Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

“Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2026 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи аналітики історії ділового чату засобами штучного інтелекту”

КбПЗ-2026

Виконав здобувач вищої освіти
IV курсу, групи КН-22
ОПП «Комп’ютерні науки»
спеціальності 122 «Комп’ютерні науки»
_____ Кучер Д. М.
«____» _____ 2026 р.

Керівник проекту кандидат технічних наук,
доцент _____ Дрєєв О.М.
«____» _____ 2026 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет
Факультет *Механіко технологічний*
Кафедра *Кібербезпеки та програмного забезпечення*
Рівень вищої освіти *бакалавр*
Галузь знань *12 "Інформаційні технології"*
Спеціальність *122 "Комп'ютерні науки"*
Освітньо-професійна (освітньо-наукова) програма *"Комп'ютерні науки"*

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
_____ року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Кучера Дениса Миколайовича
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи аналітики історії ділового чату засобами штучного інтелекту*
2. Керівник роботи *Дрєєв Олександр Миколайович, канд. техн. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу *№116-02 від 06.02.26 року*
3. Строк подання студентом роботи до захисту *09.06.2026 року*
4. Мета та завдання курсового проєкту *Метою роботи є розробка програмного забезпечення системи аналітики історії ділового чату засобами штучного інтелекту*
5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Призначення та область використання.
2. Перегляд аналогічних існуючих систем.
3. Опис і обґрунтування проектних рішень.
4. Етапи програмування системи.
5. Впровадження системи в промислову експлуатацію.
6. Висновки
6. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Структурна схема системи 1 аркуш
Функціональна схема системи 1 аркуш
Діаграма процесів 1 аркуш
Блок-схема алгоритму роботи додатку 2 аркуша

7. Дата видачі завдання 06.02.26 р.

КПІ-2026

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Приміт ка
1.	Аналіз існуючих систем	10.03.2026	
2.	Постановка задачі, оформлення ТЗ	14.03.2026	
3.	Розробка моделі компонента	18.03.2026	
4.	Розробка структур даних	25.03.2026	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2026	
6.	Програмування алгоритмів	10.04.2026	
7.	Оформлення ПЗ	20.04.2026	
8.	Попередній захист роботи	16.05.2026	

Дата видачі завдання
__06.02.26__ р.

Підпис керівника
_____ Дрєєв О.М.
(прізвище та ініціали)

Завдання прийнято до виконання
__06.02.26__ р.

Підпис здобувача
_____ Кучер Д.М.
(прізвище та ініціали)

КвПЗ_2026

АНОТАЦІЯ

Кучер Д. М. Програмне забезпечення системи аналітики історії ділового чату засобами штучного інтелекту. 122 Комп'ютерні науки. Центральноукраїнський національний технічний університет. Кропивницький 2026.

Об'єкт дослідження – накопичення текстових повідомлень у ділових чатах, а предмет – методи та засоби штучного інтелекту для їх аналітики. Мета роботи – розробка програмного забезпечення автоматизації аналізу історії ділових чатів.

У процесі дослідження виконано огляд і аналіз існуючих методів, алгоритмів та програмних засобів. Програмне забезпечення реалізовано мовою Python із використанням методів обробки природної мови, описано його компоненти та розроблено інтерфейс для взаємодії з користувачем.

Отримані результати дають змогу аналізувати історію ділових чатів, відстежувати динаміку активності учасників та формувати звіти. Програмне забезпечення може використовуватися на ПЕОМ архітектури IBM PC з ОС Windows 10/11 у малих та середніх команд для оцінки ефективності комунікацій.

Ключові слова: комп'ютерні науки, штучний інтелект, аналіз текстів, аналітика чатів, чат-бот, Telegram.

КОПЗ_2026

ABSTRACT

Kucher D. M. Software for business chat history analytics using artificial intelligence tools. 122 Computer Science. Central Ukrainian National Technical University. Kropyvnytskyi, 2026.

The object of the study is the accumulation of text messages in business chats, and the subject is methods and tools of artificial intelligence for their analytics. The aim of the work is to develop software that automates the analysis of business chat history.

The research includes a review and analysis of existing methods, algorithms, and software tools. The software is implemented in Python using an object-oriented approach and natural language processing techniques. Its components are described, and a user interface is developed.

The obtained results allow analyzing business chat history, tracking participant activity dynamics, and generating reports. The software can be used on IBM PC-compatible computers running Windows 10/11 in small and medium-sized teams to evaluate communication effectiveness.

Keywords: computer science, artificial intelligence, text analysis, chat analytics, chatbot, Telegram.

КОТЗ_2026

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ 2
ВСТУП 4
1 ПРИЗАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ 6
1.1 Призначення системи 6
1.2 Область застосування 7
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ 8
2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми кваліфікаційної бакалаврської роботи 8
2.2 Обґрунтування вибору засобів для побудови системи та мови програмування 11
2.3 Розгорнута постановка завдання 12
3 ОПИС І ОБГРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ 15
3.1 Опис функціонування системи 15
3.2 Розробка структурної схеми 18
3.3 Розробка функціональної схеми 21
3.4 Розробка діаграми процесів 24
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ 27
4.1 Блок-схеми та опис алгоритмів функціонування системи 27
4.2 Захист розробленого програмного забезпечення 38
5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ 41
6 ОСНОВНІ ВИСНОВКИ 46
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 48

КПЗ – 2026

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

API (Application Programming Interface - інтерфейс прикладного програмування) - набір інструментів, за допомогою яких одна програм може звертатися до функцій іншої програми або сервісу.

Callback-запит (Callback Query) - тип події в Telegram API, що виникає при натисканні кнопки inline-клавіатури й дозволяє боту змінювати повідомлення або виконувати дії без надсилання тексту.

FSM (Finite State Machine – скінченний автомат) - модель керування станами, переходи між якими відбуваються у відповідь на події.

Inline-клавіатура (Inline Keyboard) - тип клавіатури в Telegram, кнопки якої прикріплені до конкретного повідомлення та надсилають callback-запит при натисканні.

LLM (Large Language Model - велика мовна модель) - нейронна мережа, навчена на великих об’ємах текстових даних, здатна розуміти, генерувати та аналізувати природну мову.

NLP (Natural Language Processing - обробка природньої мови) - підгалузь штучного інтелекту, що займається взаємодією комп’ютерів із людською мовою.

PDF (Portable Document Format - переносний формат документів) - формат файлів для представлення документів у вигляді, незалежному від операційної системи та пристрою, зі збереженням форматування.

SQL (Structured Query Language - структурована мова запитів) - стандартизована мова для створення, зміни та отримання даних із реляційних баз даних.

SQLite - легка вбудована реляційна база даних, яка не потребує окремого серверного процесу й зберігає всю інформацію в одному файлі.

КБПЗ_2026

Telegram-бот - автономна програма, працююча всередині месенджера Telegram, взаємодіє з користувачем через текстові повідомлення та команди.

БД - база даних, структуроване сховище інформації, організоване для ефективного збереження, пошуку та оновлення даних.

Дедлайн (deadline) - кінцева дата або час виконання поставленого завдання.

Контекст - накопичена історія спілкування та попередні звіти, які передаються ШІ-моделі для врахування довгострокових домовленостей.

Паралельне виконання (asyncio, асинхронність) - метод організації програми, за якого вона може чекати відповіді з мережі і одночасно обробляти інші запити, не блокуючи роботу.

Планувальник (scheduler) - компонент програми, який виконує задані функції автоматично за певним розкладом.

Тегування (tagging) - процес автоматичного додавання до текстового повідомлення спеціальних тегів.

ШІ - штучний інтелект, галузь комп'ютерних наук, що створює системи, здатні імітувати людський розум, навчатися на даних, аналізувати інформацію та приймати рішення.

КбПЗ_2026

ВСТУП

Актуальність теми. В сучасному бізнесі чати стали, одним з найпростіших і найзручніших інструментів, які тільки можна уявити. Багато команд переконалися, що після запровадження окремих чат-груп працювати стало набагато швидше й ефективніше. Люди легше домовляються про нові плани, узгоджують завдання, не чекаючи на листи або зустрічі. Достатньо написати це в чат і можна бути впевненим, що повідомлення буде прочитано. Чати стали принципово новим рівним комунікацій і неможливо уявити колектив, команду, групу зі спільними інтересами без них.

Але згодом вилізла інша проблема – інформаційне перевантаження. Чати перестали бути лише робочим простором, перетворившись на дещо більше. Туди почали писати про все поспіль, Включно з особистими думками та неформальним спілкуванням, якого спочатку ніхто не планував. Важливі домовленості перемішувалися з обговоренням буденних тем. У результаті учасники чатів зіштовхнулися з надлишковим об'ємом шуму. Цей шум закривав собою дійсно важливі обговорення, дати, терміни чи завдання. Знаходити потрібне ставало дедалі важче. Проблема стала очевидною: повідомлень багато – корисної інформації мало. Саме тому з'явилася потреба у фільтрації даних.

У той же час великий розвиток отримали нейромережі. Нейромережі почали гарно розуміти тексти та зображення. Великі мовні моделі, на яких базується більш частина нейромереж – здатні аналізувати отримані текстові матеріали створюючи комплексні звіти. Отримавши дані ділових бесід вони здатні виокремлювати ключові завдання, визначати виконавців, виділяти чіткі терміни та що не менш важливо оцінювати психологічно-емоційний стан учасників. Інтеграція таких систем до роботи у сфері комунікацій отримала велику популярність за рахунок цього. Використання штучного інтелекту у ділових чатах допомогло б організувати та структурувати великі масиви даних

КПІЗ_2026

фільтруючі усе «інформаційне сміття» залишаючи лише корисну до уваги інформацію. Це також надало б керівникам можливість відстежувати діяльність своїх груп. Аналізувати активність та емоції кожного окремого учасника та пропонувати рекомендації. А це вже допомогло б уникати конфліктів у колективі.

Мета і завдання дослідження. Метою роботи є створення програмного забезпечення чат-бота на базі месенджера Telegram для аналітики та аналізу історії ділового чату.

Для досягнення поставленої мети визначено основні пункти для дослідження та реалізації:

- Проведення огляду сучасних систем чат-ботів та інтеграції ШІ в месенджерах.

- Проєктування системи чат-бота для аналітики даних методами ШІ для Telegram.

- Здійснення програмної реалізації чат-бота та його тестування.

Розробка програмного забезпечення аналітики чату засобами штучного інтелекту є відповіддю на нагальну потребу. Завдяки структурованим звітам такий підхід суттєво скорочує час, який потрібен працівникові або колезі, щоб повернутися в контекст командних обговорень.

Практична цінність роботи полягає у створенні помічника, який буде самостійно реалізовувати моніторинг чатів. Проєкт зробить можливим швидкий пошуку завдань, автоматичну генерацію звітів, оцінку психологічного стану учасників, складання рейтингів активності.

КПІЗ_2026

1 ПРИЗАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Програмне забезпечення призначене для автоматизації аналітики історії ділового чату засобами штучного інтелекту. Воно працює у вигляді Telegram-бота. Учасники отримують результати аналізу прямо в чаті – без переходу в окремі програми чи сайти.

Система бере на себе одразу кілька аналітичних завдань. Щодня вона готує звіти, вираховує рейтинги активності учасників, а ще – відповідає на прямі запити до ШІ, при цьому враховуючи контекст розмов у чаті. Окрім того, робить PDF-файли з поглибленим аналізом переписок.

Взаємодія з ботом відбувається через Telegram API шляхом використання слеш-команд та кнопок вбудованого меню клавіатури, роблячи процес використання бота інтуїтивно зрозумілим навіть для тих хто вперше стикається з таким. Кожне повідомлення проходить через попередню обробку: виділення часу повідомлення, ім’я користувача, ідентифікатор чату, а сам текст повідомлення проходить обробку на регулярні вирази.

Використання мовної моделі забезпечує якісний аналіз даних. Вона розуміє українську мову та знайома з українською культурою. Нейромережа за рахунок великої бази даних на якій вона була навчена значно ефективніше справляється з задачею аналізу даних, коли це стосується не просто пошуку ключових слів, а розуміння контексту речення в якому це слово знаходиться

Функції бота не обмежуються накопиченням даних – він також виконує їх фільтрацію та готує звіти щодо завдань, термінів і активності. При цьому користувачам не потрібне тривале навчання, тому інтеграція бота в будь-який робочий чат є досить простою справою.

КПЗ_2026

1.2 Область застосування

Розроблений чат-бот може стати в пригоді будь-де, де люди працюють у Telegram-групах і потребують автоматичного аналізу своїх дискусій.

Робочі команди, виступають головним середовищем, для якого призначено бота. У межах цих команд постійно триває обговорення завдань, розподіляються обов'язки та встановлюються терміни. Завдання бота – допомогти не губити дійсно важливу інформацію в потоці повідомлень, а саме: фіксувати, хто, кому й до якого числа щось пообіцяв.

Навчальні курси. Своєю чергою, викладачі теж можуть використовувати бота з метою аналізу активності учнів. відслідковування виконання завдань, аналізувати їх реакції на новий матеріал. Вчителі та викладачі можуть використовувати рекомендації від аналізу для підготовки навчальних матеріалів, щоб бути упевненим, що студенти засвоїли матеріал і не стикаються з труднощами. Аналіз на емоційного стану учнів може також допоможе викладачам у вчасно звертати увагу на конфлікти між учнями ще на ранніх стадіях їх зародження .

Громадські групи. Тут обговорення можуть тривати годинами. І часто важко згадати, хто що запропонував і до чого врешті дійшли. Бот складає короткі підсумки дискусій. Він виділяє пропозиції кожного учасника. Жодна ідея не губиться.

А взагалі – будь-який чат, де ведуться серйозні розмови**.** Зустрічі, задачі, терміни, дедлайни. І водночас люди хочуть спілкуватися вільно, без огляду на «контролера». Бот не втручається. Він працює у фоні. Тільки коли його викличуть. Він зберігає історію, готує звіти, показує кожному користувачеві тільки те, що стосується особисто його. Система достатньо гнучка. Вона стане в корисною скрізь, де потрібні короткі підсумки щоденних дискусій без ручної праці.

КПІЗ_2026

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми кваліфікаційної бакалаврської роботи

Серед аналітичних платформ варто відзначити ChatPulse. Це система для інтелектуального моніторингу груп у Telegram, Discord та інших месенджерах. Головне її завдання – збирати великі обсяги даних і потім виокремлювати з них суттєву інформацію.

Для аналізу текстових повідомлень у ChatPulse застосовуються методи обробки природньої мови (NLP), що дозволяють автоматично класифікувати та групувати інформацію. Оцінюється зміст діалогів, автоматично групує повідомлення за темами. Менеджменту не доводиться перечитувати всю історію листування – достатньо поглянути на згруповані теми.

ChatPulse пропонує користувачам кілька аналітичних інструментів. Бот готує стислі звіти за обраний період, а також виконує аналіз тональності повідомлень: відстежує емоційне тло учасників, щоб вчасно зафіксувати ризик виникнення конфліктів або стресу. Крім того, передбачено тематичне кластерування та метрику залученості.

Обробка повідомлень виконується автоматизовано без необхідності ручного перегляду всієї історії листувань. Під час аналізу тексту визначаються ключові сутності, а також оцінюються емоційні характеристики повідомлень. На основі цих даних будуються діаграми, які показують «тенденції» чату. Окремий показник – індекс емоційної напруженості. Він дозволяє вчасно виявити негативні емоції та конфлікти.

ChatPulse орієнтований на підвищення ефективності груп і автоматизує процеси. Він застосовує ШІ для обробки інформації, видалення спаму та підсвічування важливих даних. Але незважаючи на потужний функціонал, у

КОПЗ_2026

ChatPulse відсутня здатність до глибшої інтеграції ШІ для відслідковування та аналізу кожного окремого користувача.

Lindy.ai – це спеціалізований ШІ-агент нового покоління відповідальний за глибоку інтеграцію у життя людини. Він автоматизує робочі процеси у форматі природного листування. Система взаємодіє з електронною поштою, месенджерами, календарем, нотатниками. Також – з редакторами текстів, таблиць і зображень. Завдяки інтеграції з різними сервісами система забезпечує централізовану роботу з даними користувача. Усі вони структуровані за допомогою ШІ. Рутинні операції автоматизуються повністю.

Користувач має змогу створювати ШІ-агентів, які функціонують одночасно з кількома програмами. Агент збирає дані з електронної пошти, нотатників, згодом готує звіт і відправляє його електронною поштою. Серс підтримує глибоку інтеграцію з популярними платформами: Google Календар, Google Drive, Google Sheets, Youtube, Amazon, Microsoft Azure DevOps. Агент перевіряє календар, виявляє заплановані події, сприймає комунікації з клієнтами та надсилає регулярні нагадування. Крім того, система підтримує автоматичні нагадування про заплановані події та завдання. Налаштовувати агентів можна за допомогою текстових інструкцій – для цього не потрібно вручну програмувати складну логіку. Завдяки цьому поріг входження для користувачів стає значно нижчим.

Незважаючи на високий рівень автоматизації та зручність налаштування, Lindy.ai має обмеження. Вона орієнтована переважно на індивідуальну продуктивність. Принцип роботи – лінійна взаємодія «користувач-агент». Водночас аналіз складної взаємодії між колегами всередині чату – це «вузьке місце» системи. Глибокого аналізу динаміки групових повідомлень у форматі «користувач-користувач» Lindy.ai не забезпечує.

Metricgram – платформа для роботи з великим спільнотами Telegram, які можуть мати по декілька тисяч підписників. Її завдання - розвиток, контроль та монетизація. Але на відміну від персональних асистентів, Metricgram

КОПЗ_2026

орієнтована на адміністраторів великих груп і власників бізнес-каналів. Її інструменти: глибока аналітика й автоматизація взаємодії з учасниками.

Адміністратори отримують доступ до персонального кабінету на веб-сторінці. Там вони можуть переглядати всю статистику каналу або групи через зручний інтерфейс. Система сама формує щоденні звіти. Відстежує стан спільноти. Аналізує теми, про які говорять люди. Додатково система формує рекомендації щодо подальшого розвитку спільноти на основі зібраної статистики.

У свою чергу, Metricgram дає змогу успішно відстежувати тенденції всередині спільноти, а також теми, що обговорюються, й реакції учасників. Адміністратор має доступ до історії з усіх повідомлень і може здійснювати пошук в них. Додатково система формує щоденні звіти, автоматично надсилає новини чи нагадування та створює рейтинги найбільш активних учасників.

Є інтеграція платіжної системи Stripe, що спрощує доступ до платних груп. Винятковою функцією є «геймифікація». Бот автоматично нараховує бали за повідомлення та реакції для кожного учасника групи. Це стимулює активність через рейтинги та нагороди,та підтримує інтерес учасників до групи. Є можливість підключення та налаштування власного ШІ-асистента. Модель може адаптуватися до особливостей конкретної спільноти шляхом аналізу накопичених даних.

Суттєвим недоліком системи є концентрування на базовому та поверхневому аналізі. Вона аналізує усю спільноту загалом, але заглиблюється в деталі. Metricgram пристосований до роботи у великих спільнотах на декілька тисяч людей, але в той же час не проводиться аналіз кожного окремого індивіда. Metricgram залишається хорошим інструментом в руках адміністратора, але не для користувачів.

КПІЗ_2026

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Для розробки програмного забезпечення для аналітики бізнес-чатів засобами штучного інтелекту було обрано мову програмування Python 3.13, який має набір комфортних для розробки властивостей:

- Зрозумілість синтаксису Python порівняно з іншими мовами значно полегшує його вивчення. А це, робить довготривалу підтримку системи, її зміни та масштабування менш складними.

- Розвинена екосистема: наявність великої кількості спеціалізованих бібліотек для інтеграції з ШІ-моделями та сторонніми платформами.

- Висока швидкість ітерації: Лаконічність мови дозволяє швидко реалізовувати та тестувати нові функції аналізу тексту.

Для реалізації проєкту мовою Python було використано наступний стек бібліотек:

Aiogram 3 – сучасний асинхронний фреймворк для взаємодії з Telegram API. Він підтримує машину станів, фільтри, клавіатури, callback-запити. Цей фреймворк буде зручним для роботи через його асинхронну природу яка дозволяє користувачам продовжувати взаємодіяти з ботом, поки сам бот займається обробкою попередніх запитів.

OpenAI API – бібліотека взаємодії з API обраної великої мовної моделі. Бібліотека є гнучкою, через стандартизований інтерфейс OpenAI, що дозволяє швидко та безболісно перемикати модель і без великих змін у коді. А також забезпечує більш тонке налаштування парметрів нейромережі – кількість максимальних токенів, температуру.

База даних SQLite – історія чату, метадані повідомлень та звіти від ШІ зберігаються у спеціальній базі даних. SQLite є легшою для використання, адже не потребує окремого сервера та водночас є простою у використанні та надійною. Для збереження повідомлень цього буде достатньо.

КПТЗ 2026

APScheduler – планувальник щоденних звітів. Запускаються автоматично з запуском бота. Не потребує втручання з боку користувача. Завдяки сумісності з aiogram планувальник працює в асинхронному режимі, та не заважає виконанню завдань бота.

ReportLab – потужна бібліотека для створення звітів формату PDF. Ця біблотека корисна для підготовки звіту, який можна завантажити на комп'ютер чи телефон та продивитись окремо.

Логування Logging – стандартний засіб для логування. Він дозволяє відстежувати поведінку бота та перевіряти стабільність його роботи.

Модуль regular expression – стандартний модуль регулярних виразів. Він забезпечує попередню обробку тексту повідомлень перед їх збереженням. Виділяє ключові слова та дати для їх коректного передання до великої мовної моделі.

Як основне середовище розробки, було обрано Visual Studio Code. Це сучасний та легкий, але потужний редактор коду, на який можна легко встановити необхідні для роботи з обраною мовою програмування розширення. Visual Studio Code має тривалу та стабільну підтримку, що робить його гарним редактором, який дозволяє легко встановити усі необхідні інструменти та почати роботу не витрачаючи час на довге налаштування редактора.

КПІЗ_2026

2.3 Розгорнута постановка завдання

Основним призначенням програмного забезпечення, що розробляється, є аналітика історії ділового чату в Telegram. Система має збирати повідомлення, обробляти, створювати на їх основі запити та засобами штучного інтелекту аналізувати їх, після чого користувачам надається стислий структурований звіт.

Вхідні дані. До системи надходять текстові повідомлення від учасників чату. Кожне повідомлення містить текст, ім'я користувача, його ідентифікатор

та час відправки. Мова повідомлень - українська. Усі вони зберігаються в базі даних разом із автоматично визначеними тегами та датами.

Методи обробки. Обробка здійснюється в кілька етапів:

- Перед аналізом кожне повідомлення підлягає попередній обробці за допомогою модуля регулярних виразів. Його мета виявити семантичні маркери. Прикладом таких маркерів є «треба», «дедлайн», «домовились», «проблема» та знак питання. Крім того, система розпізнає дати і потім розділяє повідомлення за тегами категорій: завдання, дедлайн, рішення, запитання, проблема або терміново.

- Аналіз через ШІ застосовується за запитом користувача система звертається до великої мовної моделі. Вона отримує історію чату і генерує аналітичний звіт. У звіт входить виділення завдань та дедлайнів, фіксація домовленостей, оцінка емоційного стану учасників, рекомендації.

- Після отримання повного звіту ініціалізується додатковий запит до ШІ з метою фільтрації важливого.. Ціль цього запиту - витягнути з повного звіту тільки конкретні факти, що включає: плани, рішення, дати, виконавці. Це зберігається як «важливі події» чату.

Результати роботи представлені формуванням системою кількох типів вихідних даних:

- Текстовий звіт у чаті (на вимогу користувача за командою /analyze).

- Рейтинг активності учасників (команда /rating).

- Персоналізований перелік подій для конкретного користувача (/myevents).

- Щоденний автоматичний звіт.

- PDF-документ із повною аналітикою за вибраний період (команда /report).

Код написано на Python 3.13. Використано бібліотеки: aiogram 3, openai, SQLite, APScheduler, ReportLab, re та logging. Розробка велася у середовищі Visual Studio Code.

КбПЗ_2026

Оскільки система є Telegram-ботом, графічного інтерфейсу в класичному розумінні немає. Замість нього – текстовий чат-інтерфейс. Користувач взаємодіє з ботом через:

- Слеш-команди: /analyze, /rating, /events, /myevents, /report, /ask, /clear та інші.

- Кнопки головного меню (відкриваються командою /menu): «Аналіз ШІ», «Рейтинг активності», «Важливі події», «Події для мене», «Очистити історію».

- Inline-клавіатури для навігації по подіях та підтвердження дій.

Додаткові функції. Система автоматично зберігає історії чату в базі даних. Дозволяє очистити історії. Підтримує прямі запити до ШІ через команду /ask. Веде логування помилок для контролю роботи чату.

Обмеження. Система розрахована на невеликі робочі групи до кількох десятків учасників. Вона не призначена для аналізу тисяч повідомлень у великих публічних каналах. Також бот не зберігає медіа файли– аналізується тільки текст.

КбПЗ_2026

3 ОПИС І ОБГРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Розроблена система аналітики ділових комунікацій реалізована у форматі Telegram-бота та призначена для автоматизованої обробки історії повідомлень у групових чатах. Бот працює на основі бібліотеки Aiogram, яка забезпечує асинхронну обробку вхідних запитів та підтримує сучасні механізми фільтрації повідомлень, такі як диспетчер подій та машина станів. Архітектура програмного забезпечення побудована за клієнт-серверним принципом, де Telegram виступає середовищем взаємодії з користувачем, а серверна частина відповідає за обробку повідомлень та виконання аналітичних операцій. У ролі серверної частини виступає Python-скрипт, що постійно виконується на хостингу або локальному сервері, підтримуючи з'єднання з Telegram через довготривале polling-з'єднання.

Процес функціонування системи можна розділити на кілька ключових етапів:

1) Збір та первинна обробка даних.

Система в режимі реального часу безперервно відстежує вхідний потік повідомлень. Це реалізовано за допомогою обгортки функції – декоратора, який перехоплює всі текстові повідомлення, що надходить до бота. Кожне з них проходить етап парсингу, у ході якого витягуються такі метадані, як ідентифікатор користувача, ім'я, текст повідомлення та часова мітка. Ці дані формують словник, який тимчасово зберігається в оперативній пам'яті у глобальному словнику для пришвидшення доступу. Робота програмного забезпечення організована навколо асинхронного циклу подій – це дозволяє використовувати неблокуючі виклики та паралельно обробляти запити. Завдяки асинхронній взаємодії, бот здатен одночасно приймати нові повідомлення, виконувати аналіз попередніх та відповідати на команди без

КБПЗ_2026

затримок. У процесі аналізу тексту визначаються ключові семантичні маркери та згадані у повідомленнях дати. Результат – теги «task», «deadline», «decision», «question», «problem», «urgent» та дати у форматі YYYY-MM-DD зберігаються разом з повідомленням. Якщо жодне ключове слово не знайдено, повідомленню присвоюється тег «other», що дозволяє фільтрувати неінформативні розмови під час наступних етапів аналізу.

2) Збереження даних у базі SQLite.

Усі повідомлення зберігаються в базі даних SQLite. Це легковагова файлова база даних, яка не потребує окремого серверного процесу, що робить її ідеальним вибором для компактних аналітичних систем. Це забезпечує надійне та структуроване зберігання, швидкий пошук за часом та можливість виконувати SQL-запити. Наприклад, функція get_messages_since_date() використовує параметризований запит для отримання повідомлень за визначений період. Дані зберігаються у таблицях messages (текст, метадані, теги, дати) та ai_reports (звіти ШІ). SQLite не потребує окремого сервера – це один файл, який легко переносити. При запуску історія завантажується з бази даних через функцію load_all_messages(). Ця функція повертає словник, де ключем є chat_id, а значенням – список повідомлень, кожне з яких подано у вигляді вкладеного словника з основними ключами user_name, user_is, timestamp, text, tags, extracted_dates. Така структура дає змогу швидко отримувати доступ до будь-якого повідомлення під час аналізу.

3) Інтелектуальний аналіз засобами штучного інтелекту.

Основне аналітичне навантаження покладається на модуль взаємодії з великою мовною моделлю. Функціонування аналітичного блоку активується за запитом користувача або автоматично за розкладом через планувальник APScheduler, який автоматично запускає створення щоденних звітів.

Щоб звернутися до ШІ із запитом, система спочатку формує контекст. Для цього використовуються не тільки попередні звіти, а й нові повідомлення, що з’явилися після останнього аналізу. Такий підхід дає ШІ можливість розуміти довгострокові домовленості, зокрема й ті, які обговорювалися ще

КПІЗ_2026

тиждень тому. У межах аналізу історії повідомлень мовна модель використовується для: фільтрації та категоризації, виокремлення з масиву повідомлень лише суттєвої інформації (події, рішення, домовленості), виявлення дат та дедлайнів, автоматичне розпізнавання часових обмежень, згаданих у контексті бесіди. Окремо виконується оцінювання емоційного фону дискусії, що дозволяє виявляти потенційно конфліктні ситуації або підвищений рівень напруженості у спілкуванні. Після отримання повного звіту, система запускає окремий запит до ШІ, який знаходить та витягує лише конкретні факти, плани, рішення, дати, виконавців. Це зберігається окремо в таблиці ai_reports, як «важливі події» чату.

4) Генерація аналітичних звітів.

Система генерує звіти кількох типів. За командою «/analyze» бот надсилає у чат текстовий звіт, що містить аналіз останніх повідомлень. За командою «/report» формується PDF-документ за допомогою бібліотеки ReportLab з розширеним контекстом за деякий період та включає у себе аналіз подій, активності, тенденцій та емоційного стану учасників. APScheduler запускає аналіз останніх повідомлень кожного дня о конкретній годині, наприклад о 21:00, без участі користувача. Усі звіти зберігаються в базі даних разом із датою створення, що дозволяє системі великої мовної моделі будувати історію аналітики та порівнювати показники за різні періоди та формувати більш коректні звіти.

5) Інтерфейс користувача.

Взаємодія з системою реалізована через об’єкти класів ReplyKeyboardMarkup та InlineKeyboardMarkup, що мають вигляд кнопок у чаті. Учасники легко орієнтуються у функціоналі без необхідності використання сторонніх платформ чи сайтів, забезпечуючи доступність отримання результатів. Додатково засобами Telegram API підтримуються слеш-команди: /analyze, /rating, /events, /myevents, /report, /ask, /clear. Відкриття головного меню клавіатури за командою /menu.

КПІЗ_2026

3.2 Розробка структурної схеми

Структурна схема є етапом проєктування, що визначає склад компонентів програмного забезпечення у їхній сукупності та ієрархію їх взаємодії, а також відображена на рисунку 3.1. Вона дозволяє візуалізувати загальну логіку розподілу функціональних обов'язків між окремими модулями та зовнішніми сервісами. Такий підхід проєктування дає змогу чітко визначити межі відповідальності кожного компонента, що спрощує подальшу модернізацію системи та додавання нових функцій. Кожен компонент на схемі відповідає конкретному класу або функції у вихідному коді, що забезпечує пряме відображення архітектури в реалізації.

Доступ до бота здійснюється з клієнтських пристроїв – смартфони, планшети або ПК, тобто з будь-яких пристроїв, які підтримують роботу Telegram. Через ці пристрої користувач надсилає запити та повідомлення. Взаємодія при цьому будується на основі офіційного інтерфейсу месенджера Telegram. Цей інтерфейс є стандартизованим для всіх платформ, що забезпечує однаковий досвід користувача незалежно від того, чи використовує він мобільний застосунок або декстопний клієнт.

Платформа Telegram приймає введені дані від користувача та за допомогою Telegram API обробляє їх, а потім пересилає до програмної частини бота. Цей вузол забезпечує обмін інформацією, дозволяючи отримувати повідомлення та команди від користувача та повертати йому згенеровані відповіді безпосередньо у чат. У відповідь бот надсилає структуровані JSON-подібні об’кти, які Telegram трансформує у звичні для користувача текстові повідомлення, кнопки або документи.

Внутрішня архітектура розділена умовно на два основні логічні блоки: Клієнт та Сервер. Таке розділення відповідає принципам клієнт-серверної архітектури, де клієнтська частина відповідає за взаємодію з користувачем, а серверна – за логіку та зберігання даних.

КОПЗ_2026

Блок «Клієнт» відповідає за обробку вхідного потоку повідомлень у реальному часі. Диспетчер подій aiogram фільтрує та розподіляє отримані повідомлення залежно від їх вмісту на команди, натискання кнопок або простих текстів. Диспетчер використовує механізм хендлерів – функцій, обгорнутих декораторами. Кожен хендлер має власні фільтри, що дозволяє точно маршритузувати повідомлення без зайвих перевірок у коді. Обробник команд та повідомлень містить логіку реакцій на отримані повідомлення чи команди, визначаючи, яку функцію запустити. Машина станів (FSM context) дозволяє боту запам’ятовувати етап діалогу з користувачем, як наприклад перегляд історії повідомлень та підтвердження дій.

Блок «Сервер» реалізує всю внутрішню інтелектуальну та аналітичну логіку та є реакцією на команди користувача, відфільтровані диспетчером та запущені обробником команд. Цей блок не взаємодіє безпосередньо з Telegram API, а лише отримує дані від блоку «Клієнт» і повертає результати обробки назад для відправки користувачеві.

КПЗ – 2026

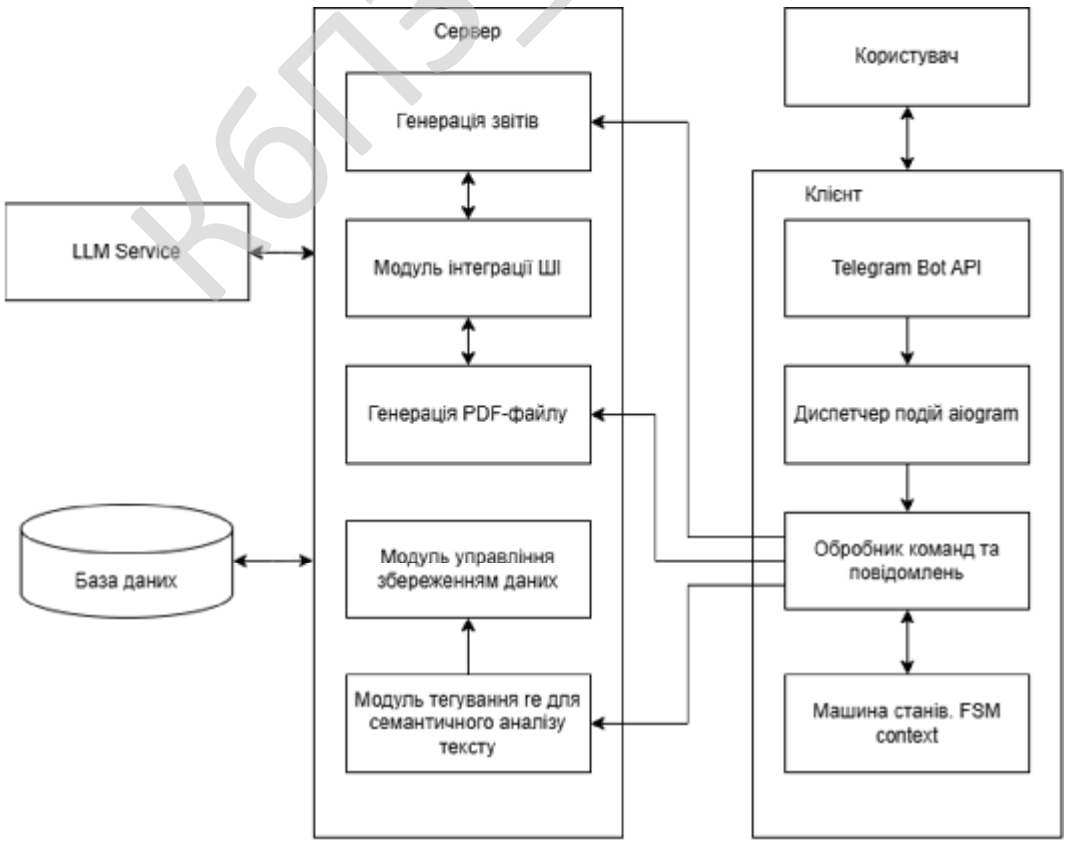


Рисунок 3.1 - Структурна схема

У генерації звітів спочатку формується запит до ШІ. Потім збирається історія повідомлень та надсилається через модуль інтеграції ШІ до зовнішнього LLM Service для подальшої обробки. Передача даних до ШІ відбувається за допомогою POST-запиту з використанням протоколу HTTPS, що робить процес передачі даних захищеним та надійним без втрати даних. Генерація звітів отримує цю відповідь і перетворює її на текст, який бот надсилає користувачеві.

Генерація PDF-звітів працює за схожим принципом з генерацією звітів, але замість повернення звичайного повідомлення, повертає структурований звіт-аналіз у формі PDF-файлу зручного до перегляду на будь-якому пристрої без втрати форматування. Процес створення PDF включає додаткову реєстрацію шрифту DejaVu для коректного відображення української мови, формування документа за допомогою ReportLab і надслається через метод message.answer_document() у вигляді файла для завантаження.

Для інтеграції ШІ, формуються спеціальні запити-інструкції, які надсилаються до великої мовної моделі через API. Кожен запит складається з двох частин: system_prompt – системна інструкція, яка визначає роль моделі, формат відповіді, обмеження, що є важливим для коректного аналізу даних моделлю, та user_prompt – безпосередній контент, який містить історію чату, теги, дати, та попередні звіти.

Модуль тегування regular expression виконує попередню обробку кожного простого повідомлення. Він використовує регулярні вирази для виділення ключових слів з тексту та дати. Визначення тегів відбувається за попередньо визначеним словником відповідності ключових слів. Результат: теги для кожного повідомлення і дати у зручному форматі YYYY-MM-DD. Після тегування дані передаються до модуля управління збереженням даних.

Модуль управління збереженням даних – відповідає за роботу з базою даних SQLite, завантаження історії, очищення даних. Усі операції з базою даних виконуються через функції окремого файла-модуля database.py. База

КПІЗ_2026

даних використовується для збереження даних оброблених повідомлень та створених звітів для їх використання в подальшому.

LLM Service – це зовнішній сервіс, який здійснює обробку природної мови, виділяючи з тексту завдання, дедлайни та психологічні профілі учасників і повертає відповіді до серверної частини для подальшого форматування фінальної відповіді користувачеві.

3.3 Розробка функціональної схеми

Відправною точкою будь-якої взаємодії з ботом у межах цієї системи є дія користувача, який через Telegram надсилає повідомлення чи команду. Ця дія може бути як текстовим повідомленням, так і натисканням inline-кнопки або командою, розпочатою з символу «/». Усі ці варіанти обробляються по-різному, але проходять через єдиний вхідний потік. Отримані дані потім передаються за допомогою API месенджера до тієї частини бота, що відповідає за їх обробку у подальшому. У даній реалізації використовується довготривале polling-з'єднання, яке підтримується бібліотекою Aiogram. Уся ця послідовність дій представлена на рисунку 3.2 функціональної схеми.

Планувальник автоматичних задач – це окремий компонент, який запускає періодичну дію щоденного звіту без участі користувача. У коді цей компонент реалізовано через бібліотеку APScheduler, а саме – клас AsyncIOScheduler, який інтегрується з асинхронним циклом подій asyncio. Сам планувальник запускається методом scheduler.start() під час ініціалізації бота у функції main() під час першого запуску бота та продовжує працювати у фоновому режимі до зупинки програми.

Обробка вхідних повідомлень, як перший функціональний компонент – це модуль обробки команд, надісланих користувачем або згенерованих системою. У коді цей модуль представлений набором хендлерів. Диспетчер автоматично маршрутизує кожне вхідне повідомлення до відповідного хендлера залежно від його типу та вмісту. Він визначає точно, чим є отримане

КОПЗ_2026

повідомлення: команда, звичайне повідомлення або сигнал від планувальника. Після чого запускає відповідні сценарії реакції. Сигнали від планувальника є прямими викликами функцій, а не повідомленнями через Telegram, що дозволяє уникнути затримок, пов’язаних з мережевою передачею даних.

Якщо обробник розпізнав повідомлення, що не містить команду, тобто є звичайним текстовим повідомленням, то одразу запускається процес обробки цього повідомлення крок за кроком. Запускається попередня обробка - виокремлення метаданих, таких як ідентифікатор користувача, його ім’я та часова мітка, тегування тексту за допомогою регулярних виразів. Після успішної обробки, дані повідомлення повністю зберігаються в базі даних SQLite для подальшого використання в аналітичних звітах.

Коли користувач надсилає запит-команду, яка використовує засоби ШІ, система звертається до модуля інтеграції API нейромережі. Модуль формує запит до великої мовної моделі. Запит містить історію чату збережену в БД у структурованому вигляді та попередні звіти для кращого розуміння контексту поточного діалогу. Сервіс LLM отримує сформований запит, обробляє його та повертає структурований звіт.

Після отримання результатів аналізу від зовнішнього сервісу виконується додаткова обробка отриманих даних для забезпечення їх безпечного повернення користувачу без втрати даних. Це включає в себе налаштування розмітки та розділення звіту на кілька послідовних повідомлень, через обмеження в кількості символів одного повідомлення месенджеру Telegram (максимум 4096 символів). Сформовані результати передаються назад користувачам через Telegram API. Якщо ж аналіз було викликано через команду /report, то формується звіт у форматі PDF замість звичайного текстового повідомлення, що дозволяє зберігати форматування та зручніше переглядати великі обсяги даних.

Отриманий звіт, який пройшов попередню обробку, повертається назад в чат користувачеві у читабельному повідомленні або PDF-форматі. Звіти

КПІЗ_2026

зберігаються у БД для використання в наступних аналізах та для формування довгострокової історії важливих подій.

Функціональна схема ілюструє повний цикл функціонування системи: від отримання повідомлень і команд користувачів через Telegram API до їх обробки, збереження, аналізу методами ШІ, а потім – повернення користувачеві структурованого звіту в межах того ж Telegram-чату.

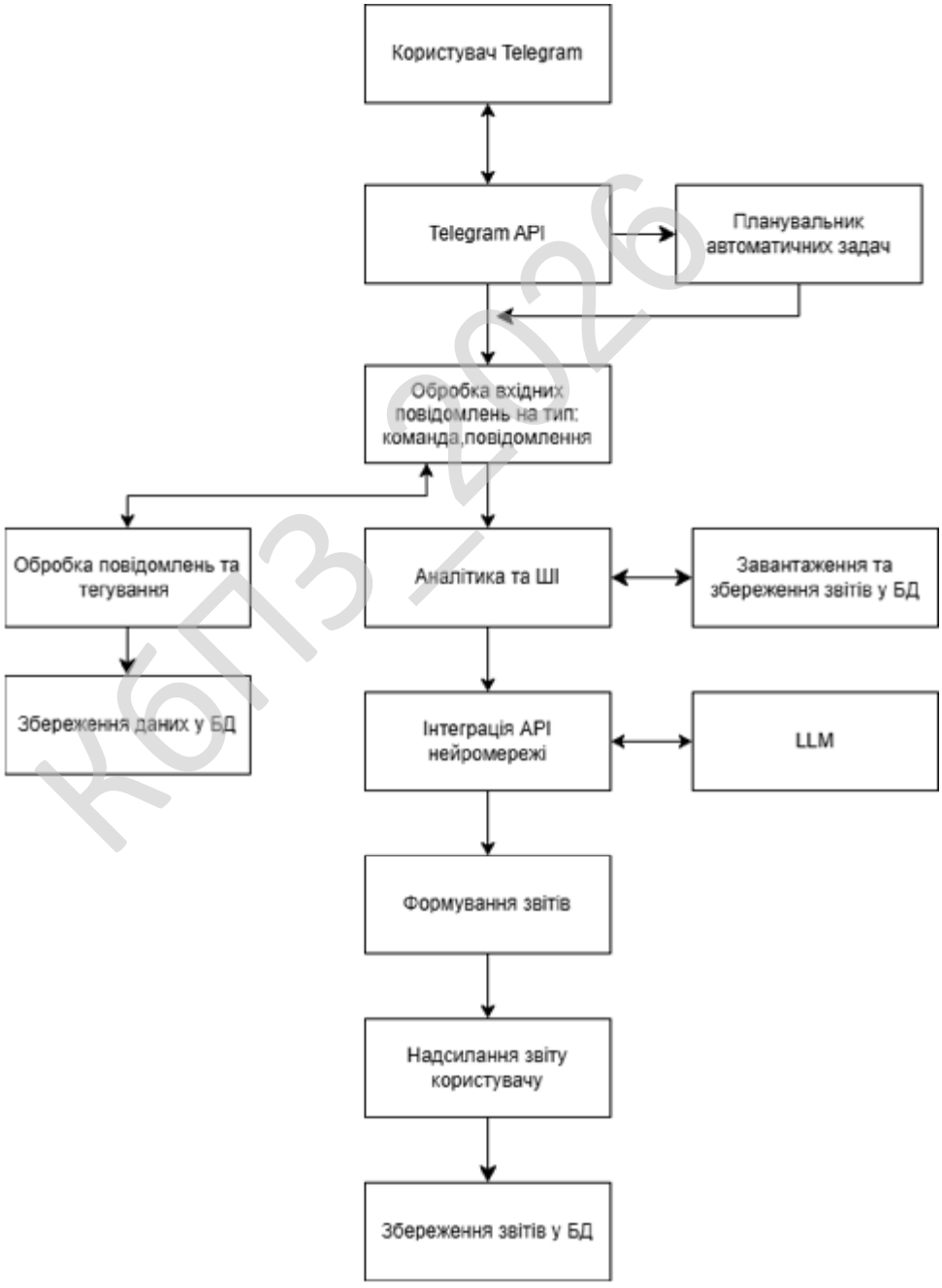


Рисунок 3.2 – Функціональна схема

КбПЗ–2026

3.4 Розробка діаграми процесів

Діаграма процесів представляє собою графічне представлення дій, які виконуються у суворій послідовності у програмі для досягнення кінцевого результату. Вона відображає не просто набір функцій, а саме потік виконання- від моменту отримання вхідних даних до моменту видачі готового результату користувачу, або збереження результату в базу даних. Діаграма показує наочно , як процеси взаємодіють один з одним, які етапи проходять та які функції виконуються. Діаграма процесів для демонстрації логіки роботи програми показана на рисунку 3.3.

Користувач взаємодіє з ботом шляхом надсилання звичайного текстового повідомлення або команди. Це може бути як просте питання у чаті, так і спеціальна службова команда на кшталт «/analyze», «/users», «/myevents», «/report». Після отримання запиту система передає його до механізму обробки подій, який визначає тип вхідних даних у реальному часі та обирає відповідний сценарій подальшої обробки цього запиту. На цьому етапі відбувається перше розгалуження алгоритму: залежно від того, чи є повідомлення звичайним текстом або командою, обирається різний шлях виконання.

У випадку надходження звичайного текстового повідомлення виконується автоматичний збір службової інформації, зокрема ідентифікатора користувача, його імені та часової позначки повідомлення. Ці метадані є критично важливими для подальшої аналітики, оскільки дозволяють визначити, хто і коли написав те чи інше повідомлення. Паралельно здійснюється попередній аналіз текстового вмісту з подальшим формування тематичних тегів на основі регулярних виразів. Після завершення обробки повідомлення разом з метаданими та результатами тегування записується постійно до бази даних SQLite. На цьому обробка звичайного повідомлення завершується – система не надсилає жодної відповіді користувачеві, а лише зберігає дані для майбутніх аналізів.

КОПЗ_2026

При отриманні службової команди, наприклад «/analyze» або «/report» чи іншої подібної, система запускає модуль аналітичної обробки даних. На відміну від попереднього сценарію, тут система не просто зберігає дані, а виконує складний багатоетапний аналіз із залученням засобів штучного інтелекту. На Для роботи модуля з бази даних завантажуються попередні повідомлення користувачів та раніше сформовані звіти – вони потрібні, щоб побудувати контекст для ШІ. Це можна вважати однією із особливостей системи: вона використовує попередні звіти для побудови єдиного контексту, щоб система могла розуміти, які зміни сталися в попередніх домовленостях. Далі на основі цих даних формується детальний структурований запит, який через API передається до великої мовної моделі. Сам процес передачі даних відбувається через захищене HTTPS-з'єднання, що гарантує конфіденційність повідомлень, які аналізуються. Там відбувається аналіз і генерація відповіді.

Після отримання результатів аналізу система виконує наступні дії у визначеному порядку:

- Нормалізацію отриманої відповіді – налаштування розмітки тексту, розділення на кілька послідовних повідомлень, якщо потрібно. Це необхідно, оскільки Telegram має жорстке обмеження на довжину одного повідомлення – не більше 4096 символів. Якщо звіт виявився довшим, система автоматично розбиває його на частини та надсилає одна за одною.

- Якщо була використана команда /report, додатково автоматично створюється PDF-файл за допомогою бібліотеки ReportLab. Цей файл місить той самий результат роботи ШІ, але у структурованому вигляді, зручному для збереження або другу.

- Підсумковий результат повертається користувачу в чат без критичних затримок, залежно від швидкості роботи зовнішнього сервісу. У середньому відповідь від ШІ надходить за 5-15 секунд, що є прийнятним показником для аналітичних систем такого рівня.

- Обробка таймауту помилки – За рахунок часу відповіді система має додатковий шар захисту від помилки: якщо результат не прийшов за

КБПЗ_2026

відведений час, користувач отримає повідомлення про помилку. Це потрібно, щоб система не зависала і користувач одразу розумів, що стався збій в роботі зовнішнього LLM сервісу.

- Одним з останніх етапів у ланцюгу аналізу даних – є попередня фільтрація отриманого аналізу ШІ з метою очищення від зайвого та збереження результатів аналізу у БД для історії і створення контексту в майбутньому.

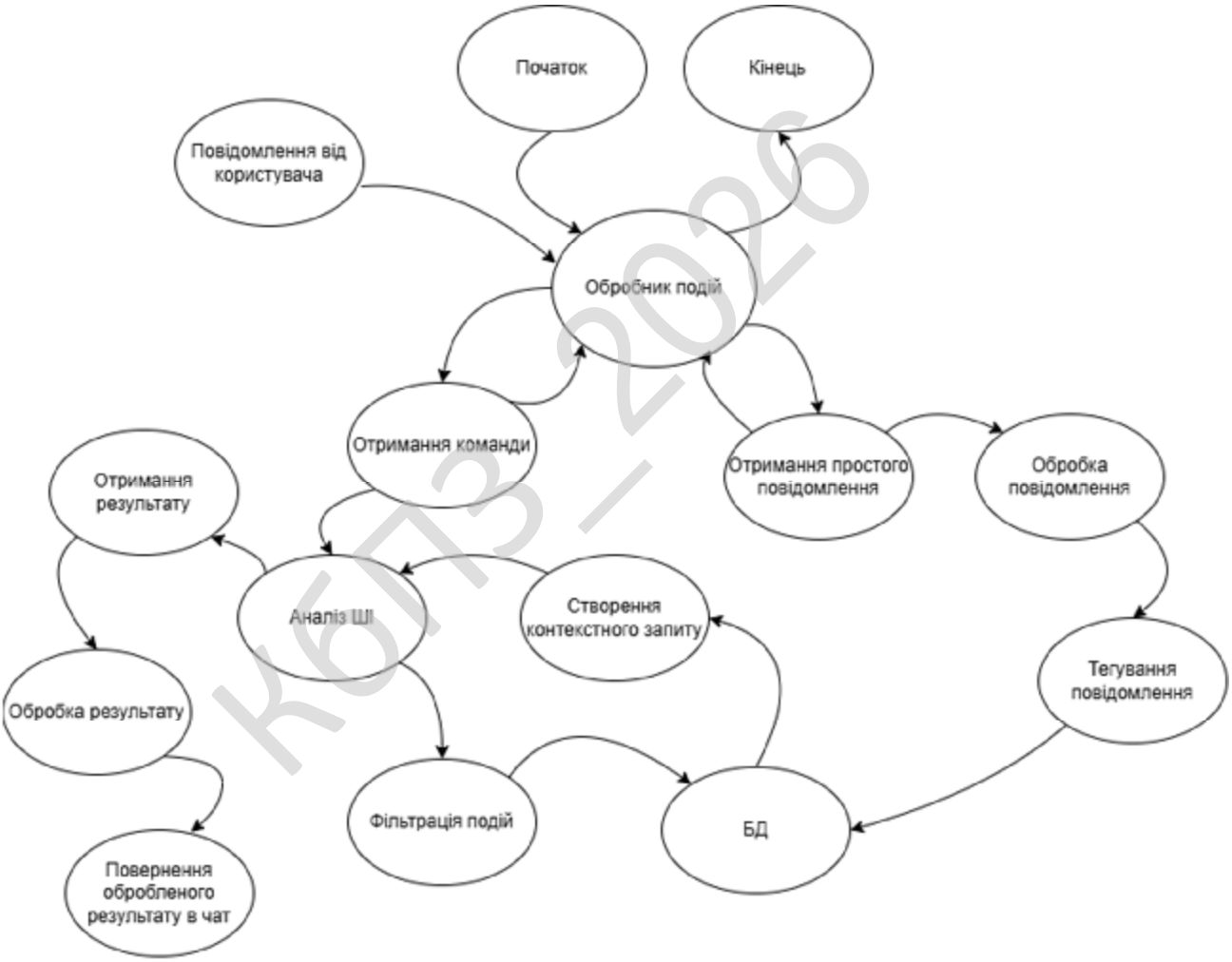


Рисунок 3.3 – Діаграма процесів

КБПЗ – 2026

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

На рисунку 4.1 зображено блок-схему алгоритму функціонування основної програми чат-бота.

1) Ініціалізація бібліотек. На початку роботи система імпортує необхідні бібліотеки: aiogram (асинхронний фреймворк для Telegram), openai (доступ до ШІ), SQLite (робота з базою даних), APScheduler (планувальник), ReportLab (генерація PDF), а також стандартні модулі regular expression, logging, json.

2) Завантаження бази даних. Програма виконує функцію init_db(), яка створює необхідні таблиці в SQLite, якщо вони ще не існують (таблиці messages та ai_reports). Далі викликається load_all_messages(), яка завантажує всю історію повідомлень з бази даних у пам'ять для швидкого доступу.

3) Запуск планувальника. Система активує планувальник автоматичних задач (APScheduler). Він налаштований на щоденне виконання функції generate_daily_report() о 21:00. Планувальник також перевіряє, чи не було пропущених запусків, і виконує їх при необхідності.

4) Очікування повідомлення. Бот переходить у режим очікування вхідних даних. Це реалізовано через асинхронний цикл dp.start_polling(bot).

5) Аналіз типу повідомлення. Коли надходить нова подія, система визначає її тип:

Якщо це команда, як наприклад: /analyze, /rating, /report, /ask, clear, запускається відповідний обробник команди. Команди обробляються через декоратори @dp.message(Command(...)).

КПІЗ_2026

Якщо це звичайне текстове повідомлення, активується функція collect_message(). Вона виконує збір метаданих (ім’я користувача, ідентифікатор, часова мітка), а потім передає повідомлення на тегування через модуль re.

6) Перевірка умови продовження роботи. Система перевіряє, чи надійшов сигнал про завершення роботи. Якщо ні – цикл очікування повторюється.

7) Зупинка бота. При отриманні сигналу завершення система коректно закриває з’єднання з базою даних, зупиняє планувальник і завершує роботу.

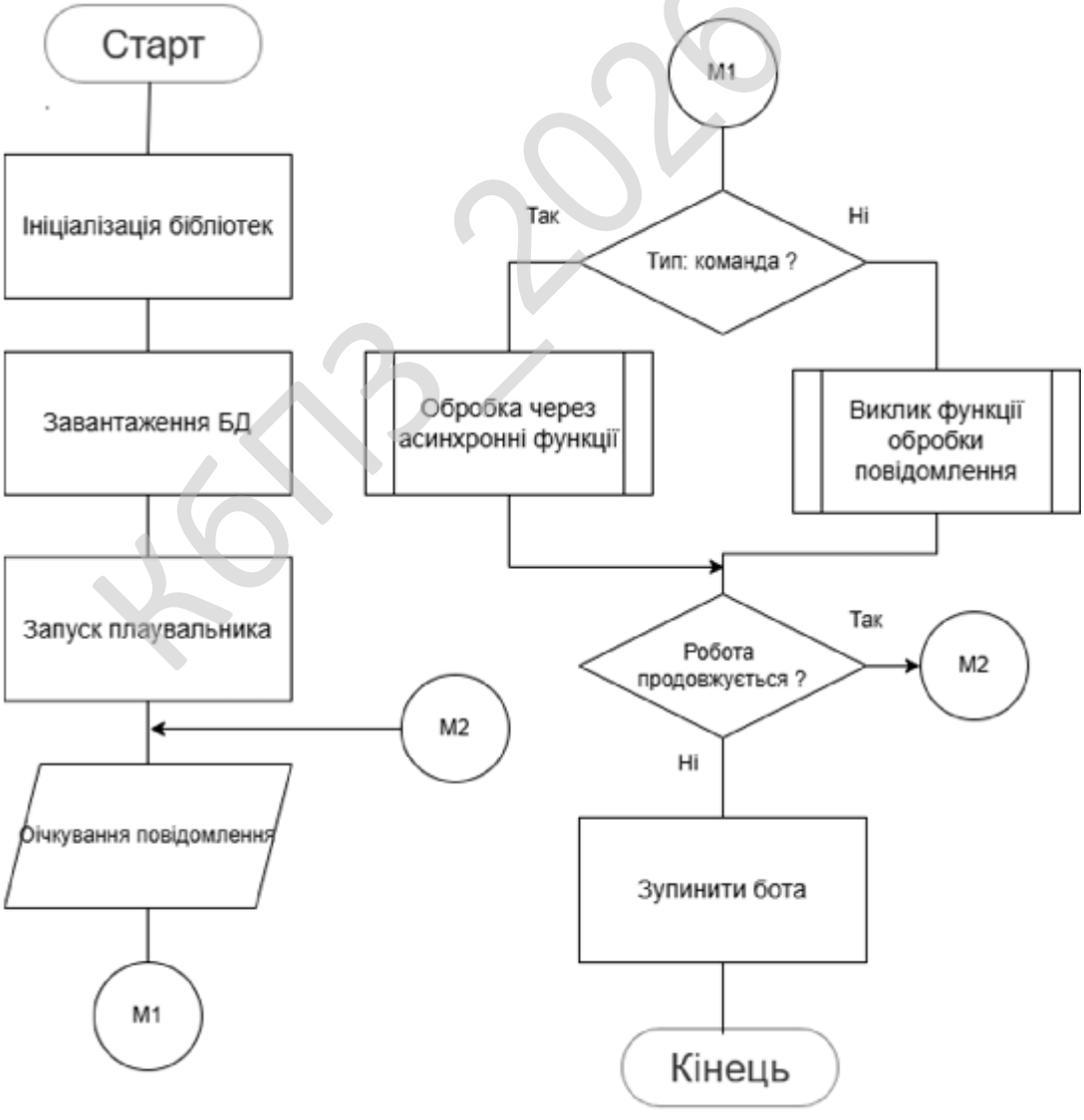


Рисунок 4.1 - Блок схема основної програми

КПЗ 2026

На рисунку блок-схеми 4.2 зображено алгоритм обробки простого текстового повідомлення від користувача.

Після отримання повідомлення від користувача, яке не є командою, запускається повний алгоритм обробки цього повідомлення для його збереження до бази даних.

1) Отримання повідомлення від користувача та його передача через Telegram API безпосередньо боту в обробник подій, де бот, не побачивши команди, почне виконання алгоритму.

2) Запуск функції collect_message(), де відбувається автоматичний збір основних метаданих повідомлення, що включає у себе: user_name, user_id, text, timestamp у текстовому вигляді.

3) Далі collect_message() викликає внутрішню функцію save_messages() і передає їй зібрані дані. Саме ця функція відповідає за підготовку даних та їх збереження у базу даних.

4) Усередині save_messages() викликається tag_message(), де відбувається аналіз тексту на ключові слова за допомогою бібліотеки regular expression у реальному часі. Якщо ключові слова знайдено – до значення tags додається відповідний тег. Якщо ключового слова не знайдено – додається тег “other”. Паралельно функція extract_date() виділяє з тексту дати, якщо такі є у тексті, і конвертує їх у формат ISO (YYYY-MM-DD) для подальшого використання у звітах ШІ.

5) Після завершення тегування та вилучення дат save_messages() формує SQL-запит на вставку і зберігає повідомлення разом із тегами та датами до таблиці messages бази даних SQLite. Завдяки цьому кожне повідомлення отримує структурований вигляд, що дає змогу штучному інтелекту краще і точніше орієнтуватися в тексті під час формування аналітичних звітів.

Таким чином, кожне текстове повідомлення проходить повний цикл обробки: від отримання від користувача через збагачення метаданими до збереження в структурованому вигляді для подальшого аналізу засобами штучного інтелекту.

КОПЗ_2026

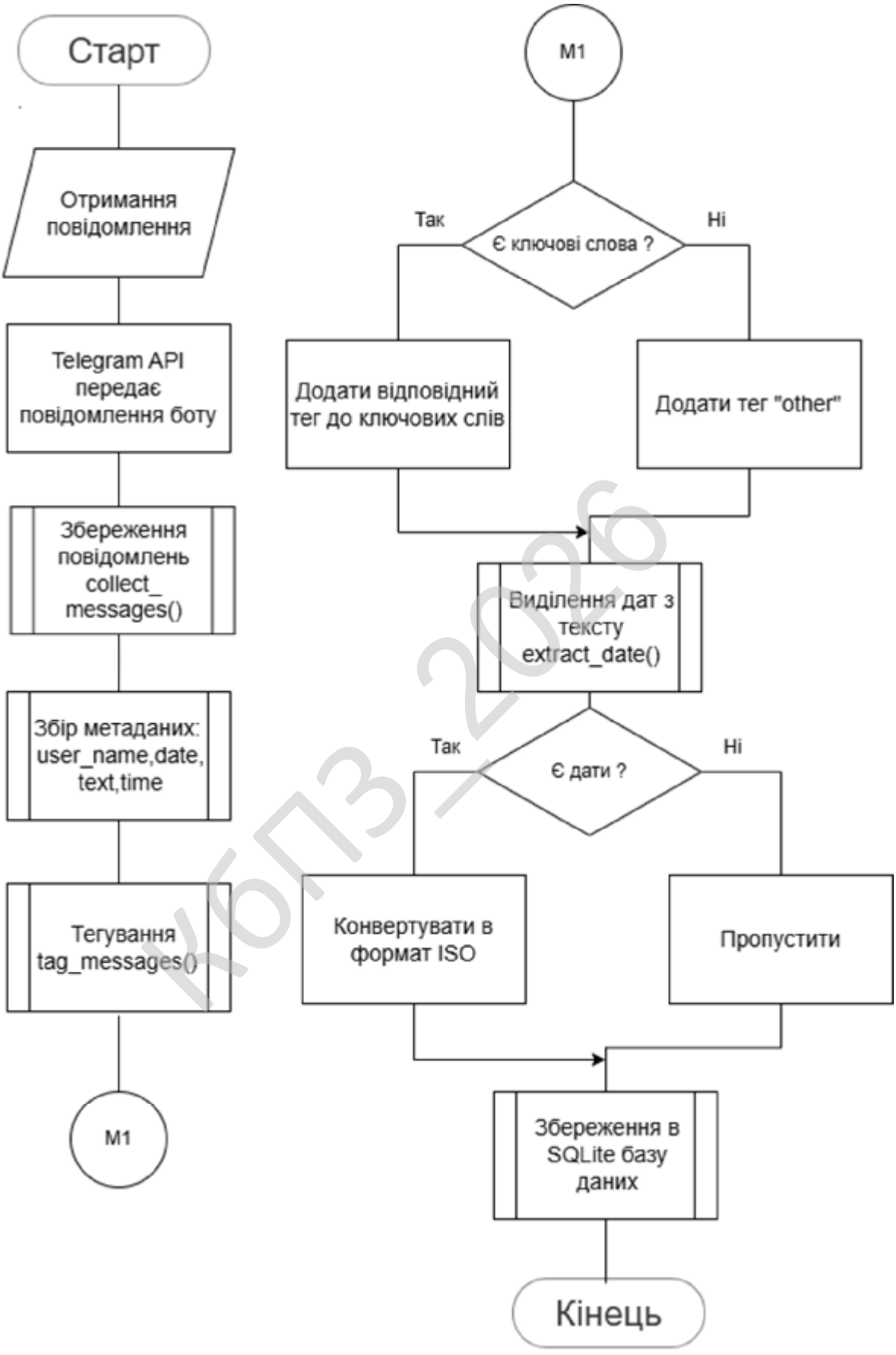


Рисунок 4.2 – Блок схема алгоритму зберігання повідомлень

КПЗ – 2026

На рисунку 4.3 зображено блок схему створення звіту за допомогою ШІ на основі накопичених повідомлень.

Команда /analyze запускає процес аналізу історії чату у поточний момент часу. Бот перевіряє нові повідомлення, які могли з’явитись після останнього звіту, формує контекст разом із попередніми звітами та отримує від ШІ згенерований структурований звіт.

1) Отримання команди /analyze від користувача у вигляді текстового повідомлення Telegram API автоматично передає її до диспетчера подій обробника бота.

2) Система викликає get_last_report() з параметром chat_id для уточнення дії для поточного чату. Ця функція звертається до таблиці ai_reports у SQLite та отримує останні три стислі звіти (якщо вони існують). Вони містять короткий виклад подій, завдань та дедлайнів за попередні періоди.

3) Якщо звіти є, система запам’ятовує дату останнього з них та викликає функцію get_messages_since_date() для завантаження нових повідомлень, які могли з’явитись з часу останнього створеного звіту.

4) Функція get_messages_since_date() повертає ті повідомлення, які з’явилися після дати створення останнього звіту, відсортовані за зростанням ідентифікаторів.

5) Система аналізує отриманий список повідомлень, якщо нових повідомлень немає, формується повідомлення про відсутність даних для аналізу. Якщо ж нові повідомлення є – вони додаються до контексту для подальшої обробки.

6) Формування контексту перед відправкою до ШІ. Нові повідомлення додаються до старих звітів. Система створює єдиний контекст, що включає нові повідомлення з часу останнього звіту та декілька останніх коротких звітів. Подібна структура допомагає боту аналізувати нові дані з урахуванням минулих домовленостей та довготривалого контексту спілкування.

7) Підготовка інструкцій для ШІ. Формуються два текстових блоки, які передаються моделі у вигляді масиву повідомлень:

КбПЗ_2026

- system_prompt – системна інструкція, яка описує ШІ його роль, формат відповіді, правила роботи з даними та обмеження (наприклад, ліміт кількості слів на кожного учасника);

- user_prompt – безпосередній запит від користувача, що містить сформований контекст у вигляді історії повідомлень із тегами та датами.

8) Запит до ШІ. Викликається функція get_ai_response(), яка надсилає підготовлені інструкції через API нейромережі. Модель аналізує текст, виявляє завдання, дедлайни, активність учасників та формує розгорнутий аналітичний звіт. Отриманий результат потрапляє у ai_content для подальшої обробки.

9) Фільтрація. Викликається функція filter_important_info(). Це окремий запит до тієї самої моделі, який витягує з повного звіту лише конкретні факти: плани, рішення, дати, виконавці, ігноруючи психологічні оцінки, рекомендації, залученість та активність учасників розмови.

10) Викликається функція save_filtered_report(), Яка зберігає відфільтрований звіт разом із повним звітом. Це дозволяє використовувати ці звіти для формування наступних аналізів.

11) Викликається функція send_safe_message(). Ця функція спрямована на перевірку довжини отриманого звіту. Якщо звіт перевищує 4096 символів (ліміт Telegram для одного повідомлення), функція розбиває звіт на кілька частин по 4000 символів кожна та послідовно надсилає їх із позначками «Частина 1/N», щоб користувачу було легше орієнтуватися у декількох послідовних повідомленнях. Виведення повного звіту користувачеві безпосередньо у чат.

Алгоритм забезпечує повний цикл аналізу чату: завантаження історії та попередніх звітів, інтелектуальну обробку за допомогою ШІ, фільтрація важливої інформації, збереження результатів та їх виведення назад у чат для користувача. Використання такого механізму допомагає ШІ пам’ятати про повний контекст роблячи звіти пов’язаними між собою.

КОПЗ_2026

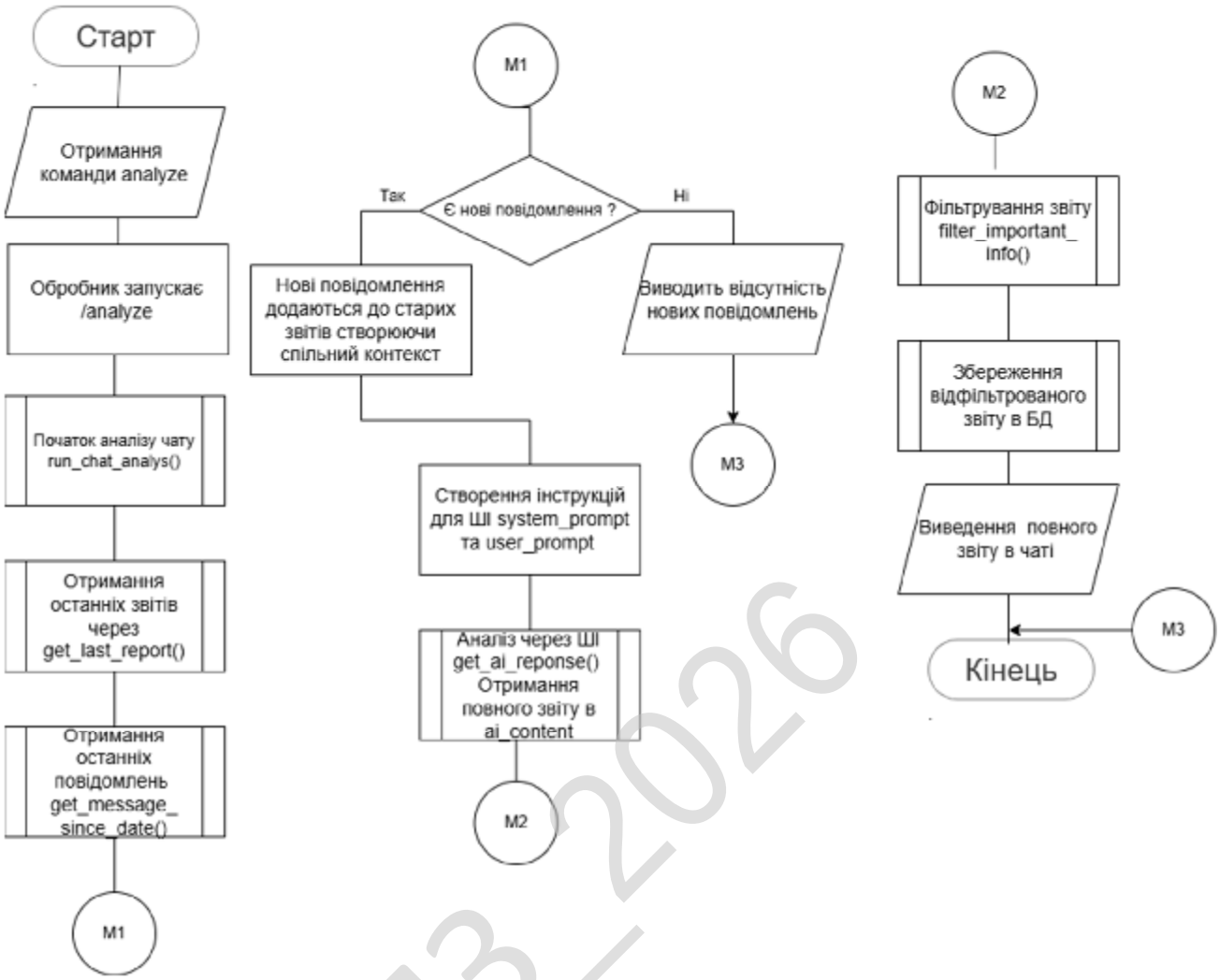


Рисунок 4.3 – Блок-схема підпрограми

На рисунку 4.4 зображено блок схему алгоритму створення PDF-файлу звіту для завантаження користувачем у форматі документа. Ця дія вимагає комплексного підходу із застосуванням декількох функцій аналізу даних, що робить структуру досить гнучкою, для подальшої модифікації: наприклад, для заміни одних функцій іншими, не змінюючи загальної логіки звіту.

1) Користувач вводить команду «/report» у текстовому полі чату. Telegram API автоматично передає команду боту через механізм диспетчера подій. Обробник подій викликає функцію full_report() з аргументом chat_id, яка у свою чергу запускає generate_full_report(), яка ініціює цілий ланцюг вкладених функцій з обробкою даних, результати яких згодом мають скластися у єдиний структурований звіт.

КПЗ_2026

2) Отримує повідомлення за вказаний період через get_messages_since_date(), використовуючи SQL-запит до таблиці messages. Потім викликає calculate_stats() для розрахунку базової статистики: загальної кількості повідомлень, активних днів, найактивнішої години, середньої активності на день, тощо. Далі викликається generate_ai_rating() для отримання текстового рейтингу активності від ШІ. Після цього викликає get_important_events() для отримання важливих подій та домовленостей з бази даних. Окремо виконує аналіз кожного окремого користувача з якого є хоча б одне повідомлення в базі даних через users_analysis_for_report(). Нарешті, викликає generate_trend_analys() для аналізу трендів у чаті через ШІ на основі зібраної статистики та прикладів повідомлень. Після завершення усіх викликів отримані дані зберігаються в єдину структуру report_data.

3) Функція full_report() перевіряє тип отриманих даних. Якщо report_data є рядком – це означає що сталася помилка. У такому разі бот повідомляє про помилку користувачеві та достроково завершує свою роботу без генерації PDF. Якщо report_data містить коректні дані у вигляді словника, викликається generate_pdf_report() для створення PDF-документа. Цей процес включає наступні етапи:

- Спочатку реєструється шрифт DejaVu, що є обов’язковим для підтримки української мови у згенерованому PDF.

- Потім формується PDF-документ за допомогою бібліотеки ReportLab, додаючи до нього усі отримані дані з report_data: загальну статистику, текстовий рейтинг активності, персональний аналіз учасників, список важливих подій та висновки про тренди зі сформованими рекомендаціями.

- Фінальним кроком будується готовий PDF-документ у вигляді байтового потоку, який повертається у функцію full_report().

4) Готовий PDF-файл надсилається користувачеві через вбудований у фреймворк Aiogram метод message.answer_document() безпосередньо у чат Telegram, після чого алгоритм повністю завершує свою роботу до наступного виклику команди.

КПІЗ_2026

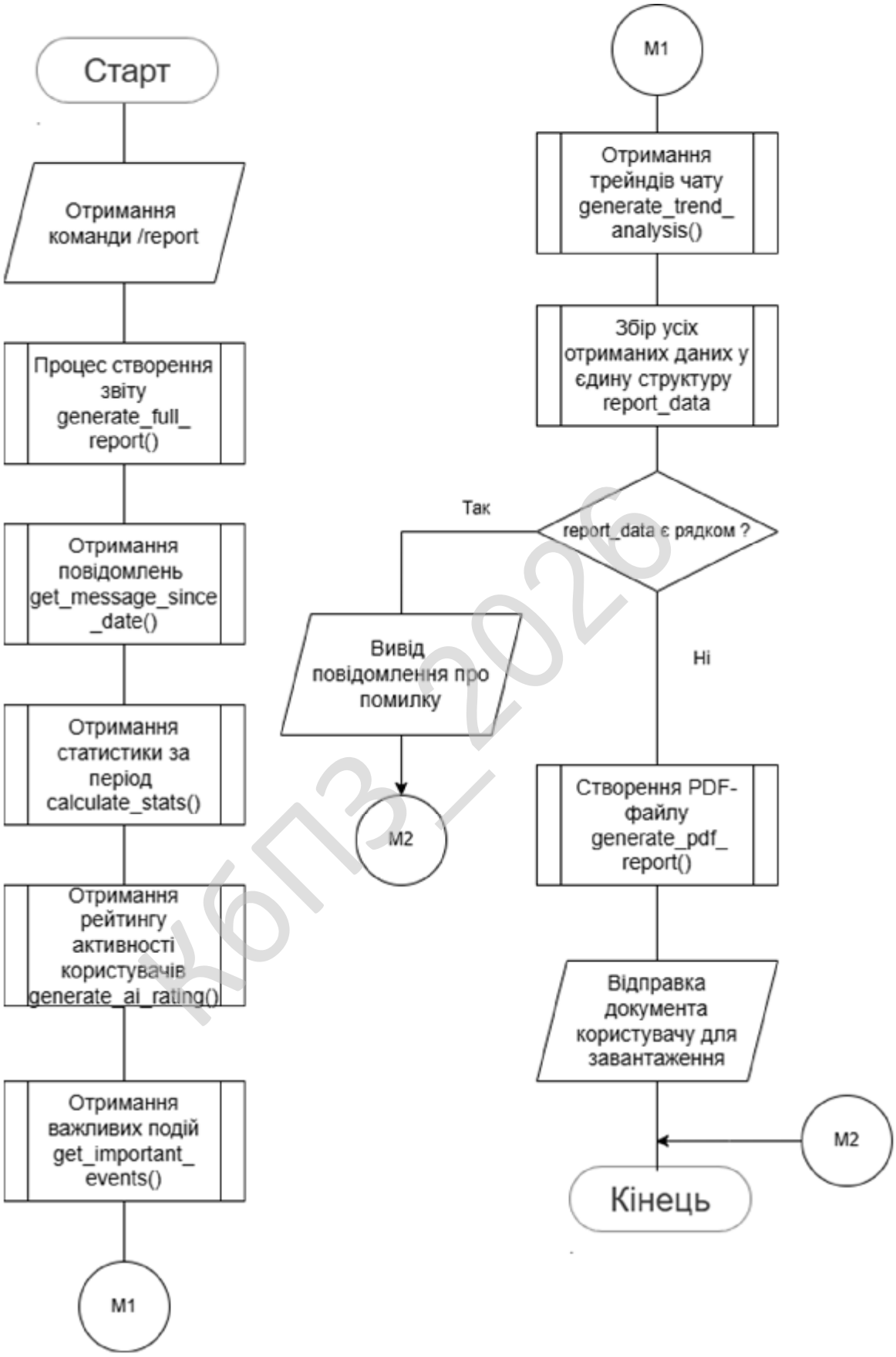


Рисунок 4.4 – Блок схема алгоритму створення PDF-звіту

КбПЗ_2026

Алгоритм команди /report можна вважати одним з найскладніших у системі через необхідність збирати та узгоджувати різнорідні дані. Він інтегрує одразу декілька аналітичних модулів, об’єднує їх результати та формує єдиний PDF- документ українською мовою зі складеною структурою.

На рисунку 4.5 зображено алгоритм створення автоматичних щоденних звітів у конкретній заздалегідь визначеній годині без участі користувачів або адміністратора. Це дозволяє системі самостійно робити звіти, які можуть бути використані надалі для аналізу, навіть якщо за цілий день жодного разу не було створено створено звіту.

1) Під час запуску бота функція setup_scheduler() створює екземпляр асинхронного планувальника завдань AsyncIOScheduler з бібліотеки APScheduler. До цього планувальника додається періодичне завдання: щодня о 21:00 виконувати функцію generate_daily_report(). Після налаштування планувальник запускається методом scheduler.start() і переходить у фоновий режим очікування.

2) Коли настає 21:00 планувальник автоматично запускає generate_daily_report(). Для чату викликається функція run_chat_analys(), яка глибоко аналізує нові повідомлення за день з урахуванням попередніх звітів. Сформований готовий звіт у вигляді текстового повідомлення надсилається безпосередньо у чат користувачам. Після успішного виконання аналізу та надсилання звіту, функція update_last_run_date() оновлює дату останнього запуску у файлі last_run.json для запобігання повторному аналізу тих самих даних.

Загалом алгоритм забезпечує практично повну автономність системи без необхідності ручного втручання. Бот під час ініціалізації самостійно перевіряє час у фоновому режимі при цьому не заважаючи роботі основної програми, а після настання години, визначеної для створення автоматичного звіту самостійно запускає повний ланцюг створення звіту включаючи застосування засобів штучного інтелекту і не потребуючи втручання адміністратора чи іншого користувача.

КПІЗ_2026

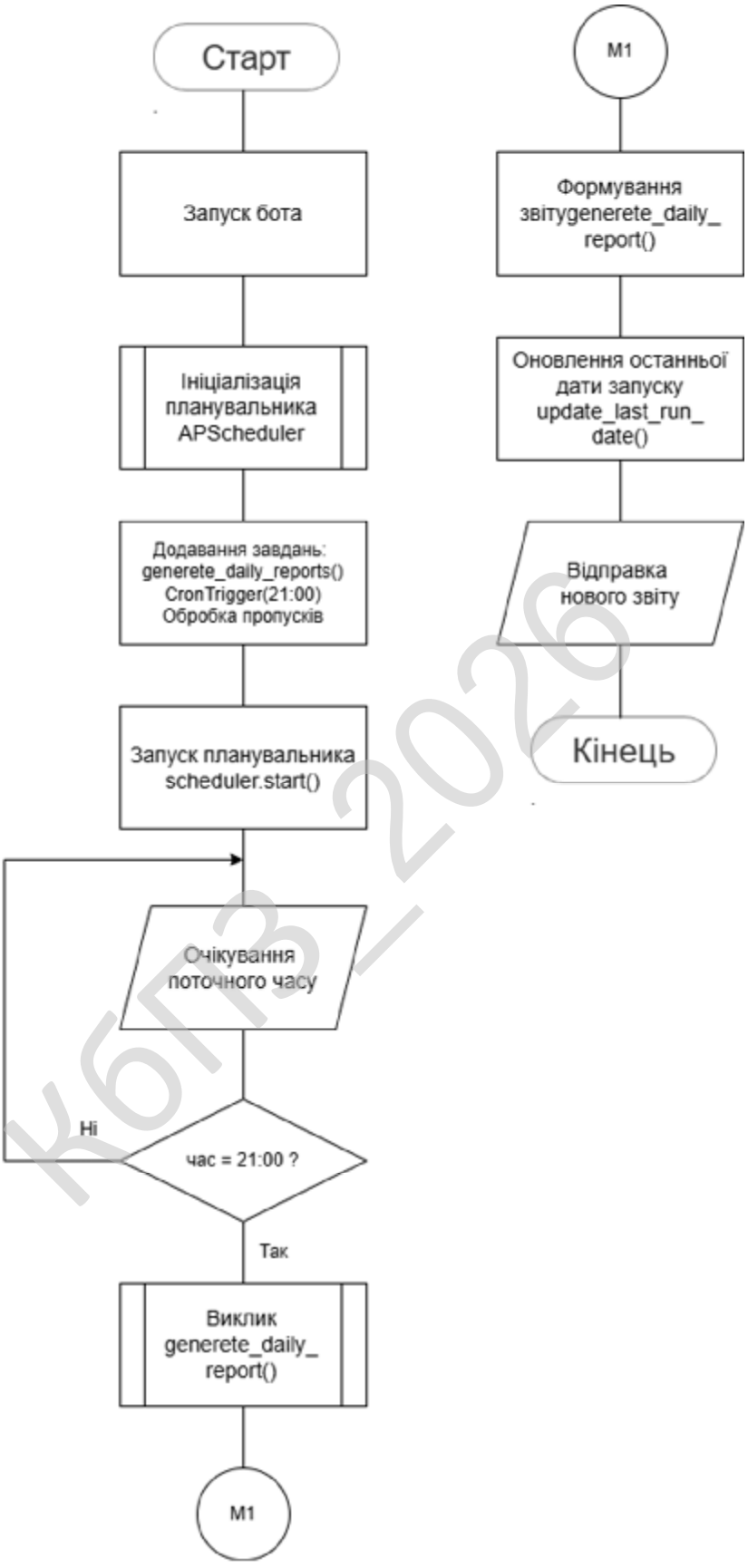


Рисунок 4.5 – Блок-схема автоматичного створення звітів

КБПЗ_2026

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення було обрано ліцензію Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0). Такий вибір зумовлений наступними причинами.

Відкрите поширення з контрольованими обмеженнями. Ліцензія надає авторам гнучкий набір умов (атрибуція, некомерційність, обов'язкове поширення похідних робіт на тих самих умовах). Це дозволяє вільно поширювати програмний код, але водночас захищає ключові права розробника.

Міжнародна юрисдикція. Ліцензії Creative Commons діють у понад 150 країнах, включаючи Україну. Це спрощує правове регулювання використання програмного забезпечення у міжнародному середовищі.

Розпізнаваність у навчальному та некомерційному середовищі. Більшість академічних та волонтерських проєктів орієнтуються на ліцензії Creative Commons. Це робить обрану модель природною для дипломної роботи.

Умови обраної ліцензії (CC BY-NC-SA 4.0)

BY (Attribution – Зазначення авторства). Кожен, хто використовує програмний код або матеріали, зобов'язаний згадати автора. У файлі документації або в описі бота необхідно вказати ім'я розробника, назву навчального закладу та посилання на репозиторій з вихідним кодом.

NC (Non-Commercial – Некомерційне використання). Забороняється пряме комерційне використання програмного забезпечення без окремої угоди з автором. Це означає, що заборонено продавати копії бота, хостити його на платних сервісах з метою отримання прибутку або включати до складу комерційних продуктів.

SA (Share-Alike – Поширення на тих самих умовах). Будь-які похідні роботи (модифікації, розширення, форки) повинні поширюватися виключно

КПІЗ_2026

на умовах тієї самої ліцензії. Це захищає від «закриття» покращеної версії бота третіми особами.

Захист конфігураційних даних та токенів

Програмне забезпечення використовує Telegram Bot API та DeepSeek API (OpenAI-сумісний інтерфейс). Для роботи з цими API необхідні секретні ключі:

BOT_TOKEN – токен Telegram-бота, отриманий від @BotFather.

API_KEY – ключ доступу до DeepSeek API.

Ці дані не можуть поширюватися разом із кодом. Тому в репозиторії передбачено наступні механізми захисту:

Окремий файл конфігурації. Усі секрети винесено у файл config.py, який не включено до системи контролю версій (додано у .gitignore).

Файл-шаблон. Для зручності розгортання створено файл config.example.py, який містить структуру конфігурації без реальних значень. Користувач, який хоче запустити бота, має скопіювати цей файл та заповнити власними токенами.

Локальне зберігання. Файл last_run.json, що містить дату останнього автоматичного звіту, не містить секретної інформації та може зберігатися разом із кодом.

Практичні кроки впровадження

Додано файл LICENSE у корінь репозиторію з повним текстом ліцензії CC BY-NC-SA 4.0 (англійською та українською мовами).

У заголовках основних файлів коду вказано дисклеймер:

python

© 2026, Кучер Д.М., ЦНТУ.

Licensed under CC BY-NC-SA 4.0

До файлу .gitignore додано config.py та інші файли, що містять секретні ключі.

Створено файл config.example.py із шаблоном конфігурації.

КбПЗ_2026

Описано умови ліцензії та порядок розгортання у документації (README.md).

Зазначено контактну адресу автора для отримання комерційної ліцензії (у разі потреби).

Переваги обраного підходу

Прозорість для користувачів. Користувачі та розробники одразу розуміють свої права й обов'язки щодо використання програмного коду.

Спрощений обмін знаннями. Викладачі та студенти можуть безперешкодно використовувати, модифікувати та покращувати програмне забезпечення в межах навчального процесу.

Легітимний захист авторських прав. Застосування ліцензії Creative Commons забезпечує мінімальні витрати на юридичний супровід – достатньо лише згадки ліцензії та правильної атрибуції.

Захист секретної інформації. Винесення конфігураційних даних за межі репозиторію запобігає витоку токенів та несанкціонованому доступу до API.

Таким чином, застосування ліцензії CC BY-NC-SA 4.0 разом із правильним налаштуванням системи контролю версій та ізоляцією конфігураційних файлів, відкриває шлях до співпраці та сприяє сталому розвитку проєкту в академічному середовищі.

КПІЗ_2026

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Система реалізована у вигляді телеграм-бота, що взаємодіє з користувачами через зручний інтерфейс чату, кнопки та команди. Для роботи з ботом достатньо наявності Telegram на мобільному пристрої або комп'ютері.

Перед початком використання системи необхідно додати бота до цільового чату (групи або супергрупи). Для цього виконуються такі дії:

1. У пошуку Telegram вводиться ім'я бота (наприклад, @ChatAnalyticsBot).

2. Відкривається профіль бота, натискається кнопка «Додати в групу».

3. Обирається необхідний чат зі списку доступних груп.

ВАЖЛИВО: Для коректної роботи всіх функцій бота (зокрема, зчитування повідомлень користувачів) йому необхідно надати права адміністратора.

Після успішного додавання бота та призначення прав адміністратора система починає автоматично збирати повідомлення чату та зберігати їх для подальшого аналізу. Без прав адміністратора бот не матиме доступу до історії повідомлень, що робить неможливим виконання команд аналізу.

На рисунку 5.1 показано початок роботи з ботом – вікно чату після вписання «/» яке відкриє користувачу наявні команди.

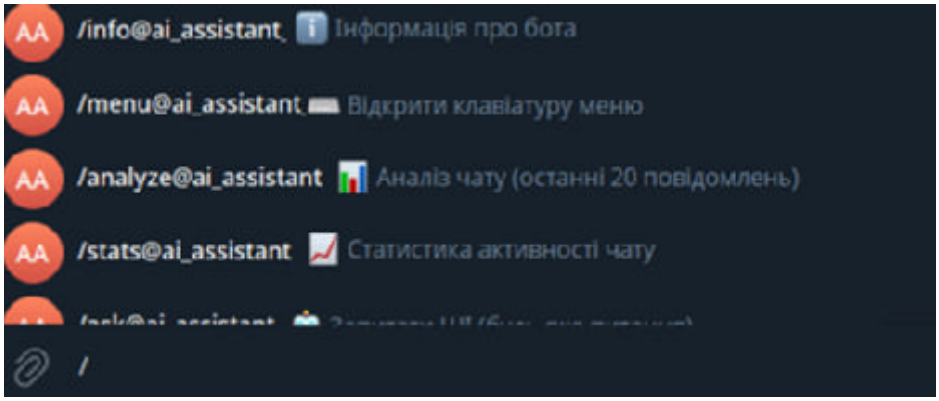


Рисунок 5.1 – Початок роботи з ботом (команда «/»)

КПІЗ_2026

Уведення команди «/help» викличе повідомлення з допомогою команд та їх пояснення, що можна побачити на на рисунку 5.2.

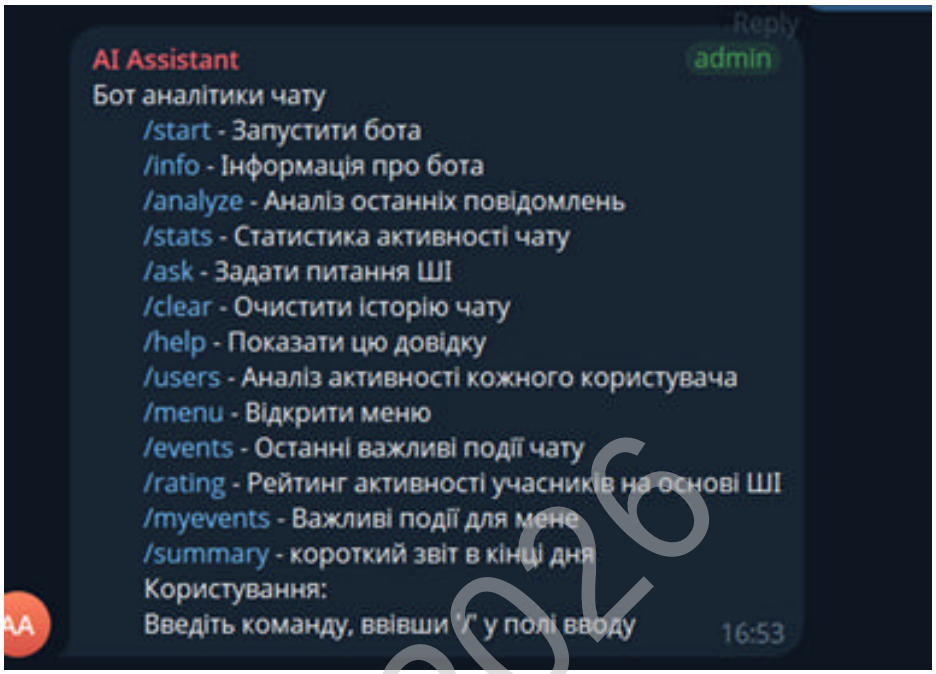


Рисунок 5.2 – Допомога користувачу з командами після команди «/help»

На рисунках 5.3 та 5.4 показано процес виконання аналізу чату. Користувач натискає кнопку вводить команду «/analyze», після чого:

1. Бот повідомляє: «Аналізую чат... Це може зайняти кілька секунд». Показ цього повідомлення означає, що бот коректно роспізнав команду і запустив функцію обробки. Користувач знає, що система працює, і не потребує повторного запит учи інших дій.

2. Система збирає останні повідомлення з історії чату. Збираються повідомлення з бази даних від часу створення останнього звіту та самі звіти в якості контексту

3. Дані надсилаються до API нейромережі для аналізу. Зовнішній сервіс приймає дані на обробку. Зазвичай отримання результату займає 5-15 секунд.

КПЗ_2026

4. Бот формує структурований звіт, який містить: оцінку активності чату, виявлені емоційні стани (стрес, роздратування, конфлікти), перелік домовленостей, планів, зустрічей, рекомендації що до покращення.

.

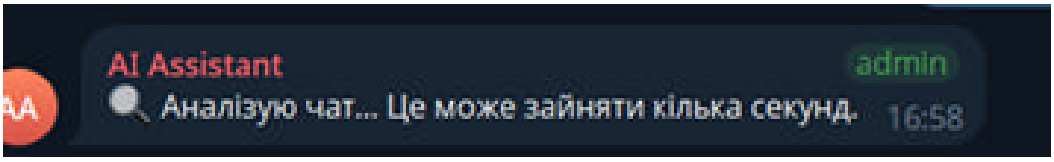


Рисунок 5.3 – Запуск аналізу чату

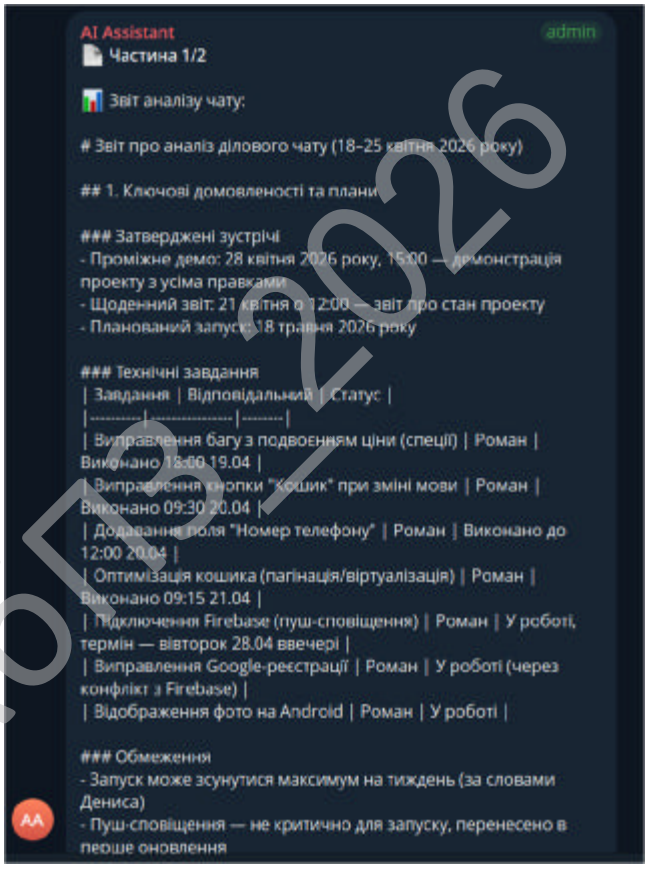


Рисунок 5.4 – Результат аналізу ШІ на тестових даних

На рисунку 5.5 показано приклад отриманого звіту. Звіт складається з таких блоків:

- Загальний аналіз – оцінка тону розмови, активності, емоційного фону.

- Домовленості та плани – структуровані задачі з виконавцями та датами.

- Поради – рекомендації щодо врегулювання конфліктів або стимулювання активності.

КБГБ-2026

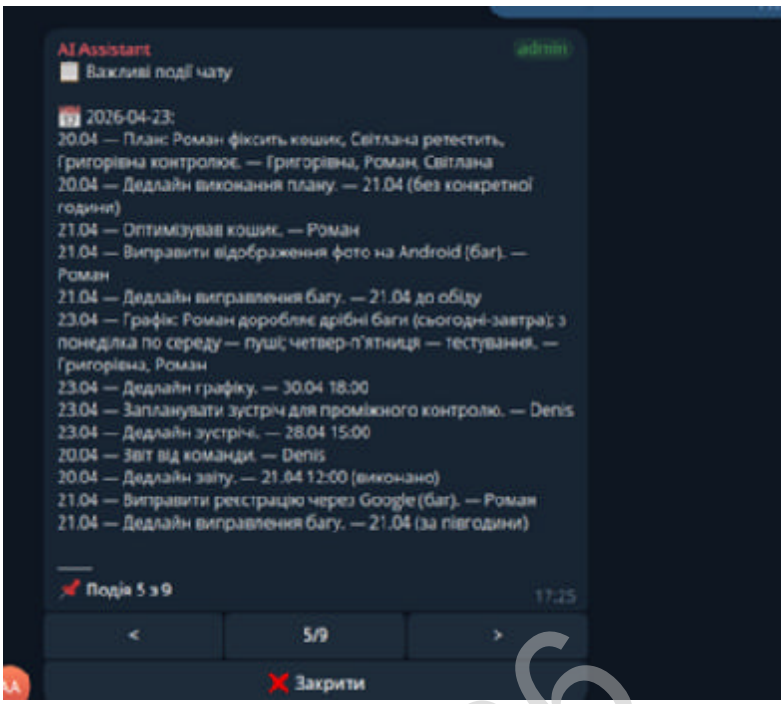


Рисунок 5.5 – Перелік звітів тестових даних

На рисунку 5.6 зображено файл, який користувач отримує в разі виконання команди «/report», або в разі натискання аналогічної кнопки клавіатури.

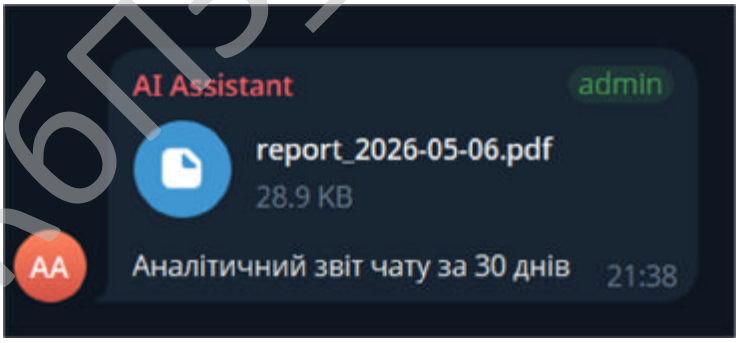


Рисунок 5.6 – PDF-файл звіт

На рисунку 5.7 зображено вміст такого файлу. Він містить тематичні розділи, щоб користувачу було легше орієнтуватися в даних, а сам звіт написаний українською мовою завдяки використанню файлу шрифтів DejaVu.

КБПЗ_2026

Аналітичний звіт чату

Період: з 2026-04-06 по 2026-05-06

Загальна статистика

Всього повідомлень: 20
Учасників: 2
Активних днів: 2
Середня активність: 10.0 повідомлень/день
Найактивніша година: 18:00

Рейтинг активності

Ось ваш дайджест активності чату: 😊 **Топ-3 активісти:** 1. **Denis** та **Григорівна** — розділили першість, ініціюючи та підтримуючи всю бесіду (по 50% повідомлень). **Динаміка:** Активність повністю рівномірна, але тримається лише на двох учасниках — решта чату мовчить. **Інсайти:** Лідери створили баланс, але без підключення інших діалог ризикує стати "розмовою вдвох". "Мовчуни" потребують залучення. **Порада:** Запропонуйте конкретне питання чи тему для голосування, щоб витягнути пасивних учасників у дискусію.

Аналіз учасників

Denis: [Denis]: Тон: Позитивний, Стиль: Професійний. Теми: організація, розклад, планування, контроль. Висновок: Відповідальний координатор, схильний до чіткої структури та делегування завдань. Григорівна: [Григорівна]: Нейтральний, Діловий. Теми: організація, уточнення деталей, форма звітності. Висновок: Відповідальна, уважна до процедур, проактивна в комунікації.

Важливі події

2026-05-05: [05.05.2026] — Списки групи сформовано, журнал буде надіслано ввечері — Григорівна [18.05.2026] — Регулярні заняття (вівторок та четвер на 1 год) — Без відповідального [20.05.2026 о 18:00] — Пробне заняття (тема: знайомство та тестування) — Без відповідального [05.05.2026] — Надіслати повний журнал студентів — Григорівна [05.05.2026] — Скинути приклад структури журналу оцінок — Denis [08.05.2026] — Додати всіх учасників до спільної групи — Denis [06.05.2026] — Підготувати простий тест для пробного заняття — Григорівна [До 20.05.2026] — Створити акаунт Google Meet та підготувати посилання — Григорівна

Тренди та рекомендації

```json { "trends": [ "Організація навчального процесу (розклад, групи, матеріали)", "Використання цифрових інструментів (Google Meet, електронна пошта, звітність у таблицях)", "Розподіл відповідальності між викладачами та

КБПІ 2026

Рисунок 5.7 – Вміст згенерованого PDF-файлу
```

6 ОСНОВНІ ВИСНОВКИ

Проблема інформаційного перевантаження ділових чатів, піднята у вступі, вирішується у роботі за допомогою створеного програмного забезпечення. У першому розділі було визначено призначення системи та область її використання – у невеликих або середніх ділових та робочих чатах. Чати є основним робочим інструментом багатьох команд, однак важливі повідомлення губляться серед особистих обговорень і буденних тем. Розроблений чат-бот виконує фільтрацію надлишкових даних, залишаючи інформацію, яка дійсно має значення для роботи.

У другому розділі проведено дослідження та аналіз існуючих аналогів, який показав, які підходи до інтеграції ШІ та систем аналітики є ефективними, а які мають суттєві недоліки. На основі цього аналізу обґрунтовано власний вибір програмних засобів для реалізації – ним стала мова програмування Python з спеціалізованими бібліотеками: Aiogram, OpenAI, Regular expression, а також SQLite база даних для збереження даних. Бот функціонує в месенджері Telegram і не висуває особливих вимог до обладнання – достатньо будь-якого пристрою з підтримкою цього месенджера. Мова програмування Python забезпечує гнучкість коду, а також зручність його супроводу та подальшого розширення.

В третьому розділі було описано структурні й функціональні схеми а також діаграми процесів для кращого розуміння поведінки системи, завдяки чому вже у четвертому розділі було реалізовано та наведено блок-схеми основних алгоритмів роботи системи. Розроблений застосунок формує щоденні звіти та PDF-файли. Додатково він будує рейтинги активності учасників та виконує аналіз кожного користувача окремо. Важливою функцією є оцінка психологічно-емоційного стану учасників – про це зазначалося на початку роботи. Керівник команди отримує можливість відстежувати загальну атмосферу, своєчасно виявляти конфлікти або ознаки

КОПЗ_2026

професійного вигорання. Без застосування штучного інтелекту такий рівень деталізації був би неможливим, що й становить головну цінність проєкту. Із практичного погляду ключовим є заощадження часу: працівникові не потрібно перечитувати тисячі повідомлень, щоб увійти в контекст – звіт дає змогу зробити це значно швидше. Автоматичне виокремлення завдань, термінів, виконавців значно скорочує робочі витрати. Приклади роботи системи та методи її впровадження наведено у п’ятому розділі, що також підтверджує її готовність до реальної експлуатації. Таким чином, розроблене ПЗ є інструментом, придатним до реального використання, який значно знижує вплив інформаційного шуму.

КБПЗ_2026

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Aiogram Documentation: Modern and fully asynchronous framework for Telegram Bot API [Електронний ресурс]. – Режим доступу: https://docs.aiogram.dev/ (дата звернення: 12.05.2026).

2. Finite State Machine // Aiogram Documentation [Електронний ресурс]. – Режим доступу: https://docs.aiogram.dev/en/latest/dispatcher/finite_state_machine/index.html (дата звернення: 12.05.2026).

3. APScheduler – Advanced Python Scheduler // APScheduler Documentation [Електронний ресурс]. – Режим доступу: https://apscheduler.readthedocs.io (дата звернення: 11.05.2026).

4. asyncio - Asynchronous I/O // Python Standard Library [Електронний ресурс]. – Режим доступу: https://docs.python.org/3/library/asyncio.html (дата звернення: 16.05.2026).

5. DeepSeek API Documentation // DeepSeek [Електронний ресурс]. – Режим доступу: https://api-docs.deepseek.com (дата звернення: 12.05.2026).

6. JSON Data Interchange Syntax [Електронний ресурс] // ECMA International. – Standart ECMA-404. – Режим доступу: https://www.ecma-international.org/publications-and-standards/standards/ecma-404/ (дата звернення: 12.05.2026).

7. LangChain DeepSeek Integration [Електронний ресурс]. – Режим доступу: https://pypi.org/project/langchain-deepseek-v4/ (дата звернення: 12.05.2026).

8. LM Studio // LM Studio [Електронний ресурс]. – Режим доступу: https://lmstudio.ai/docs/app/basics (дата звернення: 12.05.2026).

9. pyTelegramBotAPI Documentation [Електронний ресурс]. – Режим доступу: https://pypi.org/project/pyTelegramBotAPI/ (дата звернення: 12.05.2026).

КПІЗ_2026

10. re - Regular expression operations // Python Standard Library [Електронний ресурс]. – Режим доступу: https://docs.python.org/3/library/re.html (дата звернення: 12.05.2026).

11. ReportLab – PDF Generation Toolkit // ReportLab Documentation [Електронний ресурс]. – Режим доступу: https://www.reportlab.com/docs/reportlab-userguide.pdf (дата звернення: 11.05.2026).

12. SQLite Documentation // SQLite [Електронний ресурс]. – Режим доступу: https://www.sqlite.org/docs.html (дата звернення: 10.05.2026).

13. SQLite FTS5 Extension Documentation // SQLite [Електронний ресурс]. – Режим доступу: https://www.sqlite.org/fts5.html (дата звернення: 12.05.2026).

14. Telegram Bot API // Telegram Documentation [Електронний ресурс]. – Режим доступу: https://core.telegram.org/bots/api (дата звернення: 10.05.2026).

15. Visual Studio Code: Code Editing. Redefined [Електронний ресурс] // Microsoft. – Режим доступу: https://code.visualstudio.com/docs (дата звернення: 12.05.2026).

16. LLM Token Calculator [Електронний ресурс]. – Режим доступу: https://www.token-calculator.com/ (дата звернення: 12.05.2026).

17. Beazley D., Jones B. Python Cookbook. – 3rd ed. – O'Reilly Media, 2013. – 706 p.

18. Bird S., Klein E., Loper E. Natural Language Processing with Python. – Sebastopol : O'Reilly Media, 2009. – 506 p.

19. A Survey on Conversational Agents/Chatbots: Classification and Design Techniques / Nuruzzaman M., Khaled A. // International Journal of Computer Applications. – 2024. – Vol. 186, № 8. – C. 1–12.

20. Chatbot Market Size, Share & Trends Analysis Report By Offering (Solution, Services), By Type, By Medium, By Business Function, By Application, By Vertical By Region, And Segment Forecasts, 2025–2030. – URL: https://www.grandviewresearch.com/industry-analysis/chatbot-market (дата звернення: 10.05.2026).

КОПЗ_2026

21. Chowdhury S. K. Token Optimization: The Backbone of Effective Prompt Engineering // IBM Developer [Електронний ресурс]. – 2024. – Режим доступу: https://developer.ibm.com/articles/awb-token-optimization-backbone-of-effective-prompt-engineering (дата звернення: 12.05.2026).

22. Kuzminykh N. Calculating LLM Token Counts: A Practical Guide / Natalia Kuzminykh // Winder.ai Blog [Електронний ресурс]. – 2024. – Режим доступу: https://winder.ai/calculating-token-counts-llm-context-windows-practical-guide/ (дата звернення: 12.05.2026).

23. DeepSeek V4 Released: What Is and How to Access // CometAPI [Електронний ресурс]. – 2026. – Режим доступу: https://www.cometapi.com/deepseek-v4-released-what-is-and-how-to-access (дата звернення: 12.05.2026).

24. Dhanuka D. Impact of LLMs on Team Collaboration in Software Development // arXiv preprint. – 2025. – arXiv:2510.08612v1 [cs.SE].

25. Emotion Concepts and Their Function in a Large Language Model // Anthropic [Електронний ресурс]. – 2026. – Режим доступу: https://www.anthropic.com/research/emotion-concepts-function (дата звернення: 12.05.2026).

26. Ahmed A. Semantic Density Effect (SDE): Maximizing Information Per Token Improves LLM Accuracy / Amr Ahmed // arXiv preprint. – 2026. – arXiv:2604.17659 [cs.CL]. – Режим доступу: https://arxiv.org/abs/2604.17659 (дата звернення: 12.05.2026).

27. Gamma E. та ін. Design Patterns: Elements of Reusable Object-Oriented Software. – Boston : Addison-Wesley, 1995. – 395 p.

28. Harisudhan S. LLMs vs AI Agents: What Is the Actual Difference // Medium [Електронний ресурс]. – 2025. – Режим доступу: https://medium.com/@speaktoharisudhan/llms-vs-ai-agents-what-is-the-actual-difference-cebd4cb789cd (дата звернення: 12.05.2026).

КПІЗ_2026

29. Artificial Intelligence (AI) in Business // IBM Think [Електронний ресурс]. – 2026. – Режим доступу: https://www.ibm.com/think/topics/artificial-intelligence-business (дата звернення: 12.05.2026).

30. Jiang X., Hu T., Qin Y. та ін. ChatMap: Mining Human Thought Processes for Customer Service Chatbots via Multi-Agent Collaboration // Findings of the Association for Computational Linguistics: ACL 2025. – 2025. – С. 11927–11947.

31. Jurafsky D., Martin J. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. – 4th ed. – Pearson, 2023. – 1 280 p.

32. Kapoor A., Verma V. Emotion AI: Understanding Emotions Through Artificial Intelligence // International Journal of Engineering Science and Humanities (IJESH). – 2024. – Special Issue, Vol. 1, № 1. – P. 224–235.

33. Kryvun I. LLM Tokens and Foreign Languages // Ikriv Blog [Електронний ресурс]. – 2025. – Режим доступу: https://ikriv.com/blog/?p=5322 (дата звернення: 12.05.2026).

34. Li Y., Sun Q., Schlicher M. та ін. Artificial Emotion: A Survey of Theories and Debates on Realising Emotion in Artificial Intelligence // arXiv preprint. – 2025. – arXiv:2508.10286v2 [cs.HC]. – 56 p.

35. Liu Z., Yang K., Xie Q. та ін. EmoLLMs: A Series of Emotional Large Language Models and Annotation Tools for Comprehensive Affective Analysis // arXiv preprint. – 2024. – arXiv:2401.08508v2 [cs.CL].

36. LLM Model Evolution 2024–2026: The Complete Reference (244 Models) // LLM-Evolution.com [Електронний ресурс]. – 2026. – Режим доступу: https://www.llm-evolution.com (дата звернення: 12.05.2026).

37. Maksymenko M., Turuta O., Koval O. та ін. Tokenization Efficiency of Current Foundational Large Language Models for the Ukrainian Language // Frontiers in Artificial Intelligence. – 2025. – Vol. 8. – DOI: 10.3389/frai.2025.1538165.

КБГЗ_2026

38. Python in Sentiment Analysis: A Review with a Focus on Social Media Text // Applied and Computational Engineering. – 2025. – Vol. 191, № 1. – P. 46–51. – Mode of access: https://www.researchgate.net/publication/396583519 (дата звернення: 16.05.2026).

39. Fowler M. Refactoring: Improving the Design of Existing Code. – 2nd ed. – Addison-Wesley, 2019. – 448 p.

40. Sentiment Analysis and Opinion Mining / Bing Liu // Synthesis Lectures on Human Language Technologies. – Morgan & Claypool Publishers, 2020. – 167 p.

41. Sentiment Analysis Using Python [Електронний ресурс] // Analytics Vidhya. – Режим доступу: https://www.analyticsvidhya.com/blog/2022/07/sentiment-analysis-using-python/ (дата звернення: 12.05.2026).

42. Shen G., Zhao D., Bao A. та ін. StressPrompt: Does Stress Impact Large Language Models and Human Performance Similarly? // arXiv preprint. – 2025. – arXiv:2409.17167v2 [cs.HC].

43. Sweigart A. Automate the Boring Stuff with Python. – 2nd ed. – San Francisco : No Starch Press, 2019. – 592 p.

44. The impact of chatbots based on large language models on second language vocabulary acquisition // Heliyon. – 2024. – Vol. 10, № 4. – Article e14014. URL: https://www.sciencedirect.com/science/article/pii/S2405844024014014 (дата звернення: 16.05.2026).

45. Voitovych O., Ostrovska V., Zakalov I. Negative Influences Exposure in Social Internet-services [Електронний ресурс] // Digital Platform: Information Technologies in Sociocultural Sphere. – 2018. – № 2. – С. 93–105. – Режим доступу: https://doi.org/10.31866/2617-796x.2.2018.155667 (дата звернення: 12.05.2026).

46. What Are Large Language Models (LLMs)? // Microsoft Azure [Електронний ресурс]. – Режим доступу: https://azure.microsoft.com/en-

КПІЗ_2026

us/resources/cloud-computing-dictionary/what-are-large-language-models-llms (дата звернення: 12.05.2026).

47. What Is Prompt Engineering? // IBM Think [Електронний ресурс]. – 2025. – Режим доступу: https://www.ibm.com/think/topics/prompt-engineering (дата звернення: 12.05.2026).

48. Python [Електронний ресурс] // Вікіпедія : вільна енциклопедія. – Режим доступу: https://uk.wikipedia.org/w/index.php?title=Python&oldid=44658602 (дата звернення: 12.05.2026).

49. Subbotin S. O. Нейронні мережі: теорія та практика : навч. посіб. – Житомир : Вид. О.О. Євенок, 2020. – 184 с.

50. Штучна нейронна мережа // Вікіпедія [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Штучна_нейронна_мережа (дата звернення: 12.05.2026).

51. Що таке великі мовні моделі (LLM)? // Amazon Web Services (AWS) [Електронний ресурс]. – Режим доступу: https://aws.amazon.com/what-is/large-language-model (дата звернення: 12.05.2026).

КбПЗ_2026