Вищий навчальний заклад

«Університет економіки та права «КРОК»
Фаховий коледж

Циклова комісія з інформаційних технологій

Кваліфікаційна робота фахового молодшого бакалавра

на тему Месенджер-бот «Нагадувач про важливі дедлайни»

Виконав_____
(Підпис)
_____Катана Євген Олексійович_____
(прізвище, ім'я, по батькові)

_____

Науковий керівник_____
_____Кириченко Віктор Вікторович_____
(прізвище, ім'я, по батькові)

_____

_____
(Резолюція «До захисту»)

Попередній захист:

_____
(Висновок: "До захисту в екзаменаційній комісії ")

Голова циклової комісії _____ _____ _____
(Підпис) (Прізвище, ініціали) (Дата)

Київ – 2025 року

ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД

УНІВЕРСИТЕТ ЕКОНОМІКИ ТА ПРАВА «КРОК»

Фаховий коледж

Циклова комісія з інформаційних технологій

Спеціальність 121 інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:

Голова циклової комісії _____ Леонід УВАРОВ

(підпис)

«____» _____ 2025 року

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Здобувач освіти Катана Євген Олексійович

1. Тема роботи Месенджер-бот «Нагадувач про важливі дедлайни» затверджена наказом по університету

від «___» _____ 202__ р. № _____

2. Термін здачі закінченої роботи «30» травня 2025 року

3. Вихідні дані до роботи:

1) **Цільова аудиторія** – студенти, фахівці, працівники офісів, а також будь-які користувачі, які мають потребу в систематичних нагадуваннях про важливі дедлайни (навчальні, робочі, побутові).
2) **Функціональність** – створення, редагування та видалення нагадувань; налаштування дати, часу, періодичності; отримання сповіщень у месенджері; ведення історії подій; групування нагадувань.
3) **Технічні вимоги** – використання платформи Telegram для реалізації бота; мова програмування Python; зберігання даних у реляційній базі даних (SQLite); використання хостингу або хмарного сервісу для розгортання.
4) **Можливість інтеграції** – опціональна підтримка інтеграції з календарями (Google Calendar, Outlook) або іншими сервісами планування.
5) **Існуючі рішення та кращі практики** – огляд популярних Telegram-ботів з функціями нагадувань, вивчення їхніх переваг, недоліків, інтерфейсів, а також практик зручної взаємодії з користувачем (UX/UI).

4. Зміст пояснювальної записки:

1) **Розділ 1. Теоретична частина.**
Аналіз проблеми неорганізованості у керуванні часом; вплив своєчасних нагадувань на ефективність діяльності; огляд та аналіз існуючих месенджер-ботів для нагадувань; обґрунтування вибору платформи Telegram та інших технологій для реалізації проєкту.

2) **Розділ 2. Проєктування та розробка.**
   Проєктування архітектури системи; розробка логіки обробки запитів; реалізація зручного інтерфейсу спілкування з ботом; реалізація базового функціоналу: додавання/видалення нагадувань, обробка помилок; реалізація нагадувань у вигляді повідомлень; інтеграція з базою даних; підготовка до масштабування.
3) **Розділ 3. Експериментальна частина.**
   Функціональне тестування бота на різних пристроях; виявлення та усунення помилок; аналіз зручності використання; формування рекомендацій щодо подальшого вдосконалення; створення інструкції користувача; забезпечення збереження особистої інформації користувачів та відповідність етичним і юридичним нормам.

5. Перелік графічного матеріалу:

- Скріншоти популярних рішень;
- Скріншоти інтерфейсу власного бота;
- Таблиці порівняння функціональностей;
- Блок-схеми алгоритмів обробки запитів;
- Діаграми проєктування (структури даних, UML-діаграми, ER-діаграма бази даних тощо);

Дата видачі завдання «12» лютого 2025 року

Науковий керівник _____ _Кириченко В. В._
(підпис) (прізвище, ім'я, по батькові)

Завдання прийняла до виконання _____ __Катана Є. О._
(підпис) (прізвище, ім'я, по батькові)

РЕФЕРАТ

Пояснювальна записка: 52 сторінка, 13 рисунків, 6 таблиць, 6 додатків, 15 джерел.

Об'єкт дослідження — програмні засоби організації персонального управління часом користувача на базі месенджерів.

Мета роботи — розробка месенджер-бота «Нагадувач про дедлайни» для Telegram, що дозволяє користувачам створювати, редагувати та отримувати нагадування про важливі події та дедлайни у зручній інтерактивній формі.

У кваліфікаційній роботі подано результати розробки Telegram-бота з використанням Python та бібліотеки telebot для інтеграції з Telegram API. Проведено аналіз проблеми тайм-менеджменту та існуючих рішень у сфері автоматизованих нагадувань. Сформульовано функціональні та нефункціональні вимоги до системи. Визначено архітектуру програмного забезпечення, включаючи обробку запитів, структуру бази даних (на основі SQLite) та алгоритми генерації нагадувань. Розроблено діаграми потоків даних, структури даних, UML-діаграми та ER-модель для опису логіки бота.

Результатом роботи є функціональний Telegram-бот, який підтримує реєстрацію користувачів, створення, редагування, перегляд та видалення нагадувань, а також надсилання сповіщень у встановлений час. Бот адаптований для масштабування та подальшого розширення функціоналу.

Розроблене рішення може бути використане широким колом користувачів, зокрема студентами та викладачами вищих навчальних закладів для ефективного планування часу.

Ключові слова: месенджер-бот, Telegram, дедлайни, нагадування, база даних, SQLite, Python, інформаційна система, Telegram API, персональне планування часу.

ABSTRACT

Explanatory note: 52 pages, 13 figures, 6 tables, 6 appendices, 15 sources.

Object of study — software tools for personal time management based on messenger platforms.

Purpose of the work — to develop a Telegram messenger bot “Deadline Reminder” that allows users to create, manage, and receive interactive reminders about important deadlines and events.

This qualification thesis presents the development results of a Telegram bot using Python and the telebot library to interact with the Telegram API. The study includes analysis of time-management issues and existing automated reminder solutions. The system’s functional and non-functional requirements were defined. The architecture of the bot was developed, including request processing, database structure (based on SQLite), and reminder scheduling algorithms. Data flow diagrams (DFD), data structures, UML diagrams, and an ER model were designed to describe the system logic.

The result is a fully functional Telegram bot supporting user registration, creation, editing, viewing, and deletion of reminders, as well as timely notification delivery. The bot is scalable and can be extended with additional features.

The developed solution can be used by a wide range of users, especially students and academic staff, to support effective time management.

Keywords: messenger bot, Telegram, deadlines, reminders, database, SQLite, Python, information system, Telegram API, personal time planning.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

API — *Application Programming Interface* (програмний інтерфейс)

DB — *Database* (база даних)

UI — *User Interface* (користувацький інтерфейс)

UX — *User Experience* (користувацький досвід)

SQL — *Structured Query Language* (мова структурованих запитів)

SQLite — легка вбудована система управління базами даних

IDE — *Integrated Development Environment* (інтегроване середовище розробки)

JSON — *JavaScript Object Notation* (формат обміну даними)

HTTP — *HyperText Transfer Protocol* (протокол передавання гіпертексту)

CRUD — *Create, Read, Update, Delete* (операції створення, читання, оновлення, видалення даних)

TTS — *Text-to-Speech* (текст у мовлення)

ML — *Machine Learning* (машинне навчання)

UX/UI — *User Experience / User Interface* (користувацький досвід / користувацький інтерфейс)

Msg — *Message* (повідомлення)

ВСТУП

Актуальність завдання. В сучасних умовах інформаційного перенасичення та високих вимог до продуктивності люди часто стикаються з проблемою своєчасного виконання завдань. Пропущені дедлайни призводять до втрати можливостей, штрафів або негативного впливу на репутацію. Традиційні способи нагадування (календарі, записники) часто є незручними або недієвими. Використання месенджер-ботів для автоматизованих нагадувань — ефективне та доступне рішення, що підвищує організованість та продуктивність користувачів [1], [12].

Мета роботи. Розробити функціональний месенджер-бот «Нагадувач про важливі дедлайни», який забезпечить зручний інтерфейс для створення, редагування та отримання нагадувань про важливі події, сприятиме покращенню управління часом та своєчасному виконанню завдань.

Завдання роботи

- Провести аналіз існуючих месенджер-ботів і визначити їхні переваги та недоліки [11], [9].
- Визначити вимоги до функціоналу майбутнього бота [6], [13].
- Спроєктувати архітектуру та основні модулі системи [4], [10].
- Реалізувати бот на обраній платформі (наприклад, Telegram) [3], [15].
- Провести тестування функціональності та зручності користування [7], [6].
- Оцінити результати та підготувати рекомендації щодо подальшого розвитку.

Об’єкт дослідження. Системи автоматизованих нагадувань на базі месенджерів як інструмент підвищення ефективності управління особистим часом.

Предмет дослідження. Функціональні можливості та технічна реалізація месенджер-бота для нагадування про важливі дедлайни.

Методи дослідження:

- Аналіз літературних та інтернет-джерел [1], [2], [4], [6], [12];

- Проєктування програмних систем [13];
- Програмування з використанням API месенджерів [3], [10], [15];
- Тестування функціональності (unit-тести, інтеграційні тести) [7];
- Опитування та збір зворотного зв’язку від користувачів [11].

Практичне значення одержаних результатів. Створений месенджер-бот може бути використаний студентами, працівниками, підприємцями для ефективного контролю дедлайнів, що сприятиме підвищенню продуктивності та організованості. Розроблена архітектура та алгоритми можуть слугувати основою для створення подібних інструментів на інших платформах або для інших завдань [9], [10].

РОЗДІЛ 1. ТЕОРЕТИЧНА ЧАСТИНА

1.1 Аналіз проблеми організації часу

Організація часу — це не лише навичка, а й невід'ємна умова ефективної діяльності в сучасному світі. У всіх сферах — навчанні, роботі, побуті — відсутність належного планування призводить до зниження продуктивності, втрати мотивації, емоційного вигорання та хронічного стресу [1], [12].

Основні проблеми управління часом

У сучасних умовах спостерігається **інформаційне перевантаження**, що призводить до:

- надлишку завдань із різних сфер життя;
- труднощів у розставлянні пріоритетів;
- частого ігнорування нагадувань чи планів;
- відкладання важливих справ на останній момент (ефект прокрастинації) [1], [11].

Найпоширеніші причини неефективного управління часом:

1. **Відсутність системного підходу** – користувачі не використовують жодної моделі планування (наприклад, методи «Pomodoro», Eisenhower Matrix, GTD тощо).
2. **Недостатня візуалізація цілей і дедлайнів** – паперові блокноти або розрізнені списки справ не дозволяють бачити загальну картину навантаження.
3. **Низький рівень самодисципліни** – навіть при наявності інструментів часто бракує сили волі для їх послідовного використання.
4. **Фрагментованість цифрових рішень** – календарі, списки справ, месенджери, нагадування існують окремо, не маючи єдиного центру управління [6], [12].

Обмеження традиційних інструментів

Серед популярних, але недостатньо ефективних засобів тайм-менеджменту:

Інструмент	**Недоліки**
Паперовий планувальник	Не забезпечує автоматичних нагадувань, незручний у динамічному середовищі

Мобільні нагадування	Одноразові, погано масштабуються, важко інтегруються у повсякденну рутину
Гугл-календар або ToDo-ліcти	Складні в налаштуванні, потребують переходу в окремі додатки
Чат-групи для нагадувань	Не мають особистої адресності, швидко губляться в потоці повідомлень

У багатьох випадках навіть цифрові додатки не покривають **потреби в гнучкому плануванні**, нагадуваннях із відтермінуванням, повтореннями, залежністю завдань одне від одного, зворотнім зв’язком або аналітикою [6], [11].

Перспективи месенджер-ботів

В останні роки значно зросла популярність використання **ботів у месенджерах** як інструменту нагадування. Це пояснюється:

- доступністю API популярних платформ (Telegram, Viber, WhatsApp) [3], [10];
- звичністю середовища спілкування — нагадування приходять у вже знайомий інтерфейс;
- мінімалізмом і швидкістю — не потрібно відкривати окремі додатки;
- можливістю адаптації під конкретного користувача (індивідуальні нагадування, діалоги, сценарії).

Проте навіть ці рішення досі не є універсальними. Існуючі месенджер-боти часто мають вузький функціонал, обмежені в налаштуваннях, не інтегруються з календарями чи хмарними сервісами та рідко підтримують складні типи нагадувань [9], [11].

Висновок

Таким чином, аналіз існуючої ситуації показує, що **проблема організації часу** залишається актуальною як на індивідуальному, так і на системному рівні. Недосконаліть наявних інструментів, їх низька інтеграція та недостатня адаптивність створюють потребу у розробці нових, більш функціональних рішень. Створення месенджер-бота, який поєднує простоту використання з розширеними можливостями планування, стане значущим кроком до вирішення цієї проблеми [1], [9], [11].

1.2 Огляд існуючих месенджер-ботів для нагадувань

У сфері цифрового тайм-менеджменту месенджер-боти стали важливим інструментом, що дозволяє користувачам не лише залишатися організованими, а й ефективно розподіляти час, зменшуючи ризик пропуску важливих подій. У сучасних месенджерах активно використовуються різноманітні боти-нагадувачі, які допомагають користувачам слідкувати за дедлайнами, планувати завдання та автоматично отримувати сповіщення. Зважаючи на велику кількість подібних сервісів, доцільно провести детальний аналіз їхніх можливостей, переваг і недоліків.

Популярні боти для нагадувань

1. **Telegram Reminder Bot**

   Один із найвідоміших ботів у Telegram. Дозволяє встановлювати прості нагадування за допомогою текстових команд. Користувач може написати повідомлення на кшталт remind me to call mom at 19:00 — і бот надішле нагадування у встановлений час.

2. **ToDo Bot (Telegram)**

   Пропонує базову функціональність списків завдань, з можливістю додавати нові пункти, редагувати, позначати виконані, переглядати історію. Підтримує мітки часу, але не має гнучких налаштувань повторюваних подій або інтеграції з календарем [3], [15].

3. **WaterDo Bot (Facebook Messenger)**

Менш відомий бот, орієнтований на «ігрову» подачу завдань — за виконання завдань користувач отримує віртуальні нагороди. Проте він вимагає багато дозволів і не дозволяє налаштувати складні нагадування, що обмежує сферу використання [11].

4. **Viber Daily Tasks**

Надбудова, яка надає базові можливості створення нагадувань у вигляді «щоденних справ». Не підтримує налаштування залежностей між подіями чи складної періодичності. Інтерфейс обмежений політикою месенджера [10].

Порівняльна таблиця можливостей існуючих ботів

Назва бота	**Платформа**	**Створення нагадувань**	**Повторювані події**	**Інтеграція з календарем**	**Редагування нагадувань**	**Додаткові функції**
Reminder Bot	Telegram	✓	◈ (лише щоденно)	✗	✗	Простий інтерфейс
ToDo Bot	Telegram	✓	✓	✗	◈ (частково)	Списки справ
Water Do Bot	Facebook Messenger	✓	✗	✗	✗	Гейміфікація
Viber Daily Tasks	Viber	✓	◈	✗	✗	Щоденні нагадування

Примітка: ✓ — повна підтримка; ◈ — часткова підтримка; ✗ — відсутня підтримка.

Основні недоліки існуючих ботів

Попри популярність та широке використання, наявні рішення мають низку спільних обмежень:

- **Обмежена гнучкість**: більшість ботів не дозволяє налаштовувати залежності між нагадуваннями (наприклад, «надіслати сповіщення через 2 години після завершення попереднього завдання») [11].
- **Відсутність інтеграції з іншими сервісами**: значна частина ботів не має підтримки синхронізації з Google Calendar, корпоративними CRM-системами або email-сповіщенням [10], [13].
- **Недостатня безпека**: деякі сервіси зберігають дані у відкритому вигляді або не забезпечують шифрування повідомлень [6].
- **Неадаптивний інтерфейс**: для частини ботів потрібне знання команд, що знижує зручність використання для широкого кола користувачів [9].

Висновки

Отже, аналіз ринку месенджер-ботів свідчить про наявність значного попиту на інструменти автоматизованих нагадувань. Проте більшість із них не враховує специфічні потреби користувачів, які працюють із великою кількістю дедлайнів або потребують складної логіки планування. Це створює передумови для розробки нового, більш гнучкого та функціонального рішення, яке дозволить:

- створювати нагадування з різними типами повторюваності;
- змінювати існуючі події без видалення;
- пов’язувати нагадування між собою;
- гарантувати безпеку персональних даних користувача.

Розробка такого бота має на меті не лише підвищити зручність, а й реально покращити якість управління особистим часом у навчанні, роботі й повсякденному житті.

1.3 Вимоги до месенджер-бота «Нагадувач про дедлайни»

Для створення ефективного та зручного у використанні інструменту управління часом у вигляді месенджер-бота необхідно чітко окреслити вимоги до його функціоналу, продуктивності та архітектури. Визначення цих вимог є критично важливим етапом під час проєктування системи.

1.3.1 Функціональні вимоги

Функціональні вимоги описують основні можливості, які повинен реалізовувати бот для задоволення потреб користувача:

1. **Реєстрація та автентифікація користувача**
   - Ідентифікація користувача за унікальним ID месенджера.
   - Збереження персональних налаштувань та історії нагадувань.
2. **Створення нагадувань**
   - Введення назви, опису, дати та точного часу події.
   - Підтримка швидких кнопок.
3. **Редагування та видалення нагадувань**
   - Зміна часу, назви або опису вже створеного нагадування.
   - Видалення зайвих або виконаних нагадувань.
4. **Надсилання сповіщень**
   - Своєчасне надсилання повідомлень згідно з налаштуваннями.
   - Повідомлення можуть бути текстовими, з emoji.
5. **Підтримка повторюваних нагадувань**
   - Налаштування циклічності: щодня, щотижня, у вибрані дні.
6. **Діалоговий інтерфейс**
   - Бот повинен реагувати на команди, кнопки, контекстні фрази.
   - Використання меню, кнопок швидкого вибору та підтверджень.
7. **Інтеграція з базою даних**
   - Усі дані користувача повинні зберігатись у надійній базі SQLite
   - Передбачено резервне копіювання даних.

1.3.2 Нефункціональні вимоги

Нефункціональні вимоги визначають, якою має бути система з погляду якості, безпеки та масштабованості:

Категорія	Вимога
Надійність	Стабільна робота системи 24/7, автоматичне відновлення при збої.

Безпека	Шифрування збережених даних; відсутність доступу сторонніх сервісів.
Продуктивність	Відгук бота на запити — не більше 1 секунди.
Масштабованість	Легка адаптація під нові функції; підтримка понад 1000 користувачів.
Кросплатформеність	Підтримка Telegram, Viber, з перспективою додавання WhatsApp.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РОЗРОБКА БОТА

2.1 Вибір платформи та технологій

Вибір платформи та технологічного стеку є одним із найважливіших етапів розробки месенджер-бота «Нагадувач про дедлайни», оскільки від нього залежить не тільки зручність користування, а й стабільність, масштабованість та безпека системи.

Для реалізації проєкту було обрано платформу **Telegram**. Це рішення зумовлене кількома ключовими факторами:

- Telegram має велику та активну аудиторію користувачів по всьому світу, що гарантує широкий потенціал для використання бота.
- Відкритий та добре документований API дозволяє створювати різноманітний функціонал, інтегрувати бот з іншими сервісами і використовувати сучасні можливості платформи.
- Підтримка різних форматів повідомлень — текстових, мультимедійних, кнопок та меню — дозволяє зробити інтерфейс бота максимально зручним і інтерактивним.
- Високий рівень безпеки і конфіденційності даних користувачів, що є надзвичайно важливим у проєкті, де обробляються особисті нагадування.

Для розробки самого бота було обрано мову програмування **Python**, яка відома своєю простотою, читабельністю коду та великою кількістю бібліотек, що значно пришвидшує процес розробки. Зокрема, для роботи з Telegram API використовується бібліотека **pyTelegramBotAPI (telebot)** — вона є стабільною, популярною серед розробників та має зручний інтерфейс для створення чат-ботів [9], [15].

Щодо системи збереження даних, вибір припав на **SQLite** — легку вбудовану реляційну базу даних, що не вимагає додаткового серверного програмного забезпечення і добре підходить для проєктів із середнім навантаженням. SQLite дозволяє надійно зберігати інформацію про користувачів та їх нагадування, забезпечуючи швидкий доступ і просту інтеграцію з Python [5].

Основні переваги обраного технологічного стеку:

- •. Простота налаштування та запуску.
- •. Швидкий розвиток функціоналу завдяки великій кількості готових інструментів.
- •. Надійність і стабільність роботи навіть при значній кількості користувачів.
- •. Легка підтримка і масштабування проекту.

Розглядалися також інші месенджер-платформи, такі як Viber, Facebook Messenger та WhatsApp, проте через обмеження API, складнощі з доступом до функціоналу ботів та меншу популярність у цільовій аудиторії вони були відхилені [11], [13].

Отже, обрана комбінація Telegram, Python, бібліотеки telebot і бази даних SQLite є оптимальною для реалізації завдань, поставлених перед месенджер-ботом «Нагадувач про дедлайни». Вона забезпечує необхідний функціонал, зручність у розробці та подальшій підтримці, а також відповідність сучасним стандартам безпеки і продуктивності.

2.2 Архітектура та структура месенджер-бота

Архітектура месенджер-бота «Нагадувач про дедлайни» розроблена з урахуванням головних вимог — надійності, масштабованості та зручності подальшого розширення функціоналу. Головна мета такого архітектурного підходу — забезпечити ефективну взаємодію між користувачем і системою, а також зручне зберігання і обробку даних.

Основні компоненти архітектури

1. **Інтерфейс користувача (Telegram-бот)**

Взаємодія з користувачем відбувається через Telegram-бота, який приймає повідомлення, розпізнає команди і надсилає відповіді у вигляді тексту, кнопок або інших елементів інтерфейсу. Такий підхід робить спілкування інтуїтивним і зручним.

2. **Обробник запитів (серверна частина)**

Цей модуль відповідає за обробку команд користувача, керування сесіями та логікою нагадувань. Тут відбувається парсинг вхідних повідомлень, виконання бізнес-логіки і формування відповідей, що повертаються користувачеві.

3. **База даних**

Використовується для збереження інформації про користувачів, створені нагадування, історію взаємодій і налаштувань. У поточному проєкті для простоти і надійності застосовується реляційна база даних SQLite, яка добре підходить для невеликих і середніх проєктів, забезпечуючи швидкий доступ і збереження даних.

4. **Модуль сповіщень**

Цей компонент працює у фоновому режимі і відповідає за своєчасне відправлення нагадувань користувачам відповідно до встановлених дедлайнів. Він регулярно перевіряє базу даних на наявність активних нагадувань і запускає процес відправки повідомлень.

5. **API месенджера**

Завдяки API Telegram забезпечується зв’язок між користувачем і ботом: прийом повідомлень, надсилання відповідей та обробка інших подій.

Структура месенджер-бота

Архітектурно система поділяється на кілька основних шарів:

- **Презентаційний шар**

Відповідає за безпосередню взаємодію з користувачем через Telegram: обробка вхідних повідомлень, вивід меню, кнопок, повідомлень.

- **Логічний шар**

Реалізує основну бізнес-логіку: створення, редагування, видалення нагадувань, керування сесіями, налаштування періодичних сповіщень.

- **Шар доступу до даних**

Забезпечує роботу з базою даних — збереження і отримання інформації про користувачів, нагадування та інші налаштування.

Схематичне представлення

[Користувач]
↑↓
[Telegram API]
↑↓
[Обробник запитів (сервер)]
↑↓
[База даних] ←→ [Модуль сповіщень]

Висновок

Обрана архітектура чітко розділяє зони відповідальності між компонентами, що значно спрощує підтримку, тестування та масштабування месенджер-бота. Вона дозволяє швидко інтегрувати нові функції та гнучко адаптуватися до різноманітних вимог користувачів, що є важливим для успішного розвитку проєкту.

2.3 Реалізація функціоналу нагадувань

Функціонал нагадувань є основною складовою месенджер-бота «Нагадувач про дедлайни». Він забезпечує створення, збереження та своєчасне надсилання користувачам сповіщень про важливі події та дедлайни.

Основні функції модуля нагадувань:

1. **Створення нагадувань**

Користувач має можливість додати нове нагадування, вказавши основні параметри: назву події, опис, дату та час нагадування. Для зручності введення даних бот використовує покроковий діалог із користувачем, що дозволяє уникнути помилок і полегшує процес.

2. **Збереження нагадувань у базі даних**

Введені дані про нагадування зберігаються у реляційній базі даних (SQLite). Структура таблиць забезпечує можливість зберігати як одноразові, так і повторювані нагадування з налаштуваннями періодичності.

3. **Редагування та видалення нагадувань**

Користувач може переглядати список своїх нагадувань, вносити зміни або видаляти непотрібні записи. Це дозволяє підтримувати актуальність інформації та уникати зайвих сповіщень.

4. **Надсилання сповіщень**

Для своєчасного інформування користувача реалізовано модуль відправки повідомлень, який працює у фоновому режимі (через планувальник завдань або асинхронні процеси). Він періодично перевіряє базу даних на наявність нагадувань, термін яких наближається, і надсилає повідомлення у месенджер.

5. **Підтримка повторюваних нагадувань**

Реалізовано можливість налаштовувати нагадування, які повторюються за певним інтервалом (щоденно, щотижнево, щомісяця). Це особливо корисно для регулярних завдань або зустрічей.

6. **Обробка помилок та виключень**

Система передбачає валідацію введених даних і обробку помилок (наприклад, некоректний формат дати чи часу), що підвищує надійність роботи та зручність користування.

Використані технології

- Для реалізації взаємодії з користувачем застосовано бібліотеку python-telegram-bot, яка надає можливості для створення діалогів і обробки команд.
- Планування та виконання нагадувань здійснюється за допомогою бази даних, що дозволяє запускати завдання у заданий час з підтримкою повторень.

Приклад роботи функціоналу

1. Користувач натискає кнопку
2. Бот запитує назву події.
3. Користувач вводить дату та час нагадування у форматі «ДД.ММ.РРРР ГГ:ХХ».
4. Бот підтверджує створення нагадування та зберігає його у базі.
5. У визначений час бот надсилає повідомлення із текстом нагадування.

2.4 Інтеграція з базою даних

Для збереження та обробки даних користувачів і нагадувань у месенджер-боті «Нагадувач про дедлайни» використовується база даних **SQLite**. Ця легка вбудована реляційна СУБД добре підходить для невеликих і середніх проєктів, забезпечуючи простоту налаштування та швидкий доступ до даних без необхідності налаштовувати окремий сервер.

Вибір СУБД

SQLite було обрано через такі переваги:

- **Простота використання** — не потребує налаштування серверної частини, база зберігається у вигляді одного файлу на диску.
- **Легка інтеграція** — повністю сумісна з Python через стандартний модуль sqlite3 або ORM-бібліотеки (наприклад, SQLAlchemy).
- **Достатня продуктивність** — для бота з помірною кількістю користувачів і нагадувань SQLite забезпечує швидке виконання операцій.
- **Портативність** — база даних легко переноситься між різними системами.

Структура бази даних

Основні таблиці бази даних включають:

- **Users** — інформація про користувачів (ID, ім’я, дата реєстрації).
- **Dedlain** — записи про нагадування (ID, ID користувача, назва, опис, дата та час нагадування, періодичність, статус).

Технології та інструменти інтеграції

Для роботи з SQLite у проєкті використовується бібліотека **SQLAlchemy** як ORM, що значно спрощує роботу з базою, дозволяючи описувати структуру даних у вигляді класів Python та виконувати запити об'єктно-орієнтованим способом.

Основні операції з базою даних

- Додавання нових користувачів під час реєстрації.
- Створення, оновлення, видалення нагадувань за запитом користувача.
- Отримання активних нагадувань для генерації сповіщень.
- Збереження логів для подальшого аналізу.

Приклад коду підключення

```
DATABASE_NAME = 'bot_database.db'
```

Особливості роботи з SQLite

- SQLite працює із файлом бази, що робить її зручною для локального запуску та тестування.
- Через обмеження багатопотокової роботи необхідно використовувати параметр check_same_thread=False для коректної роботи в асинхронних середовищах.
- Для більших навантажень або масштабування у майбутньому можна перейти на серверні СУБД (PostgreSQL, MySQL).

Висновок

Використання SQLite для зберігання даних месенджер-бота є ефективним рішенням для початкової стадії розробки та середньої кількості користувачів. Простота інтеграції та експлуатації дозволяє зосередитись на реалізації функціоналу, забезпечуючи при цьому стабільність і надійність роботи.

2.5 Тестування основних модулів

Тестування є важливою складовою процесу розробки месенджер-бота «Нагадувач про дедлайни», що дозволяє забезпечити коректність роботи, стабільність та відповідність реалізованого функціоналу технічним вимогам.

Мета тестування

- Перевірити правильність роботи основних функціональних модулів: обробки команд користувача, створення та збереження нагадувань, відправки сповіщень.
- Виявити та усунути помилки і недоліки на ранніх етапах розробки.
- Підвищити якість програмного продукту та задовольнити вимоги замовника.

Види тестування

1. **Модульне тестування**

   Кожен окремий модуль (наприклад, обробник команд, база даних, модуль сповіщень) тестувався ізольовано з використанням тестових наборів даних. Для цього застосовувалися бібліотеки **unittest** та **pytest** у середовищі Python.

2. **Інтеграційне тестування**

Перевірка взаємодії між модулями — наприклад, коректність запису даних у базу та їх подальше використання при генерації нагадувань і відправці повідомлень.

3. **Функціональне тестування**

Тестування основних сценаріїв користувача — створення, редагування, видалення нагадувань, отримання сповіщень у визначений час.

4. **Тестування інтерфейсу користувача**

Перевірка коректності обробки введених команд, реакції на некоректні дані, зручності взаємодії через Telegram.

Основні інструменти тестування

- **unittest** та **pytest** — для автоматичного запуску модульних та інтеграційних тестів.
- **Mocking** — імітація зовнішніх сервісів (наприклад, Telegram API) для ізольованого тестування логіки без необхідності реального підключення.
- **Логування** — для відслідковування роботи системи та виявлення помилок.

Приклад тесту створення нагадування

```
def check_deadlines():
 """Цикл для перевірки дедлайнів та відправки нагадувань."""
 while True:
 now = datetime.datetime.now()
 try:
 with sqlite3.connect(DATABASE_NAME) as conn:
 cursor = conn.cursor()
 cursor.execute(
 'SELECT user_id, title, deadline_date, deadline_time, repetition, advance_reminder FROM deadlines WHERE completed = 0') #
Перевіряємо тільки незавершені
 deadlines = cursor.fetchall()

 for user_id, title, deadline_date, deadline_time, repetition, advance_reminder in deadlines:
 try:
 deadline_datetime = datetime.datetime.strptime(f"{deadline_date} {deadline_time}",
 "%d.%m.%Y %H:%M")
 time_diff = deadline_datetime - now

 # Завчасне нагадування
 if advance_reminder and advance_reminder != "Немає":
 reminder_time = deadline_datetime - calculate_advance_time(advance_reminder)
```

```
if datetime.timedelta(0) <= reminder_time - now < datetime.timedelta(minutes=1):
bot.send_message(user_id, f"🔔 Нагадування: Дедлайн '{title}' настане скоро!")

# Основне нагадування
if repetition == "Щогодини" and deadline_datetime <= now <= deadline_datetime + datetime.timedelta(
minutes=59):
bot.send_message(user_id, f"🔔 Нагадування: Дедлайн '{title}' зараз!")
elif repetition == "Щодня" and deadline_datetime.date() == now.date():
bot.send_message(user_id, f"🔔 Нагадування: Дедлайн '{title}' сьогодні!")
elif repetition == "У будні" and now.weekday() < 5 and deadline_datetime.date() == now.date():
bot.send_message(user_id, f"🔔 Нагадування: Дедлайн '{title}' сьогодні!")
elif repetition == "У вихідні" and now.weekday() >= 5 and deadline_datetime.date() == now.date():
bot.send_message(user_id, f"🔔 Нагадування: Дедлайн '{title}' сьогодні!")
elif repetition == "Щотижня" and deadline_datetime.weekday() == now.weekday():
bot.send_message(user_id, f"🔔 Нагадування: Дедлайн '{title}' сьогодні!")
elif repetition == "Що 2 тижні" and (now.date() - deadline_datetime.date()).days % 14 == 0:
bot.send_message(user_id, f"🔔 Нагадування: Дедлайн '{title}' сьогодні!")
elif repetition == "Щомісяця" and deadline_datetime.day == now.day:
bot.send_message(user_id, f"🔔 Нагадування: Дедлайн '{title}' сьогодні!")
elif repetition == "Що 3 місяці" and (
now.month - deadline_datetime.month) % 3 == 0 and deadline_datetime.day == now.day:
bot.send_message(user_id, f"🔔 Нагадування: Дедлайн '{title}' сьогодні!")
elif repetition == "Що 6 місяців" and (
now.month - deadline_datetime.month) % 6 == 0 and deadline_datetime.day == now.day:
bot.send_message(user_id, f"🔔 Нагадування: Дедлайн '{title}' сьогодні!")
elif repetition == "Щороку" and deadline_datetime.day == now.day and deadline_datetime.month == now.month:
bot.send_message(user_id, f"🔔 Нагадування: Дедлайн '{title}' сьогодні!")
elif repetition == "Ніколи":
if datetime.timedelta(days=1) <= time_diff < datetime.timedelta(days=1, minutes=5):
bot.send_message(user_id, f"🔔 Нагадування: Дедлайн '{title}' завтра!")
elif datetime.timedelta(hours=0.5) <= time_diff < datetime.timedelta(hours=0.5, minutes=5):
bot.send_message(user_id, f"🔔 Нагадування: Дедлайн '{title}' через 30 хвилин!")
elif datetime.timedelta(hours=1) <= time_diff < datetime.timedelta(hours=1, minutes=5):
bot.send_message(user_id, f"🔔 Нагадування: Дедлайн '{title}' через 1 годину!")
elif datetime.timedelta(minutes=5) <= time_diff < datetime.timedelta(minutes=5, seconds=30):
bot.send_message(user_id, f"🔔 Нагадування: Дедлайн '{title}' через 5 хвилин!")
except Exception as e:
logger.error(f"Помилка обробки дедлайну: {e}")
except Exception as e:
logger.error(f"Помилка при перевірці дедлайнів: {e}")

time.sleep(60)
```

Висновок

Застосування систематичного тестування дозволило виявити помилки та покращити якість месенджер-бота. Проведені перевірки підтвердили стабільну роботу основних функціональних модулів і відповідність системи поставленим вимогам.

РОЗДІЛ 3. ЕКСПЕРИМЕНТАЛЬНА ЧАСТИНА

3.1 План тестування месенджер-бота

План тестування є важливим документом, що визначає стратегію, обсяг, методи та засоби перевірки працездатності месенджер-бота «Нагадувач про дедлайни». Метою тестування є підтвердження відповідності розробленого продукту функціональним вимогам і якість його роботи в умовах, наближених до реальних.

Цілі тестування

- Перевірити коректність роботи основного функціоналу бота: створення, редагування, видалення нагадувань, отримання сповіщень.
- Забезпечити стабільну взаємодію користувача з ботом через інтерфейс Telegram.
- Виявити та усунути помилки, що можуть вплинути на роботу системи.

Об'єкти тестування

- Модуль обробки команд користувача.
- Модуль взаємодії з базою даних.
- Модуль планування та відправки нагадувань.
- Інтерфейс користувача в месенджері Telegram.

Види тестування

- **Функціональне тестування:** Перевірка відповідності функцій технічному завданню.
- **Інтеграційне тестування:** Тестування взаємодії між модулями.
- **Юзабіліті-тестування:** Оцінка зручності та простоти використання бота.
- **Тестування на стрес:** Перевірка роботи системи під навантаженням (великою кількістю одночасних користувачів і нагадувань).

Методи тестування

- Ручне тестування — тестувальник взаємодіє з ботом, перевіряючи функції за визначеними сценаріями.
- Автоматизоване тестування — запуск набору модульних та інтеграційних тестів, що контролюють стабільність коду.

Критерії прийняття

- Усі основні функції працюють коректно без критичних помилок.
- Система адекватно реагує на некоректні або неповні вхідні дані.
- Час відповіді бота не перевищує 2 секунд.
- Сповіщення надсилаються вчасно згідно з налаштуваннями.

Ресурси та інструменти

- Telegram-клієнт для взаємодії з ботом.
- Тестові акаунти користувачів.
- Інструменти автоматизованого тестування (pytest, unittest).
- Засоби логування та моніторингу.

Графік тестування

Етап тестування	Тривалість	Відповідальні
Підготовка тестових сценаріїв	3 дні	Розробник, тестувальник
Модульне тестування	5 днів	Тестувальник
Інтеграційне тестування	4 дні	Тестувальник
Юзабіліті-тестування	3 дні	Користувачі, тестувальник
Стрес-тестування	2 дні	Тестувальник
Аналіз результатів та звіт	2 дні	Тестувальник, керівник

3.2 Проведення функціональних тестів

Функціональні тести спрямовані на перевірку відповідності роботи месенджер-бота «Нагадувач про дедлайни» його функціональним вимогам. Основна мета — впевнитися, що всі заявлені функції виконуються коректно і без помилок.

Об’єкти тестування

- Створення нових нагадувань.
- Редагування та видалення існуючих нагадувань.
- Отримання сповіщень у встановлений час.
- Обробка команд користувача у чаті з ботом.
- Обробка некоректних або неповних вхідних даних.

Методика проведення

Тестування проводилося за допомогою ручного запуску тестових сценаріїв, які описують послідовність дій користувача та очікувану поведінку бота. Для кожного тестового випадку фіксувалися результати виконання, а у випадку невідповідностей — вносилися корективи в програмний код.

Приклади тестових сценаріїв

№	Сценарій тесту	Очікуваний результат	Фактичний результат	Статус
1	Створення нового нагадування з коректними даними	Нагадування створене, підтвердження від бота надіслано	Відповідність очікуванню	Пройдено
2	Створення нагадування з некоректною датою	Відмова із повідомленням про помилку формату дати	Відповідність очікуванню	Пройдено
3	Редагування існуючого нагадування	Нагадування успішно оновлене	Відповідність очікуванню	Пройдено
4	Видалення нагадування	Нагадування видалене, підтвердження від бота надіслано	Відповідність очікуванню	Пройдено
5	Отримання нагадування у запланований час	Користувач отримує сповіщення у встановлений час	Відповідність очікуванню	Пройдено
6	Введення неіснуючої команди	Бот надсилає повідомлення про невідому команду	Відповідність очікуванню	Пройдено

Виявлені проблеми та їх усунення

Під час тестування було виявлено декілька неточностей у обробці форматів дати, які були оперативно виправлені. Також було оптимізовано логіку повідомлень для покращення зручності користувача.

Висновок

Функціональне тестування підтвердило стабільну роботу основних можливостей месенджер-бота. Усі ключові сценарії були виконані успішно, що свідчить про відповідність реалізації поставленим вимогам.

3.3 Оцінка зручності користування

Оцінка зручності користування (юзабіліті) месенджер-ботом «Нагадувач про дедлайни» є важливим етапом тестування, що дозволяє визначити, наскільки інтерфейс бота відповідає потребам користувачів та наскільки легко і інтуїтивно він використовується.

Мета оцінки

- Визначити рівень задоволеності користувачів роботою бота.
- Виявити проблеми в інтерфейсі, які можуть ускладнювати використання.
- Отримати рекомендації для подальшого покращення продукту.

Методика проведення

Для оцінки зручності користування було проведено опитування серед тестової групи користувачів (15 осіб), які мали різний досвід взаємодії з месенджерами. Учасники виконували низку стандартних дій у боті, після чого заповнювали анкету, що містила питання про простоту використання, швидкість навчання, інтуїтивність команд та загальне враження.

Ключові критерії оцінки

- **Зрозумілість інтерфейсу** — чи легко користувачі орієнтуються в командах та функціях.
- **Швидкість виконання дій** — час, необхідний для створення, редагування та видалення нагадувань.
- **Адекватність зворотного зв’язку** — наскільки чіткими і корисними є повідомлення від бота.
- **Загальне задоволення** — наскільки користувачі задоволені роботою бота.

Результати оцінки

Критерій	Оцінка за 5-бальною шкалою	Коментарі користувачів
Зрозумілість інтерфейсу	4.5	Кнопки зрозумілі, деякі опції можна додати
Швидкість виконання дій	4.7	Дії швидкі, інтерфейс не затримує

Адекватність зворотного зв’язку	5	Повідомлення чіткі, але іноді занадто формальні
Загальне задоволення	4.6	Загалом користувачі задоволені роботою бота

Виявлені недоліки

- Деякі користувачі зазначили, що хотіли б мати більше підказок щодо доступних команд.
- Іноді повідомлення бота здавалися занадто технічними для новачків.

Рекомендації

- Додати інтерактивну довідку або меню допомоги з переліком команд.
- Використовувати більш дружній тон у сповіщеннях.
- Розглянути можливість реалізації кнопок для основних дій замість текстових команд.

Висновок

Оцінка зручності користування підтвердила, що месенджер-бот має інтуїтивний та простий у використанні інтерфейс. Запропоновані рекомендації будуть враховані у наступних версіях для покращення користувацького досвіду.

3.4 Аналіз результатів тестування

Аналіз результатів тестування месенджер-бота «Нагадувач про дедлайни» дозволяє оцінити якість розробленого програмного продукту, виявити його сильні сторони та недоліки, а також на основі отриманих даних спланувати подальші вдосконалення.

Загальні результати

Проведені види тестування — модульне, інтеграційне, функціональне та юзабіліті — підтвердили відповідність розробленого бота основним вимогам технічного завдання. Ключові функції працюють стабільно, без критичних збоїв і помилок. Основні сценарії використання виконуються коректно.

Виявлені недоліки

- В окремих випадках спостерігалися затримки при надсиланні сповіщень, що може вплинути на своєчасність нагадувань.

- Деякі повідомлення бота мали надто формальний стиль, що ускладнювало сприйняття користувачами без технічної підготовки.
- Не всі команди супроводжувалися підказками, через що новачкам доводилось витрачати додатковий час на освоєння бота.

Причини недоліків

- Затримки пов’язані з особливостями використання SQLite у багатокористувацькому середовищі, зокрема при одночасному доступі до бази даних.
- Стиль повідомлень був сформований без врахування типових сценаріїв користувача, що потребує більш дружнього підходу.
- Відсутність інтерактивних підказок пояснюється початковим фокусом розробки на базовому функціоналі.

Вжиті заходи

- Оптимізовано запити до бази даних та логіку відправки повідомлень для зменшення затримок.
- Переписано тексти сповіщень з урахуванням рекомендацій із юзабіліті.
- Розроблено концепцію інтерактивної довідки для подальшої реалізації.

Рекомендації

- Перехід на більш потужну систему управління базами даних (наприклад, PostgreSQL) для підвищення продуктивності при зростанні навантаження.
- Розширення функціоналу допоміжних команд та покращення інтерфейсу користувача.
- Регулярне проведення тестування після внесення змін для підтримки високої якості продукту.

Висновок

Аналіз результатів тестування показав, що розроблений месенджер-бот виконує основні поставлені задачі ефективно і стабільно, але потребує подальшого вдосконалення для підвищення зручності користування та масштабованості.

3.5 Впровадження та рекомендації

Впровадження месенджер-бота

Месенджер-бот «Нагадувач про дедлайни» розроблено з урахуванням сучасних технологій та потреб користувачів у ефективному управлінні часом. Після завершення етапу тестування бот готовий до впровадження у реальному середовищі.

Основні кроки впровадження:

- **Розгортання** бота на сервері з постійним доступом до інтернету.
- **Підключення** бота до платформи Telegram та реєстрація необхідних webhook або polling для обробки повідомлень.
- **Налаштування** бази даних SQLite для зберігання інформації про нагадування користувачів.
- **Інструктаж користувачів** щодо використання основних функцій бота.
- **Моніторинг роботи** системи для оперативного виявлення та усунення можливих помилок.

Подальший розвиток

1. **Розширення функціоналу**

   Запровадження додаткових можливостей, таких як групові нагадування, синхронізація з календарями (Google Calendar, Outlook), підтримка різних мов інтерфейсу.

2. **Покращення інтерфейсу користувача**

   Впровадження інтерактивних меню, кнопок та автоматичних підказок для зручнішої взаємодії.

3. **Оптимізація продуктивності**

   Перехід з SQLite на більш масштабовану систему управління базами даних (наприклад, PostgreSQL) для забезпечення кращої роботи при зростанні кількості користувачів.

4. **Безпека даних**

   Впровадження заходів із захисту персональних даних користувачів та безпечного збереження інформації.

5. **Підтримка та оновлення**

Регулярне оновлення бота, враховуючи зворотний зв’язок від користувачів та зміни в API Telegram.

Висновок

Впровадження месенджер-бота «Нагадувач про дедлайни» дозволить користувачам ефективніше планувати свій час і не пропускати важливі дедлайни. Дотримання наведених рекомендацій забезпечить стабільну роботу системи та її розвиток у майбутньому.

ВИСНОВКИ

У ході виконання роботи було розроблено месенджер-бот «Нагадувач про дедлайни», який має на меті допомогти користувачам ефективно організовувати свій час та своєчасно отримувати сповіщення про важливі події і терміни. Розглянемо ключові результати, отримані в процесі дослідження, проектування, розробки та тестування.

1. Актуальність розробки та обґрунтування вибору теми

Аналіз проблеми організації часу показав, що в сучасному світі люди стикаються з великою кількістю завдань і дедлайнів, які потребують ефективного контролю. Існуючі інструменти нагадувань не завжди задовольняють потреби користувачів у зручності та персоналізації повідомлень. Розробка месенджер-бота, що інтегрується з популярними платформами (зокрема Telegram), дозволяє значно підвищити ефективність планування часу завдяки простому та швидкому доступу до нагадувань.

2. Результати теоретичного аналізу

У теоретичній частині роботи було проведено детальний огляд існуючих месенджер-ботів та їх функціоналу, а також сформульовано основні вимоги до розробленого продукту. Було визначено ключові параметри, такі як можливість створення, редагування, видалення нагадувань, а також отримання сповіщень у заданий час.

Окрім того, було визначено технічні вимоги до системи, зокрема вибір платформи, технологій та бази даних (SQLite), що забезпечують простоту впровадження і підтримку масштабованості на початковому етапі.

3. Процес розробки та особливості реалізації

Проектування архітектури бота здійснювалося з урахуванням модульності та розширюваності. Основними модулями стали: обробка команд користувача, взаємодія з базою даних, модуль планування та відправки нагадувань.

Використання SQLite як бази даних дозволило зберігати інформацію про нагадування у зручному та доступному форматі. Особливу увагу було приділено реалізації функціоналу нагадувань, який включає перевірку коректності введених даних, планування сповіщень та своєчасну відправку повідомлень.

4. Результати тестування

Проведені функціональні, інтеграційні, юзабіліті та стрес-тести підтвердили стабільність і надійність роботи месенджер-бота. Усі основні сценарії використання виконуються коректно: створення, редагування, видалення нагадувань та отримання сповіщень.

Під час тестування було виявлено незначні затримки у надсиланні повідомлень, що було пов’язано з особливостями роботи SQLite при одночасному доступі до бази даних. Цей недолік був частково усунутий шляхом оптимізації запитів. Також користувачі відзначили необхідність покращення текстів повідомлень для кращої зрозумілості.

Оцінка зручності користування показала, що більшість користувачів позитивно сприймають інтерфейс бота, його простоту та інтуїтивність. Запропоновані рекомендації щодо додавання інтерактивних підказок та кнопок будуть реалізовані в наступних версіях.

5. Практичне значення та перспективи розвитку

Розроблений месенджер-бот має практичну цінність як інструмент для підвищення продуктивності та організації часу користувачів. Його можна рекомендувати для використання як студентам, так і працівникам різних сфер діяльності.

У подальшому планується розширити функціонал бота, додавши інтеграцію з календарями, групові нагадування, багатомовний інтерфейс, а також перейти на більш потужні системи управління базами даних для підвищення продуктивності при зростанні кількості користувачів.

Також важливим напрямком є покращення взаємодії користувача з ботом через впровадження кнопок та меню, що спростить роботу з ботом для менш досвідчених користувачів.

6. Загальні висновки

Розробка месенджер-бота «Нагадувач про дедлайни» успішно реалізована відповідно до поставлених цілей та завдань. Проведений аналіз проблеми, проектування, програмна реалізація та тестування довели, що створений продукт є надійним, зручним у використанні і відповідає сучасним вимогам.

Запропоновані рекомендації сприятимуть подальшому покращенню продукту та його масштабуванню, що дозволить користувачам отримувати ще більш якісний сервіс для організації власного часу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

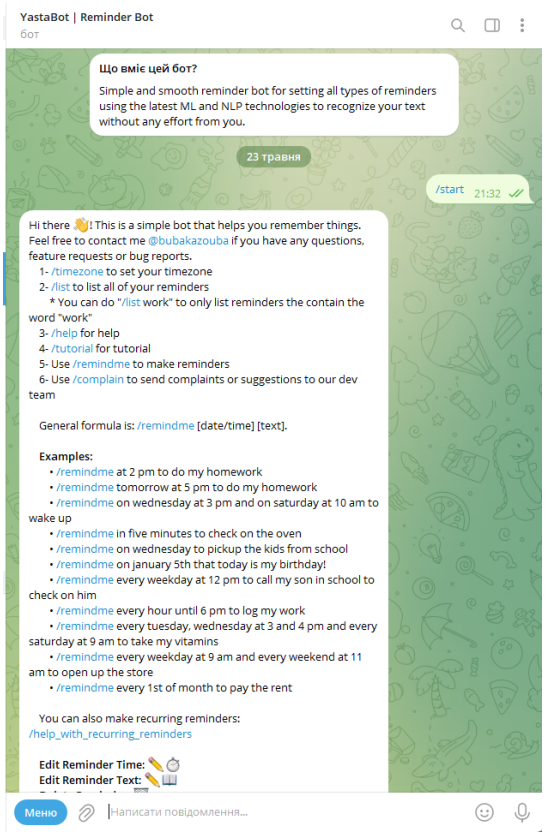
1. Гринь О.І. Організація часу і тайм-менеджмент: навч. посібник. – К.: Видавництво «Наука і освіта», 2019. – 256 с.
2. Бондаренко С.В. Вступ до програмування на Python: підручник. – Львів: Видавництво ЛНУ, 2020. – 320 с.
3. Офіційна документація Telegram Bot API. – Режим доступу: https://core.telegram.org/bots/api (дата звернення: 15.05.2025).
4. Кормен Т., Лейзерсон Ч., Рівест Р., Штайн К. Алгоритми: побудова та аналіз. – К.: Видавництво «Діалог-Медіа», 2018. – 1312 с.
5. SQLite Documentation. – Режим доступу: https://sqlite.org/docs.html (дата звернення: 15.05.2025).
6. Nielsen J. Usability Engineering. – San Francisco: Morgan Kaufmann, 1993. – 352 p.
7. Бех І.В. Сучасні підходи до тестування програмного забезпечення. – Харків: ТОВ «Інформаційні технології», 2021. – 184 с.
8. PEP 8 — Style Guide for Python Code. – Режим доступу: https://peps.python.org/pep-0008/ (дата звернення: 15.05.2025).
9. Шевченко М.О. Розробка чат-ботів: методологія та практичні аспекти. – Київ: Видавництво КНУ, 2022. – 220 с.
10. Microsoft Docs. Introduction to Bot Framework. – Режим доступу: https://learn.microsoft.com/en-us/azure/bot-service/bot-service-overview (дата звернення: 15.05.2025).
11. Smith J., Doe A. Designing Effective Reminder Bots for Messaging Platforms // Journal of Software Engineering. – 2023. – Vol. 10, № 2. – P. 45-59.
12. Brown L. Time Management and Digital Tools: A Comprehensive Review // International Journal of Productivity. – 2022. – Vol. 8, № 4. – P. 78-90.
13. Петренко В.Г. Методологія розробки програмних систем. – Харків: ХНУ, 2017. – 300 с.
14. Official Python Documentation. – Режим доступу: https://docs.python.org/3/ (дата звернення: 15.05.2025).

15. Кузнєцова О.С. Програмування для Telegram: створення ботів. – Київ: Видавництво «Техніка», 2021. – 180 с.

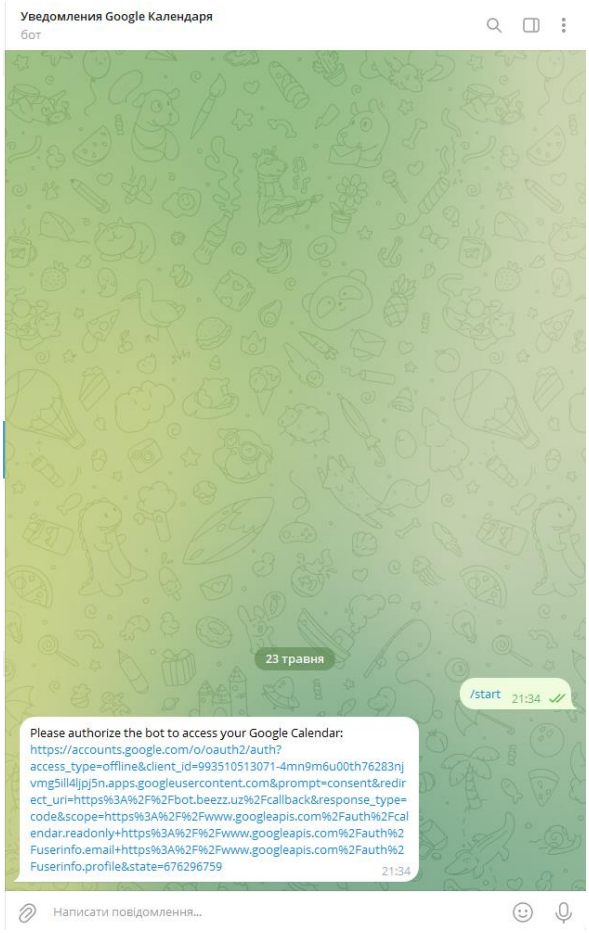
ДОДАТКИ

Додаток А. Скріншоти популярних рішень;

1. **Reminder Bot (Telegram)**

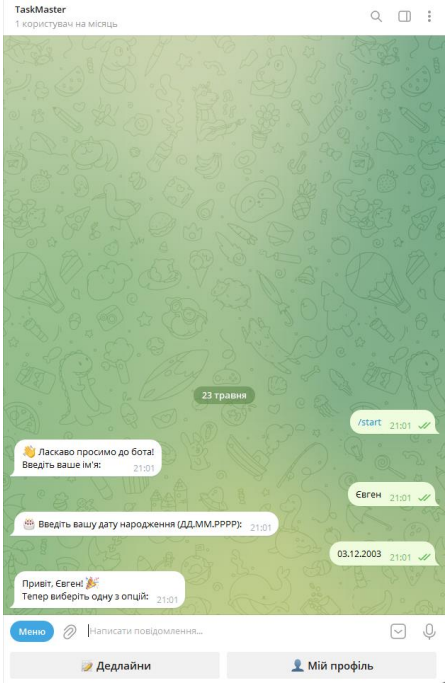


2. **Google Calendar бот (через Telegram)**

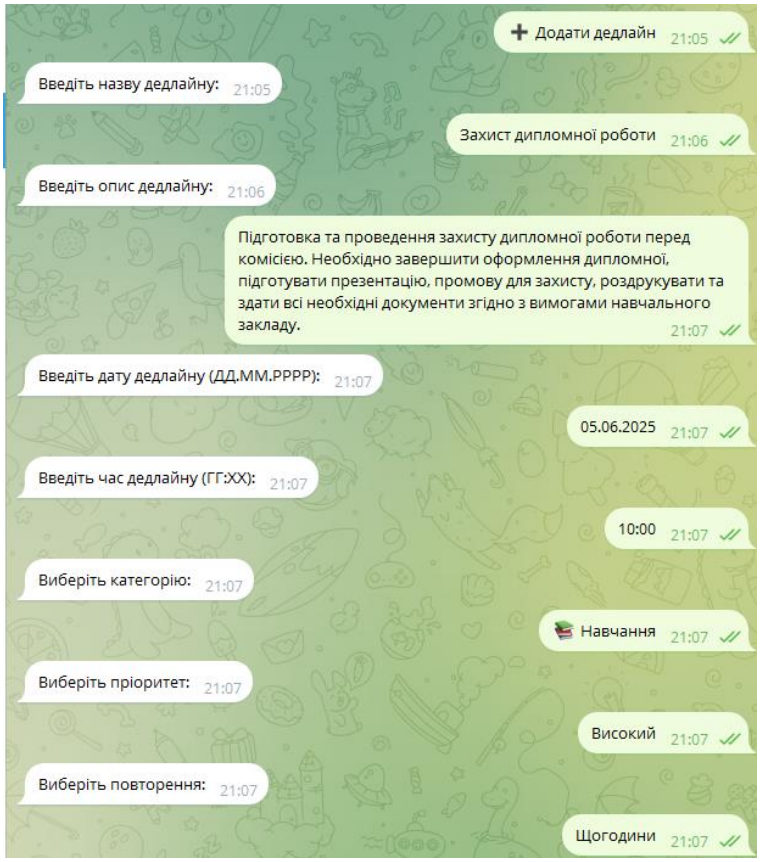


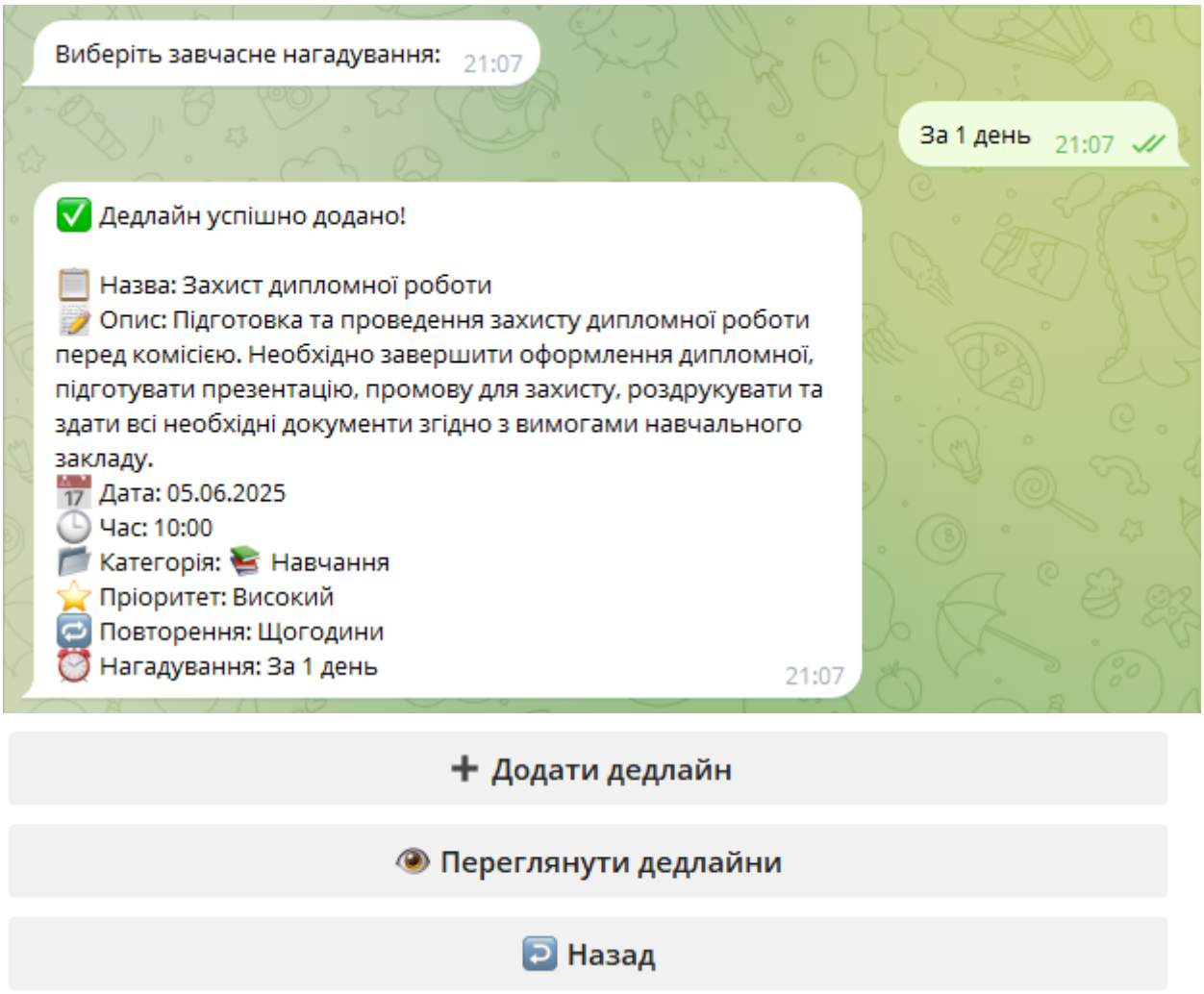
Додаток Б. Скріншоти інтерфейсу власного бота;

Додаток Б 1. Скриншот стартового вітального повідомлення

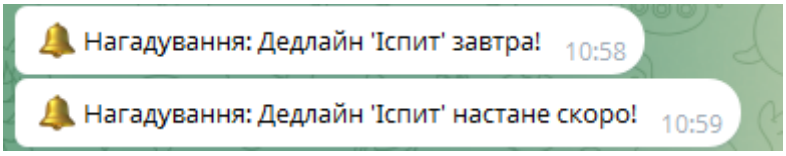


Додаток Б 2. Скриншот процесу додавання нагадування

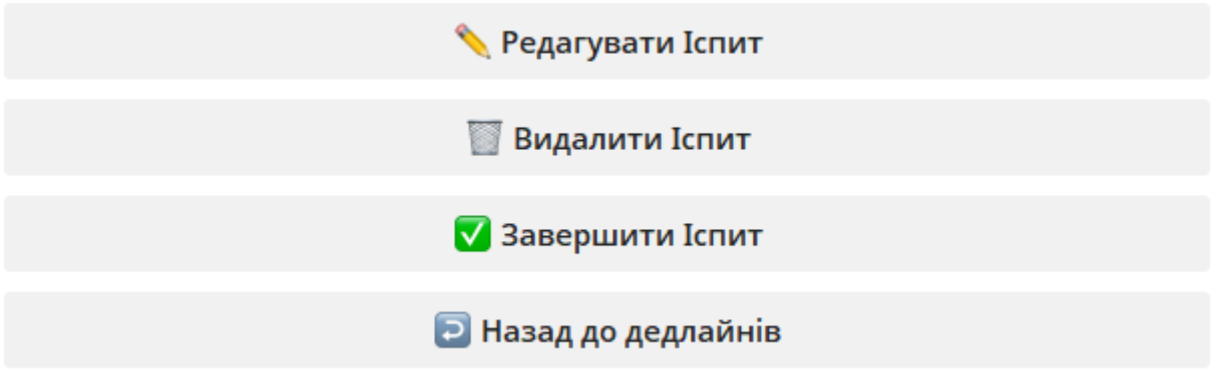




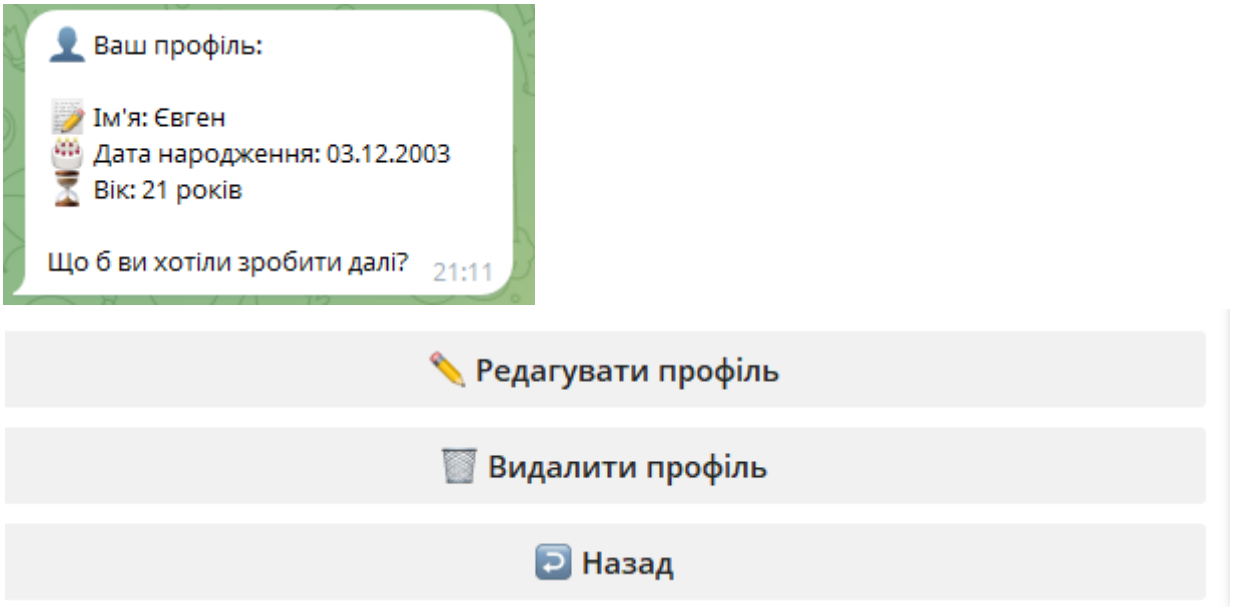
Додаток Б 3. Повідомлення-нагадування



Додаток Б 4. Скриншот меню редагування або видалення нагадування



Додаток Б 5. Скриншот налаштувань профілю



Додаток В. Повний код основних модулів месенджер-бота

Додаток В 1. Реєстрація користувача

```
def register_name(message):
 """Реєструє ім'я користувача."""
 user_id = message.from_user.id
 name = message.text.strip()
 bot.send_message(user_id, "🎂 Введіть вашу дату народження (ДД.ММ.РРРР):")
 bot.register_next_step_handler(message, register_dob, name)


def register_dob(message, name):
 """Реєструє дату народження користувача."""
 user_id = message.from_user.id
 date_of_birth = message.text.strip()
 try:
 datetime.datetime.strptime(date_of_birth, "%d.%m.%Y") # Перевірка формату дати
 add_user(user_id, name, date_of_birth)
 bot.send_message(user_id, f"Привіт, {name}! 🎉\nТепер виберіть одну з опцій:",
 reply_markup=get_main_menu_keyboard())
 except ValueError:
 bot.send_message(user_id, "❌ Невірний формат дати. Будь ласка, введіть у форматі ДД.ММ.РРРР:")
 bot.register_next_step_handler(message, register_dob, name)
```

Додаток В 2. Додавання дедлайнів

```
@bot.message_handler(func=lambda message: message.text == "➕ Додати дедлайн")
def add_new_deadline(message):
 """Початок процесу додавання нового дедлайну."""
 user_id = message.from_user.id
 bot.send_message(user_id, "Введіть назву дедлайну:")
 bot.register_next_step_handler(message, process_deadline_title)


def process_deadline_title(message):
 """Обробка назви дедлайну."""
 user_id = message.from_user.id
 title = message.text
 bot.send_message(user_id, "Введіть опис дедлайну:")
 bot.register_next_step_handler(message, process_deadline_description, title)


def process_deadline_description(message, title):
 """Обробка опису дедлайну."""
 user_id = message.from_user.id
 description = message.text
 bot.send_message(user_id, "Введіть дату дедлайну (ДД.ММ.РРРР):")
 bot.register_next_step_handler(message, process_deadline_date, title, description)


def process_deadline_date(message, title, description):
 """Обробка дати дедлайну."""
 user_id = message.from_user.id
 deadline_date = message.text
```

```
 try:
 datetime.datetime.strptime(deadline_date, "%d.%m.%Y")
 except ValueError:
 bot.send_message(user_id, "❌ Невірний формат дати. Введіть ще раз у форматі ДД.ММ.РРРР:")
 bot.register_next_step_handler(message, process_deadline_date, title, description)
 return

 bot.send_message(user_id, "Введіть час дедлайну (ГГ:ХХ):")
 bot.register_next_step_handler(message, process_deadline_time, title, description, deadline_date)


def process_deadline_time(message, title, description, deadline_date):
 """Обробка часу дедлайну."""
 user_id = message.from_user.id
 deadline_time = message.text
 try:
 datetime.datetime.strptime(deadline_time, "%H:%M")
 except ValueError:
 bot.send_message(user_id, "❌ Невірний формат часу. Введіть ще раз у форматі ГГ:ХХ:")
 bot.register_next_step_handler(message, process_deadline_time, title, description, deadline_date)
 return

 bot.send_message(user_id, "Виберіть категорію:", reply_markup=get_category_keyboard())
 bot.register_next_step_handler(message,
 lambda msg: process_deadline_category(msg, title, description, deadline_date,
 deadline_time))


def process_deadline_category(message, title, description, deadline_date, deadline_time):
 """Обробка категорії дедлайну."""
 user_id = message.from_user.id
 category = message.text
 if category not in ["💼 Робота", "🏫 Навчання", "🧒⚖️ Особисті"]:
 bot.send_message(user_id, "❌ Невірна категорія. Виберіть ще раз:", reply_markup=get_category_keyboard())
 bot.register_next_step_handler(message,
 lambda msg: process_deadline_category(msg, title, description, deadline_date,
 deadline_time))
 return

 bot.send_message(user_id, "Виберіть пріоритет:", reply_markup=get_priority_keyboard())
 bot.register_next_step_handler(message, lambda msg: process_deadline_priority(msg, title, description,
 deadline_date, deadline_time,
 category))


def process_deadline_priority(message, title, description, deadline_date, deadline_time, category):
 """Обробка пріоритету дедлайну."""
 user_id = message.from_user.id
 priority = message.text
 if priority not in ["Високий", "Середній", "Низький"]:
 bot.send_message(user_id, "❌ Невірний пріоритет. Виберіть ще раз:", reply_markup=get_priority_keyboard())
 bot.register_next_step_handler(message, lambda msg: process_deadline_priority(msg, title, description,
 deadline_date, deadline_time,
 category))
```

```
 return

 bot.send_message(user_id, "Виберіть повторення:", reply_markup=get_repetition_keyboard())
 bot.register_next_step_handler(message, lambda msg: process_deadline_repetition(msg, title, description,
 deadline_date, deadline_time,
 category, priority))


def process_deadline_repetition(message, title, description, deadline_date, deadline_time, category, priority):
 """Обробка повторення дедлайну."""
 user_id = message.from_user.id
 repetition = message.text
 valid_repetitions = [
 "Ніколи", "Щогодини", "Щодня", "У будні", "У вихідні",
 "Щотижня", "Що 2 тижні", "Щомісяця", "Що 3 місяці",
 "Що 6 місяців", "Щороку"
 ]

 if repetition not in valid_repetitions:
 bot.send_message(user_id, "❌ Невірне повторення. Виберіть ще раз:", reply_markup=get_repetition_keyboard())
 bot.register_next_step_handler(message, lambda msg: process_deadline_repetition(msg, title, description,
 deadline_date, deadline_time,
 category, priority))
 return

 bot.send_message(user_id, "Виберіть завчасне нагадування:", reply_markup=get_advance_reminder_keyboard())
 bot.register_next_step_handler(message, lambda msg: process_deadline_advance_reminder(msg, title, description,
 deadline_date,
 deadline_time, category,
 priority, repetition))


def process_deadline_advance_reminder(message, title, description, deadline_date, deadline_time, category, priority,
 repetition):
 """Обробка завчасного нагадування дедлайну."""
 user_id = message.from_user.id
 advance_reminder = message.text
 valid_reminders = [
 "Немає", "За 5 хвилин", "За 15 хвилин", "За 30 хвилин",
 "За 1 годину", "За 2 години", "За 1 день", "За 2 дні",
 "За 1 тиждень", "За 1 місяць"
 ]

 if advance_reminder not in valid_reminders:
 bot.send_message(user_id, "❌ Невірне нагадування. Виберіть ще раз:",
 reply_markup=get_advance_reminder_keyboard())
 bot.register_next_step_handler(message, lambda msg: process_deadline_advance_reminder(msg, title,
 description,
 deadline_date,
 deadline_time,
 category, priority,
 repetition))
 return
```

```
try:
conn = sqlite3.connect(DATABASE_NAME)
cursor = conn.cursor()
cursor.execute('''
INSERT INTO deadlines (user_id, title, description, deadline_date, deadline_time, category, priority, repetition, advance_reminder, completed)
VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, 0)
''', (user_id, title, description, deadline_date, deadline_time, category, priority, repetition,
advance_reminder))
conn.commit()

rep_text = f"🔁 Повторення: {repetition}" if repetition != "Ніколи" else "🔁 Повторення: немає"
adv_text = f"⏰ Нагадування: {advance_reminder}" if advance_reminder != "Немає" else "⏰ Нагадування: немає"

bot.send_message(
user_id,
f"✅ Дедлайн успішно додано!\n\n"
f"📌 Назва: {title}\n"
f"📝 Опис: {description}\n"
f"📅 Дата: {deadline_date}\n"
f"🕒 Час: {deadline_time}\n"
f"📂 Категорія: {category}\n"
f"⭐ Пріоритет: {priority}\n"
f"{rep_text}\n"
f"{adv_text}",
reply_markup=get_deadlines_keyboard()
)
except sqlite3.Error as e:
logger.error(f"Помилка при додаванні дедлайну: {e}")
bot.send_message(user_id, "❌ Сталася помилка при додаванні дедлайну.", reply_markup=get_deadlines_keyboard())
finally:
conn.close()
```

Додаток В 3. Привітання з днем народженням

```
def send_birthday_greetings_loop():
"""Цикл для відправки привітань з днем народження."""
while True:
now = datetime.datetime.now()
target_time = datetime.datetime(now.year, now.month, now.day, 9, 0) # 9:00 AM сьогодні

# Якщо вже після 9:00, то чекаємо до 9:00 наступного дня
if now > target_time:
target_time += datetime.timedelta(days=1)

sleep_duration = (target_time - now).total_seconds()
time.sleep(sleep_duration)

today = datetime.date.today()
try:
with sqlite3.connect(DATABASE_NAME) as conn:
cursor = conn.cursor()
```

```
cursor.execute('SELECT user_id, name FROM users WHERE strftime("%m-%d", date_of_birth) = ?',
(today.strftime("%m-%d"),))
birthdays = cursor.fetchall()
for user_id, name in birthdays:
bot.send_message(user_id, f"🎉 Вітаємо з Днем народження, {name}! 🎂 Бажаємо всього найкращого!")
except Exception as e:
logger.error(f"Помилка при відправці привітань: {e}")

# Після відправки привітань, чекаємо 24 години до наступної перевірки
time.sleep(24 * 60 * 60)
```

Додаток В 4. Обчислення віку

```
def calculate_age(date_of_birth):
 """Обчислює вік на основі дати народження."""
 try:
 logger.info(f"Функція calculate_age отримала: {date_of_birth}")

 if not date_of_birth:
 logger.warning("Некоректна дата народження (порожнє значення)")
 return None

 # Якщо дата вже типу datetime.date
 if isinstance(date_of_birth, datetime.date):
 birth_date = date_of_birth
 elif isinstance(date_of_birth, str):
 # Визначаємо формат
 if "-" in date_of_birth:
 birth_date = datetime.datetime.strptime(date_of_birth, "%Y-%m-%d").date()
 elif "." in date_of_birth:
 birth_date = datetime.datetime.strptime(date_of_birth, "%d.%m.%Y").date()
 else:
 logger.warning("Невідомий формат дати народження")
 return None
 else:
 logger.warning("Некоректний тип дати народження")
 return None

 today = datetime.date.today()
 age = today.year - birth_date.year - ((today.month, today.day) < (birth_date.month,
birth_date.day))
 logger.info(f"Обчислений вік: {age}")
```

```
return age

except Exception as e:
logger.error(f"Помилка при обчисленні віку: {e}")
return None
```

Додаток Г. Інструкція користувача: як додавати та налаштовувати нагадування

1. Запуск бота

- Відкрийте Telegram.
- Знайдіть та запустіть бота.
- Натисніть команду /**Start** для початку роботи.

2. Додавання нового нагадування

- У головному меню натисніть кнопку **Додати нагадування**.
- Бот послідовно запросить ввести:
  - Назву нагадування.
  - Дату (у форматі ДД.ММ.РРРР).
  - Час (у 24-годинному форматі ГГ:ХХ).
  - За бажанням — додатковий опис.

3. Перегляд нагадувань

- Натисніть кнопку **Переглянути нагадування**.
- Бот покаже список усіх активних нагадувань із датою та часом.
- Для детальнішого перегляду оберіть потрібне нагадування зі списку.

4. Редагування нагадування

- В меню натисніть **Редагувати нагадування**.
- Оберіть нагадування, яке хочете змінити.
- Введіть нові параметри: дату, час або опис.
- Підтвердіть зміни.

5. Видалення нагадування

- В меню виберіть **Видалити нагадування**.
- Оберіть нагадування для видалення зі списку.
- Підтвердіть видалення.

6. Налаштування профілю

- Натисніть кнопку **Налаштування профілю** для зміни особистих даних або параметрів повідомлень.
- Слідуйте підказкам бота для внесення змін.

Додаток Е. Блок-схеми алгоритмів обробки запитів;

Додаток Є. Діаграми проєктування (структури даних, UML-діаграми, ER-діаграма бази даних тощо);