних емоцій. Нотатка зберігається для подальшого використання у статистиці.

На рисунку 4.6 показано приклад формування загальної статистики. Бот відображає кількість записів настрою та кількість збережених нотаток. Також система розраховує середній настрій користувача на основі попередніх оцінок. Окремо виводяться найкращий і найважчий день із зазначенням оцінки та часу запису. Наприкінці повідомлення відображаються останні результати опитування за основними показниками стану.

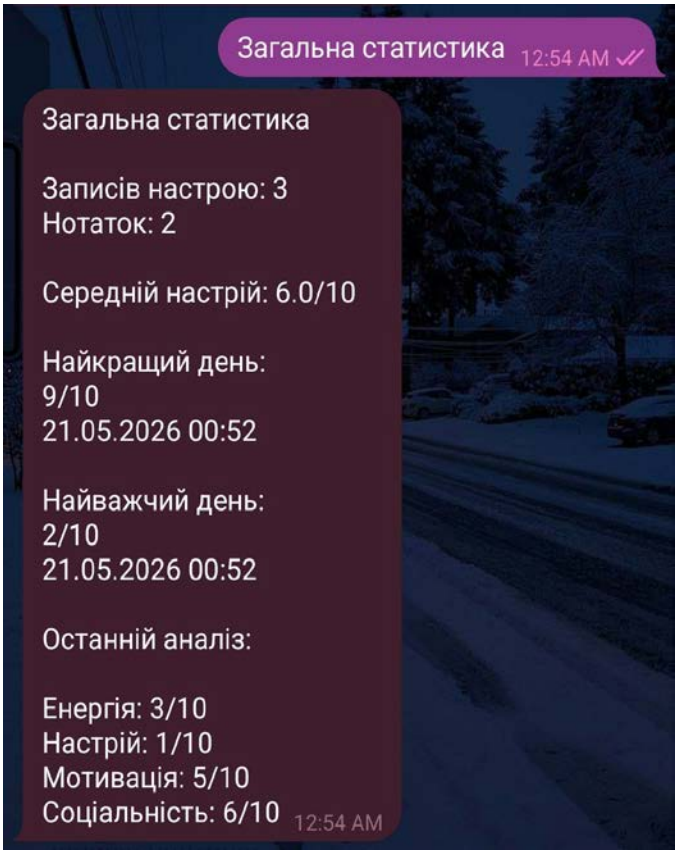

Рисунок 4.6 – Відображення загальної статистики користувача

На рисунку 4.7 наведено результат роботи функції побудови графіка емоцій. Користувач натискає кнопку «Графік емоцій», після чого бот формує зображення з динамікою оцінок настрою. На графіку по горизонталі відображається дата запису, а по вертикалі - числова оцінка настрою. Для

побудови візуалізації використовується бібліотека Matplotlib. Графік дає змогу швидко оцінити зміну емоційного стану користувача у часі.

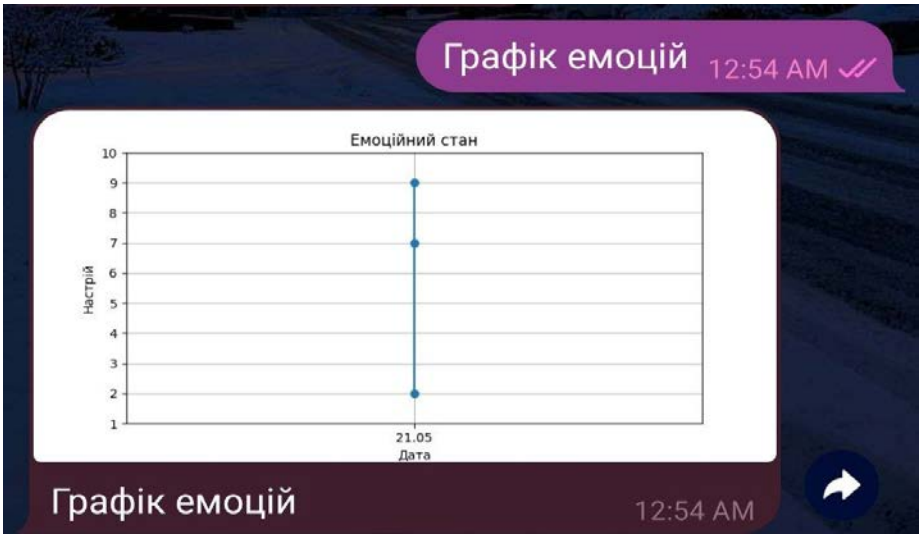

Рисунок 4.7 – Побудова графіка емоційного стану

На рисунку 4.8 показано приклад роботи функції аналізу тренду стану. Бот порівнює середні оцінки настрою у першій та другій частинах збережених записів. Якщо середнє значення зменшується, система повідомляє про ознаки погіршення. У прикладі бот показує попередній середній показник і поточне середнє значення. Такий підхід дозволяє користувачу побачити загальну динаміку без ручних розрахунків.

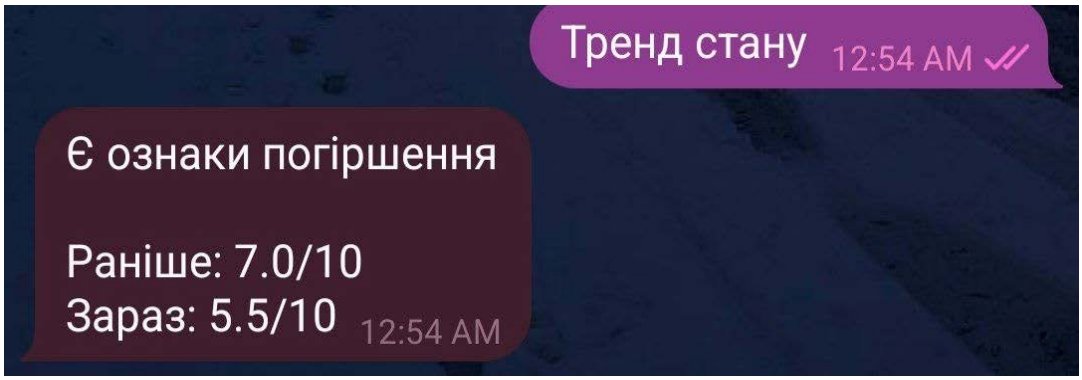

Рисунок 4.8 – Визначення тренду емоційного стану

4.5 Висновки до розділу 4

У четвертому розділі проведено тестування та дослідження роботи Telegram-бота Mind Balance. Перевірено основні функції системи, зокрема введення настрою, обробку нотаток, проходження опитування та побудову статистики. Результати тестування підтвердили коректність реалізованих алгоритмів та стабільність роботи системи. Також визначено основні обмеження розробленого рішення та напрями його подальшого вдосконалення.

ВИСНОВКИ

У дипломній роботі розроблено та досліджено Telegram-бот Mind Balance, призначений для фіксації, первинного аналізу та візуалізації емоційного стану користувача. Проаналізовано предметну область, розглянуто методи sentiment analysis, обґрунтовано вибір Telegram, Python, Aiogram і Matplotlib.

Система підтримує введення оцінки настрою, збереження нотаток, проведення опитування, розрахунок показників енергії, настрою, мотивації та соціальності, формування статистики, побудову графіка емоцій і визначення тренду стану.

Побудовано UML-діаграми, які відображають варіанти використання, компоненти системи, активність користувача та послідовність обробки повідомлень. Це дозволило формалізувати архітектуру програмного засобу.

Тестування підтвердило коректність основних функцій. Система правильно перевіряє введення, виконує словниковий аналіз нотаток, розраховує показники опитування, формує статистику і будує графік настрою.

Практична цінність роботи полягає у створенні навчального прототипу цифрового щоденника емоційного стану. Система не є медичним діагностичним засобом, але демонструє принципи розроблення чат-ботів, аналізу тексту, статистичної обробки даних і візуалізації.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Teens, screens and mental health. URL: https://www.who.int/europe/news/item/25-09-2024-teens--screens-and-mental-health (date of access: 27.05.2026).

2. Mood Tracking Apps: Benefits and Features. URL: https://www.healthline.com/health/mental-health/mental-health-apps (date of access: 27.05.2026).

3. Speech and Language Processing. URL: https://web.stanford.edu/~jurafsky/slp3/ (date of access: 25.06.2026).

4. Natural Language Processing with Python. URL: https://www.nltk.org/book/ (date of access: 25.06.2026).

5. Python 3.14.5 Documentation. URL: https://docs.python.org/3/ (date of access: 25.06.2026).

6. Aiogram Documentation. URL: https://docs.aiogram.dev/en/v3.28.2/ (date of access: 25.06.2026).

7. Telegram Bot API Documentation. URL: https://core.telegram.org/bots/api (date of access: 25.06.2026).

8. Matplotlib 3.10.9 Documentation. URL: https://matplotlib.org/stable/ (date of access: 25.06.2026).

9. Opinion Mining and Sentiment Analysis. URL: https://www.cs.cornell.edu/home/llee/omsa/omsa.pdf (date of access: 25.06.2026).

10. Sentiment Analysis and Opinion Mining. URL: https://www.cs.uic.edu/~liub/FBS/SentimentAnalysis-and-OpinionMining.pdf (date of access: 25.06.2026).

11. Unified Modeling Language (UML) Overview. URL: https://www.ibm.com/docs/en/engineering-lifecycle-management-suite/design-rhapsody/10.0.2 (date of access: 25.06.2026).

12. Software Engineering – Tutorials and Concepts. URL: https://www.geeksforgeeks.org/software-engineering/software-engineering/ (date of access: 25.06.2026).

13. Software Engineering Methodologies and Practices. URL: https://www.atlassian.com/software-development (date of access: 25.06.2026).

ДОДАТОК А

Текст програми

А.1 Програмний код Telegram-бота Mind Balance

```
from aiogram import Bot, Dispatcher, F
from aiogram.types import (
    Message,
    ReplyKeyboardMarkup,
    KeyboardButton,
    FSInputFile
)
from aiogram.filters import CommandStart
import asyncio
from datetime import datetime
import matplotlib.pyplot as plt

TOKEN = "МІЙ ТОКЕН"

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

user_data = {}

menu = ReplyKeyboardMarkup(
    keyboard=[
        [KeyboardButton(text="Емоційний стан дня")],
        [KeyboardButton(text="Опитування")],
        [KeyboardButton(text="Нотатки дня")],
        [KeyboardButton(text="Загальна статистика")],
        [KeyboardButton(text="Графік емоцій")],
        [KeyboardButton(text="Тренд стану")]
    ],
    resize_keyboard=True
)

questions = [
    "Чи важко тобі вставати з ліжка останнім часом?",
    "Чи відчуваєш ти постійну втому?",
    "Чи важко тобі концентруватися?",
    "Чи стало менше мотивації?",
    "Чи часто тобі сумно без причини?",
    "Чи є проблеми зі сном?",
    "Чи хочеться уникати людей?",
```

```
    "Чи втратили речі свою цікавість?",
    "Чи часто ти тривожишся?",
    "Чи важко тобі емоційно?"
]

positive_words = [
    "добре",
    "щаслива",
    "щасливий",
    "весело",
    "чудово",
    "супер",
    "радість",
    "люблю",
    "спокійно",
    "нормально",
    "краще"
]

negative_words = [
    "погано",
    "жахливо",
    "сумно",
    "боляче",
    "самотньо",
    "тривога",
    "втома",
    "апатія",
    "ненавиджу",
    "плачу",
    "страшно",
    "депресія"
]

@dp.message(CommandStart())
async def start(message: Message):

    user_id = message.from_user.id

    user_data[user_id] = {
        "mode": None,
        "moods": [],
```

```
        "notes": [],
        "analysis": None,
        "survey_step": 0,
        "survey_answers": []
    }

    await message.answer(
        "Привіт! Я твій емоційний щоденник! Обери функцію:",
        reply_markup=menu
    )

@dp.message(F.text == "Емоційний стан дня")
async def mood(message: Message):

    user_id = message.from_user.id

    user_data[user_id]["mode"] = "mood"

    await message.answer(
        "Оціни свій настрій від 1 до 10"
    )

@dp.message(F.text == "Нотатки дня")
async def notes(message: Message):

    user_id = message.from_user.id

    user_data[user_id]["mode"] = "note"

    await message.answer(
        "Напиши свої думки або емоції"
    )

@dp.message(F.text == "Опитування")
async def survey(message: Message):

    user_id = message.from_user.id

    user_data[user_id]["mode"] = "survey"
    user_data[user_id]["survey_step"] = 0
    user_data[user_id]["survey_answers"] = []
```

```
    await message.answer(
        "Відповідай: так або ні."
    )

    await message.answer(questions[0])

@dp.message(F.text == "Загальна статистика")
async def general_stats(message: Message):

    user_id = message.from_user.id

    moods = user_data[user_id]["moods"]
    notes = user_data[user_id]["notes"]
    analysis = user_data[user_id]["analysis"]

    if not moods:
        await message.answer("Недостатньо даних")
        return

    average_mood = sum(
        m["score"] for m in moods
    ) / len(moods)

    best_day = max(
        moods,
        key=lambda x: x["score"]
    )

    worst_day = min(
        moods,
        key=lambda x: x["score"]
    )

    text = (
        "Загальна статистика\n\n"

        f"Записів настрою: {len(moods)}\n"
        f"Нотаток: {len(notes)}\n\n"

        f"Середній настрій: "
        f"{average_mood:.1f}/10\n\n"
```

```
        f"Найкращий день:\n"
        f"{best_day['score']}/10\n"
        f"{best_day['date'].strftime('%d.%m.%Y %H:%M')}\n\n"

        f"Найважчий день:\n"
        f"{worst_day['score']}/10\n"
        f"{worst_day['date'].strftime('%d.%m.%Y %H:%M')}\n"
    )

    if analysis:

        text += (
            "\nОстанній аналіз:\n\n"

            f"Енергія: "
            f"{analysis['energy']}/10\n"

            f"Настрій: "
            f"{analysis['mood']}/10\n"

            f"Мотивація: "
            f"{analysis['motivation']}/10\n"

            f"Соціальність: "
            f"{analysis['social']}/10\n"
        )

    await message.answer(text)

@dp.message(F.text == "Графік емоцій")
async def emotion_graph(message: Message):

    user_id = message.from_user.id

    moods = user_data[user_id]["moods"]

    if len(moods) < 2:
        await message.answer(
            "Недостатньо даних для графіка."
        )
        return
```

```
dates = []
scores = []

for mood in moods:

    dates.append(
        mood["date"].strftime("%d.%m")
    )

    scores.append(
        mood["score"]
    )

plt.rcParams["font.family"] = "DejaVu Sans"

plt.figure(figsize=(8, 4))

plt.plot(
    dates,
    scores,
    marker="o"
)

plt.title("Емоційний стан")
plt.xlabel("Дата")
plt.ylabel("Настрій")

plt.ylim(1, 10)

plt.grid(True)

filename = f"graph_{user_id}.png"

plt.savefig(filename)

plt.close()

photo = FSInputFile(filename)

await message.answer_photo(
    photo=photo,
    caption="Графік емоцій"
```

```
    )

@dp.message(F.text == "Тренд стану")
async def trend(message: Message):

    user_id = message.from_user.id

    moods = user_data[user_id]["moods"]

    if len(moods) < 3:
        await message.answer(
            "Недостатньо даних для аналізу."
        )
        return

    scores = [
        mood["score"]
        for mood in moods
    ]

    first_half = scores[:len(scores)//2]
    second_half = scores[len(scores)//2:]

    first_avg = (
        sum(first_half) / len(first_half)
    )

    second_avg = (
        sum(second_half) / len(second_half)
    )

    if second_avg > first_avg:

        result = (
            "Твій стан покращується"
        )

    elif second_avg < first_avg:

        result = (
            "Є ознаки погіршення"
        )
```

```
    else:

        result = (
            "Стан стабільний."
        )

    await message.answer(
        f"{result}\n\n"

        f"Раніше: {first_avg:.1f}/10\n"
        f"Зараз: {second_avg:.1f}/10"
    )

@dp.message()
async def handle(message: Message):

    user_id = message.from_user.id

    if user_id not in user_data:
        await message.answer("Напиши /start")
        return

    mode = user_data[user_id]["mode"]

    if mode == "note":

        note = message.text.lower()

        positive_count = 0
        negative_count = 0

        for word in positive_words:
            if word in note:
                positive_count += 1

        for word in negative_words:
            if word in note:
                negative_count += 1

        if negative_count > positive_count:
```

```
            emotion_comment = (
                "Більше негативних емоцій."
            )

        elif positive_count > negative_count:

            emotion_comment = (
                "Помітні позитивні емоції."
            )

        else:

            emotion_comment = (
                "Стан виглядає змішаним."
            )

        user_data[user_id]["notes"].append({
            "text": message.text,
            "date": datetime.now()
        })

        user_data[user_id]["mode"] = None

        await message.answer(
            f"{emotion_comment}\n\n"
            "Нотатку збережено."
        )

        return

    if mode == "mood":

        if not message.text.isdigit():

            await message.answer(
                "Введи число від 1 до 10."
            )

            return

        score = int(message.text)
```

```
if score < 1 or score > 10:

    await message.answer(
        "Число повинно бути від 1 до 10."
    )

    return

user_data[user_id]["moods"].append({
    "score": score,
    "date": datetime.now()
})

user_data[user_id]["mode"] = None

if score <= 3:

    text = (
        "Мені шкода, "
        "що тобі важко."
    )

elif score <= 6:

    text = (
        "День виглядає непростим."
    )

else:

    text = (
        "Рада чути, "
        "що тобі краще."
    )

await message.answer(
    f"{text}\n\n"
    "Запис збережено"
)

return
```

```
if mode == "survey":

    answer = message.text.lower()

    if answer not in ["так", "ні"]:

        await message.answer(
            "Відповідай: так або ні."
        )

        return

    user_data[user_id]["survey_answers"].append(
        answer
    )

    user_data[user_id]["survey_step"] += 1

    step = user_data[user_id]["survey_step"]

    if step < len(questions):

        await message.answer(
            questions[step]
        )

    else:

        answers = user_data[user_id]["survey_answers"]

        energy = 10
        mood_value = 10
        motivation = 10
        social = 10

        if answers[0] == "так":
            energy -= 2

        if answers[1] == "так":
            energy -= 3

        if answers[2] == "так":
```

```
    motivation -= 2

if answers[3] == "так":
    motivation -= 3

if answers[4] == "так":
    mood_value -= 2

if answers[5] == "так":
    energy -= 2

if answers[6] == "так":
    social -= 4

if answers[7] == "так":
    mood_value -= 2

if answers[8] == "так":
    mood_value -= 2

if answers[9] == "так":
    mood_value -= 3

user_data[user_id]["analysis"] = {
    "energy": max(1, energy),
    "mood": max(1, mood_value),
    "motivation": max(1, motivation),
    "social": max(1, social)
}

analysis = user_data[user_id]["analysis"]

result = (
    "Аналіз стану:\n\n"

    f"Енергія: "
    f"{analysis['energy']}/10\n"

    f"Настрій: "
    f"{analysis['mood']}/10\n"

    f"Мотивація: "
```

```
            f"{analysis['motivation']}/10\n"

            f"Соціальна активність: "
            f"{analysis['social']}/10\n"
        )

        await message.answer(result)

        user_data[user_id]["mode"] = None

async def main():

    print("Бот запущений...")

    await dp.start_polling(bot)

asyncio.run(main())
```

ДОДАТОК Б

Слайди презентації

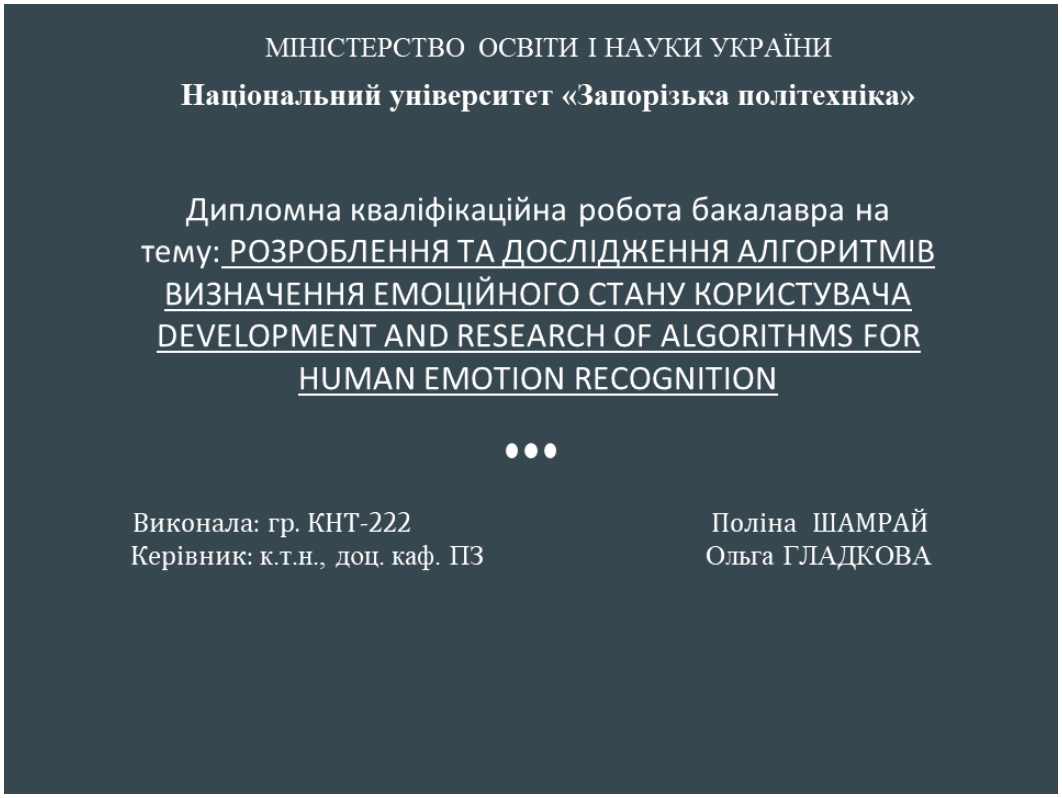

Рисунок Б.1 – Слайд 1

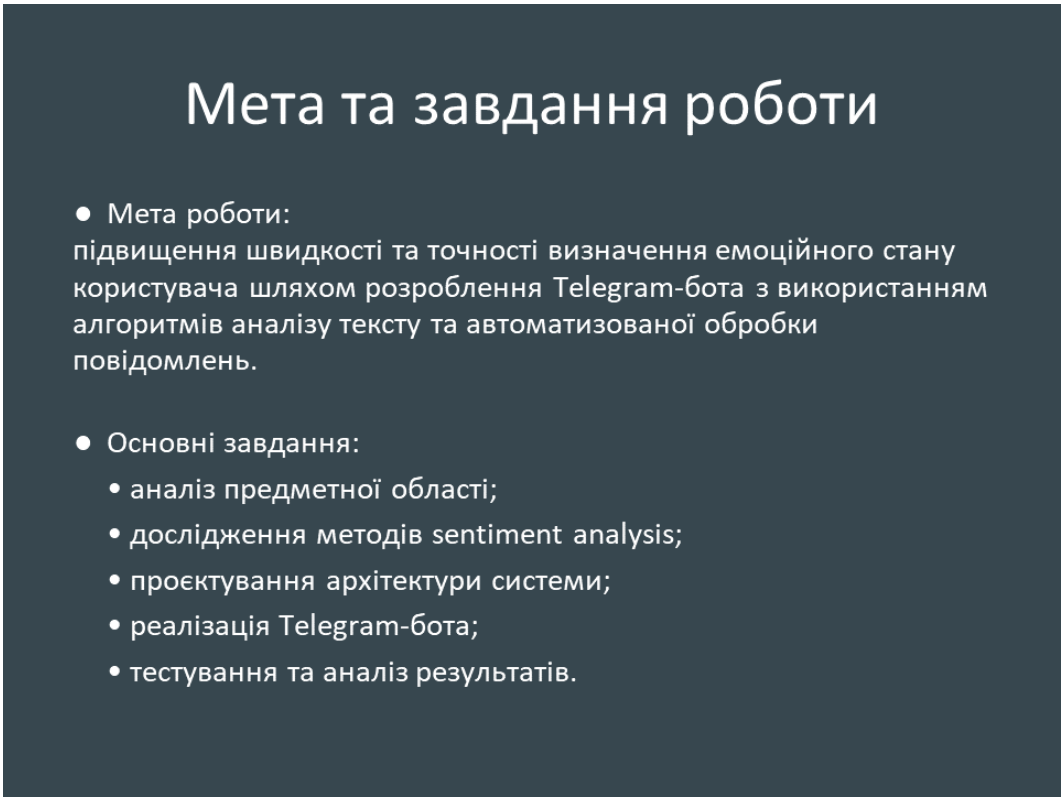

Рисунок Б.2 – Слайд 2

Рисунок Б.3 – Слайд 3

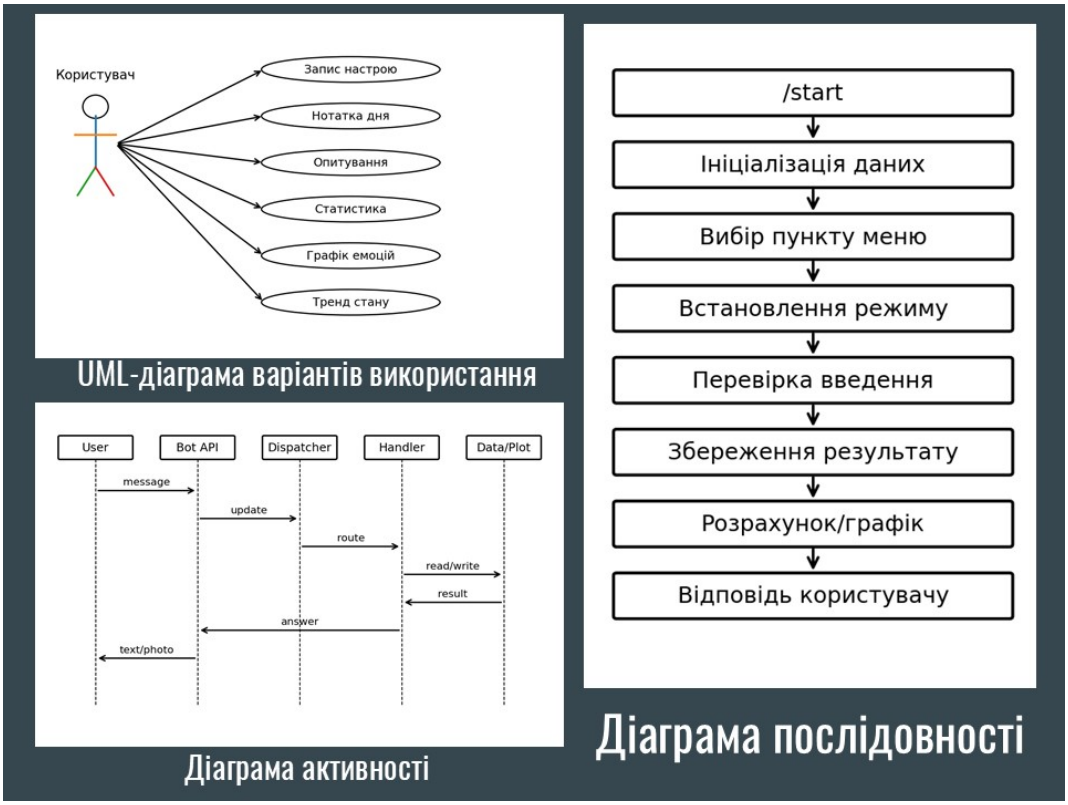

Рисунок Б.4 – Слайд 4

Рисунок Б.5 – Слайд 5

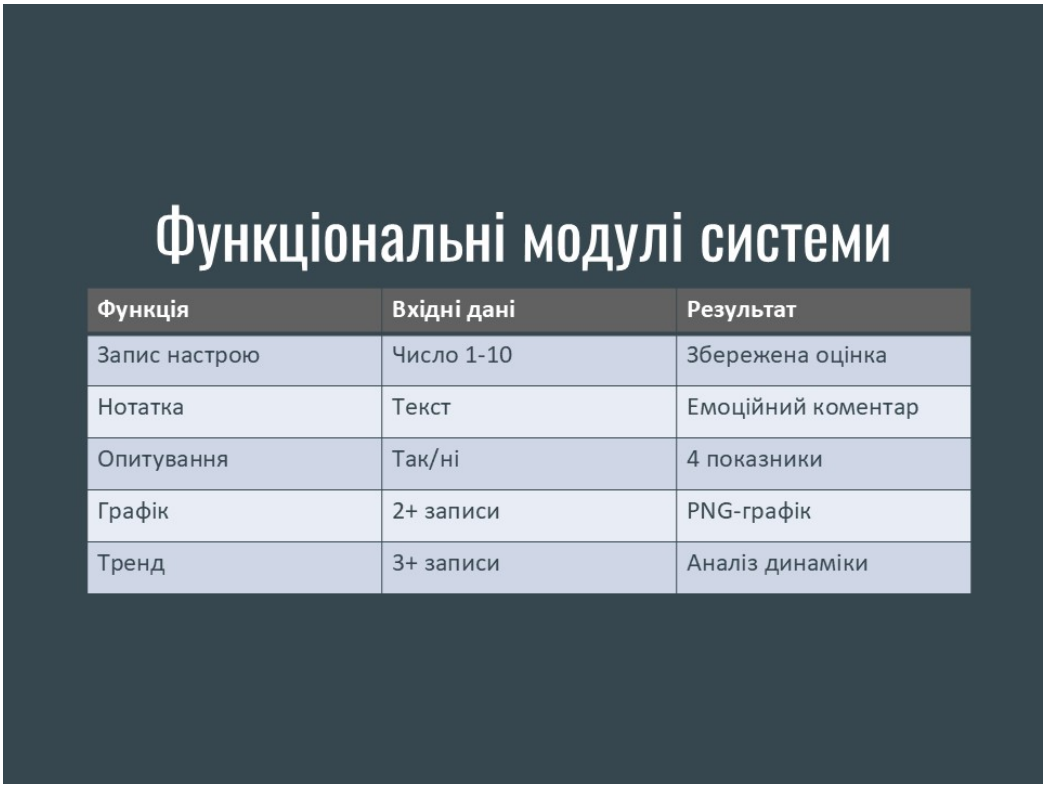

Рисунок Б.6 – Слайд 6

Рисунок Б.7 – Слайд 7

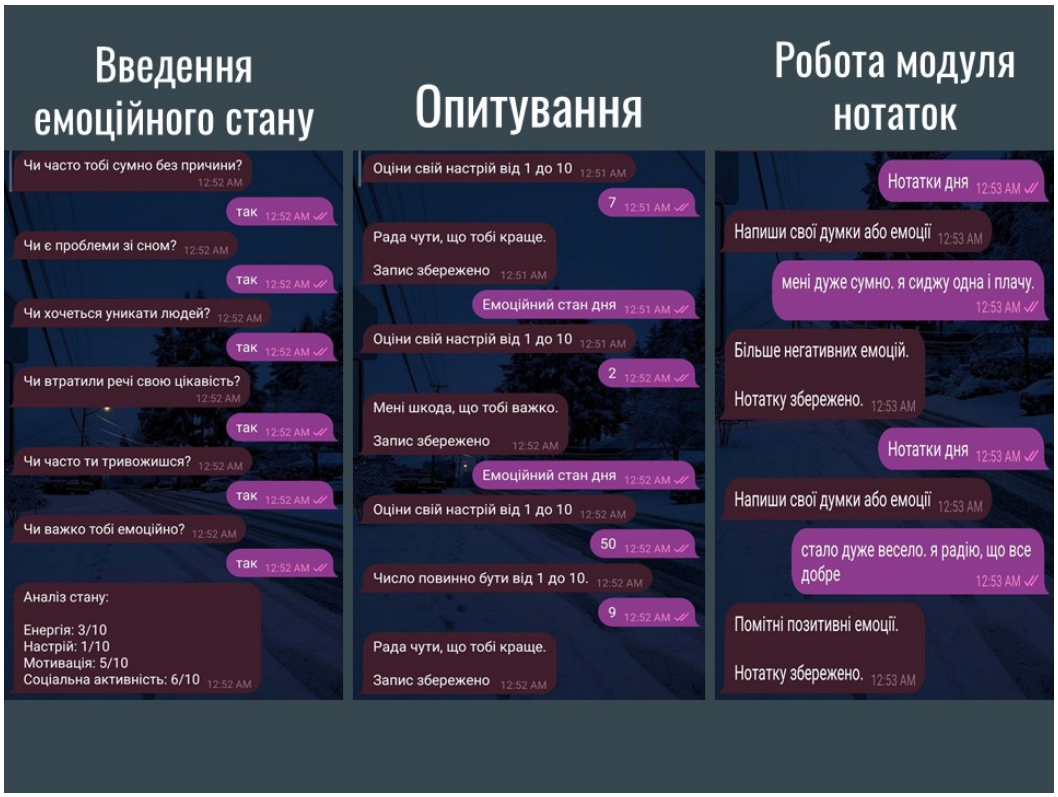

Рисунок Б.8 – Слайд 8

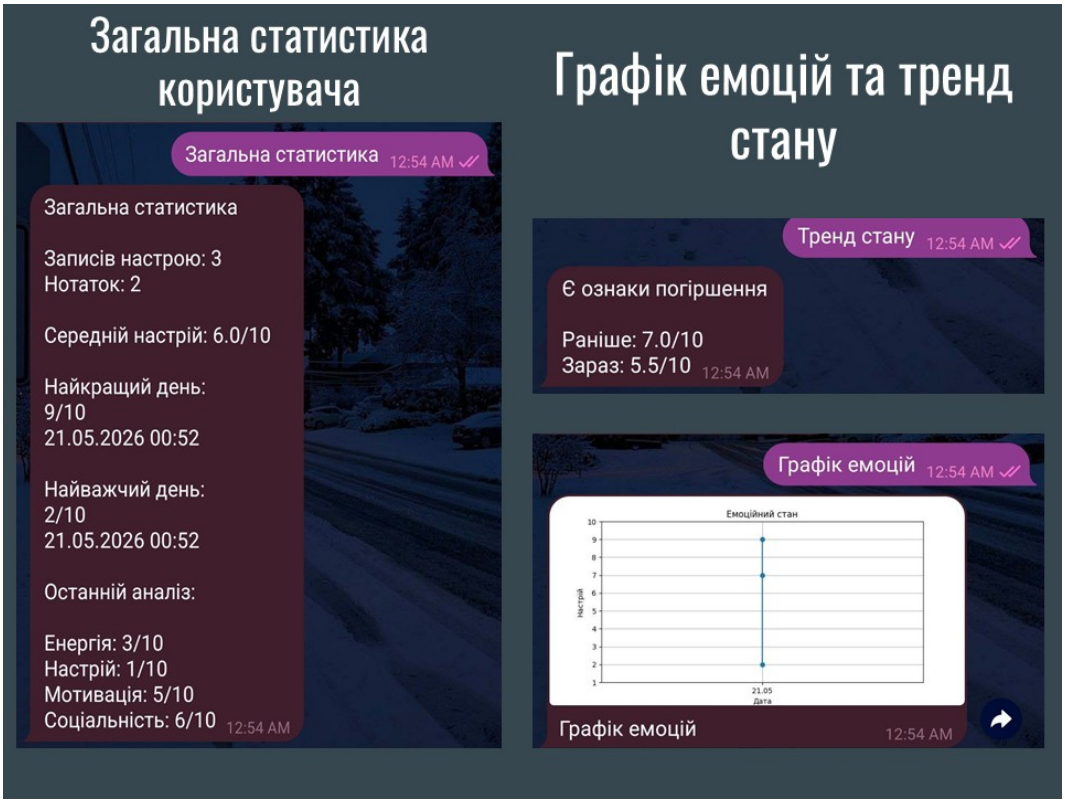

Рисунок Б.9 – Слайд 9

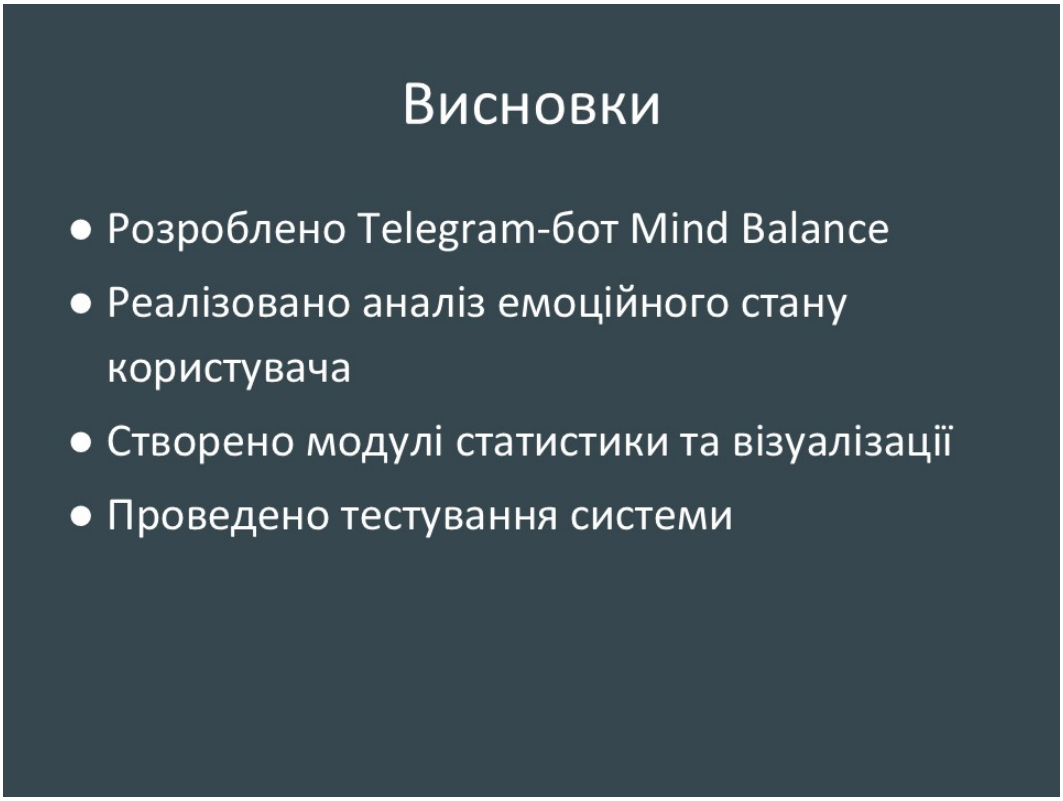

Рисунок Б.10 – Слайд 10