Міністерство освіти і науки України
Харківський національний університет імені В. Н. Каразіна
Навчально-науковий інститут комп'ютерних наук та штучного інтелекту
Кафедра комп'ютерних систем та робототехніки

«Затверджую»
в.о. завідуючого кафедри
комп'ютерних систем та робототехніки
_____ к. ф.-м. н., доцент Максим ХРУСЛОВ
«___» червня 2025 р.

Пояснювальна записка

до кваліфікаційної роботи
бакалавра

на тему: «КОМП'ЮТЕРНА МОДЕЛЬ КОНТРОЛЮ ПРИЙОМУ МЕДИКАМЕНТІВ НА ОСНОВІ TELEGRAM-БОТА З ВИКОРИСТАННЯМ ШТУЧНОГО ІНТЕЛЕКТУ»

Спеціальність 151 – Автоматизація, комп'ютерно-інтегровані технології
Галузь знань 15 – Автоматизація та приладобудування
Освітня програма «Автоматизація та комп'ютерно-інтегровані технології»

Захищено на засіданні
Екзаменаційної комісії № 46
протокол № __ від __.06.2025 р.
Оцінка _____ / _____
Голова Екзаменаційної комісії
_____ **ЧУГАЙ А. М.**

Виконав:
Студент групи КУ– 41
ЛУЦЕНКО Едуард Арсенович
_____

Керівник: доцент, доцент зво кафедри КСР, к.т.н.
Булавін Дмитро Олексійович _____

Рецензент: д.т.н., доцент, професор кафедри ТПІ
Руккас Кирило Маркович _____

Харків – 2025

АНОТАЦІЯ

Пояснювальна записка до кваліфікаційної роботи магістра складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків. Загальний обсяг роботи – 50 сторінок, з яких 37 сторінок становить основна частина, ілюстрована 10 рисунками. Список використаних джерел містить 8 найменувань.

Мета роботи – розробити комп'ютерну модель системи контролю прийому медикаментів на основі Telegram-бота

Об'єкт дослідження – інформаційно-комунікаційні системи для автоматизації нагадувань про прийом лікарських засобів.

Предмет дослідження – методи проєктування Telegram-бота, модулі планування нагадувань і штучного інтелекту для персоналізації сповіщень.

Актуальність теми в необхідності підвищення точності та своєчасності прийому ліків пацієнтами, зниження ризиків пропусків доз і ускладнень лікування шляхом використання зручних інтерфейсів і адаптивних AI-рішень.

У роботі:

- проведено теоретичний аналіз Telegram Bot API, методів машинного навчання та стратегій планування нагадувань;
- спроектовано архітектуру системи, структуру бази даних і модулі безпеки (TLS, AES, аутентифікація за Telegram ID);
- реалізовано Telegram-бота на Python (aiogram, SQLAlchemy), інтегровано AI-модуль (TensorFlow), налаштовано планувальник APScheduler та обробку винятків;

***Ключові слова*:** TELEGRAM-BOT, ШТУЧНИЙ ІНТЕЛЕКТ, НАГАДУВАЧ ПРИЙОМУ ЛІКІВ, APSCHEDULER, TLS, AES, AI-МОДУЛЬ, PYTHON, SQLALCHEMY, PYTEST.

ABSTRACT

The master's thesis explanatory note consists of an introduction, three chapters, conclusions, a list of references, and appendices. The total length of the work is 50 pages, of which 37 pages comprise the main body, illustrated with 10 figures. The list of references includes 8 items.

Objective of the study – to develop a computer model of a medication intake control system based on a Telegram bot.

Research object – information and communication systems for automating medication-intake reminders.

Research subject – design methods for the Telegram bot, reminder-scheduling modules, and artificial intelligence for personalized notifications.

Relevance of the topic stems from the need to improve the accuracy and timeliness of patients' medication intake, reduce the risk of missed doses and treatment complications through the use of user-friendly interfaces and adaptive AI solutions.

In this work:

- a theoretical analysis of the Telegram Bot API, machine learning methods, and reminder-scheduling strategies was conducted;
- the system architecture, database structure, and security modules (TLS, AES, Telegram ID authentication) were designed;
- a Telegram bot was implemented in Python (aiogram, SQLAlchemy), an AI module was integrated (TensorFlow), and the APScheduler planner and exception-handling routines were configured.

Keywords: TELEGRAM BOT, ARTIFICIAL INTELLIGENCE, MEDICATION REMINDER, APSCHEDULER, TLS, AES, AI MODULE, PYTHON, SQLALCHEMY, PYTEST.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ 5

ВСТУП 6

РОЗДІЛ 1 8

ТЕОРЕТИЧНІ АСПЕКТИ КОНТРОЛЮ ПРИЙОМУ МЕДИКАМЕНТІВ ЗА ДОПОМОГОЮ TELEGRAM-БОТА ТА ШІ 8

1.1. Огляд архітектури Telegram API та особливості ботів-нагадувачів 8

1.2. Методи штучного інтелекту для персоналізації рекомендацій 9

1.3. Стратегії планування нагадувань та обробки винятків 12

1.4. Вимоги конфіденційності та безпеки персональних медичних даних 14

Висновки до розділу 1 15

РОЗДІЛ 2 17

АРХІТЕКТУРА ТА ПРОЕКТУВАННЯ СИСТЕМИ КОНТРОЛЮ ПРИЙОМУ МЕДИКАМЕНТІВ 17

2.1 Архітектура рішення та структура бази даних 17

2.2 Проєктування модуля нагадувань (регулярні та разові задачі) 19

2.3 Проєктування AI-модуля для аналізу й персоналізації 22

2.4 Проєктування заходів безпеки та шифрування даних 25

Висновки до розділу 2 27

РОЗДІЛ 3 29

РЕАЛІЗАЦІЯ СИСТЕМИ КОМП'ЮТЕРНОГО КОНТРОЛЮ ПРИЙОМУ МЕДИКАМЕНТІВ НА ОСНОВІ TELEGRAM-БОТА З ВИКОРИСТАННЯМ ШІ 29

3.1. Реалізація Telegram-бота: основні функції 29

3.2. Інтеграція та налаштування AI-модуля 32

3.3. Впровадження механізмів планування нагадувань і обробки винятків 34

3.4. Тестування функціональності, продуктивності та надійності 36

Висновки до розділу 3 38

ВИСНОВКИ 39

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 40

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

API	—	Application Programming Interface
DB	—	Database
TLS	—	Transport Layer Security
HTTPS	—	HTTP Secure
AES	—	Advanced Encryption Standard
CRUD	—	Create, Read, Update, Delete
AI	—	Artificial Intelligence
ML	—	Machine Learning
NLP	—	Natural Language Processing
UMS	—	Universal Medication Schedule
ORM	—	Object-Relational Mapping
JWT	—	JSON Web Token
RL	—	Reinforcement Learning

ВСТУП

Актуальність дослідження. Своєчасний та коректний прийом лікарських препаратів є ключовим чинником ефективності лікування хронічних та гострих захворювань. Проте численні дослідження свідчать про низький рівень комплаєнсу пацієнтів: пропуски доз, неправильні інтервали між прийомами та підвищений ризик ускладнень. Телемедичні рішення на базі месенджер-ботів здатні забезпечити зручний, доступний 24/7 інтерфейс для пацієнта й нагадувати про прийом ліків. Використання штучного інтелекту дозволяє додатково персоналізувати сповіщення залежно від поведінкових патернів користувача, що актуалізує тему дослідження.

Об’єкт дослідження. Інтерактивні інформаційно-комунікаційні системи для нагадування про прийом медикаментів із використанням месенджер-інтерфейсів.

Предмет дослідження. Методика побудови та інтеграції Telegram-бота з модулями планування нагадувань і штучного інтелекту для персоналізації рекомендацій.

Мета роботи. Розробити та обґрунтувати комп’ютерну модель системи контролю прийому медикаментів на основі Telegram-бота із застосуванням алгоритмів штучного інтелекту для підвищення рівня комплаєнсу пацієнтів.

Завдання дослідження

1. Проаналізувати технологічні особливості Telegram Bot API та існуючі підходи до реалізації бот-нагадувачів.
2. Розробити структуру бази даних і архітектуру серверної логіки для зберігання профілів, розкладів і журналу прийомів.
3. Спроектувати модуль планування регулярних і разових нагадувань із механізмами обробки винятків.

4. Інтегрувати AI-модуль для аналізу текстових повідомлень і персоналізації рекомендацій.
5. Впровадити заходи безпеки й шифрування даних відповідно до найкращих практик та нормативних вимог.
6. Реалізувати прототип системи і провести комплексне тестування функціональності, продуктивності та надійності.

Методи дослідження

- **Аналіз літератури та огляд технологій** (Telegram API, планувальники, AI-фреймворки).
- **Методи структурного й об’єктного проєктування** (ER-діаграми, UML-схеми, модульне розбиття).
- **Емпіричне моделювання** (розробка прототипу Telegram-бота на Python із aiogram та SQLAlchemy).
- **Методи машинного навчання** (класифікація тексту, LSTM-моделі, sklearn/TensorFlow).
- **Тестування** (функціональне за допомогою PyTest, навантажувальне з Locust, юніт-тести).

Практичне значення роботи. Запропонована модель може бути безпосередньо впроваджена у лікарських закладах, телемедицині та фармацевтичних сервісах для автоматизації контролю прийому ліків. Система сприятиме підвищенню комплаєнсу пацієнтів, зниженню ризику пропусків доз і ускладнень, а також забезпечить індивідуальний підхід завдяки механізмам штучного інтелекту. У майбутньому рішення може бути розширене інтеграцією із смарт-пристроями для моніторингу стану здоров’я в реальному часі.

РОЗДІЛ 1
ТЕОРЕТИЧНІ АСПЕКТИ КОНТРОЛЮ ПРИЙОМУ МЕДИКАМЕНТІВ ЗА ДОПОМОГОЮ TELEGRAM-БОТА ТА ШІ

1.1. Огляд архітектури Telegram API та особливості ботів-нагадувачів

Архітектура Telegram побудована за клієнт-серверною моделлю з елементами peer-to-peer і хмарним сховищем даних [2]. Усі повідомлення у «хмарних» чатах шифруються перед зберіганням і розподілені по кількох дата-центрах [6], що підвищує надійність та безпеку системи. Telegram Bot API — це HTTP-інтерфейс, який надає JSON-кодувані відповіді на запити бота [2]. Бот-компонента є скриптом чи програмою, що надсилає HTTPS-запити до серверів Telegram і отримує відповіді у форматі JSON [1] [8]. Таким чином комунікація з Bot API відбувається через захищене SSL/TLS-з'єднанн [1], що гарантує конфіденційність даних при передачі.

Telegram-боти підтримують різноманітні види вводу та взаємодії. Наприклад, користувач може надсилати боту текстові повідомлення, зображення, файли, геолокацію, стікери, голосові повідомлення тощо [8]. Боти розпізнають спеціальні команди (/назва) і можуть надавати інтерфейс з клавіатурами та кнопками для вводу користувачем [8]. Інтерактивність розширюється інлайн-клавіатурами, спливними формами та можливістю створювати власні веб-додатки (Web Apps) для глибокої кастомізації. Для отримання оновлень бот може використовувати два основні механізми: опитування серверу (метод getUpdates) або веб-хуки. При опитуванні програма періодично запитує Telegram-сервер про нові повідомлення, тоді як із веб-хуками сервер сам надсилає події на заздалегідь вказаний URL-адрес.

Боти-нагадувачі зазвичай реалізовуються як серверні додатки на платформах Python, JavaScript, PHP тощо. Існують популярні бібліотеки (наприклад, *python-telegram-bot*, *aiogram*, *node-telegram-bot-api*), які спрощують роботу з API та мають механізми планування завдань (job queues) і зберігання

даних. Зазвичай реалізується база даних (SQL/NoSQL) для зберігання профілів користувачів і графіків прийому ліків. Такі боти обслуговують множину користувачів одночасно, тому архітектура проекту може бути клієнт-серверною: сам Telegram є рівнем-проксі між ботом і мобільним клієнтом [2].

Особливості Telegram-бота-нагадувача:

- **Мультимедійні повідомлення:** підтримка тексту, файлів, картинок, голосових чи відеоповідомлень для надання інструкцій (наприклад, фото упаковки ліків) [8].
- **Інтерактивні елементи:** використання команди (наприклад, /start, /add, /list, /done) і кнопок («Так/Ні», меню вводу) для зручного введення та управління графіком.
- **Налаштування розкладу:** можливість встановити повторювані задачі (наприклад, за допомогою cron-like планувальника чи APScheduler) для регулярного надсилання нагадувань в потрібний час.
- **Збереження стану:** ведення журналу прийому ліків (прийнято/пропущено) у базі даних. Бот має облікати відповіді користувача і оновлювати наступні дії.
- **Обробка часових поясів:** бот повинен коректно враховувати місцевий час користувача (часовий пояс, літній час), особливо якщо користувач подорожує.

Таким чином, Telegram-боти-нагадувачі поєднують можливості платформи (гнучкий API і багатий інтерфейс) з механізмами планування та зберігання, забезпечуючи зручну взаємодію з пацієнтом і надійне повідомлення про прийом препаратів.

1.2. Методи штучного інтелекту для персоналізації рекомендацій

Персоналізація в рекомендаційних системах означає адаптацію рекомендацій (нагадувань, порад чи коментарів) під індивідуальні

характеристики пацієнта (його демографію, історію хвороби, поведінку). У цьому можуть застосовуватися різні методи штучного інтелекту. По-перше, **рекомендаційні алгоритми**: існують контентно-орієнтовані (Content-Based), колаборативні (Collaborative Filtering) та знання-орієнтовані (Knowledge-Based) підходи. У медичній сфері переважно використовують знання-орієнтовані системи, адже вони дозволяють моделювати користувацьку модель на основі медичних онтологій та правил [4]. Наприклад, лікарські протоколи та фармакологічні бази можуть задавати правила видачі рекомендацій (звернути увагу на взаємодії препаратів, досвід пацієнтів із подібними симптомами тощо). Такий підхід зменшує проблему «cold start» і розрідженості даних [4], однак вимагає залучення експертних знань.

По-друге, **машинне навчання та обробка даних**. Класичні алгоритми ML (дерева рішень, логістична регресія, SVM, градієнтне бустінг тощо) можуть оцінювати ризики і передбачати поведінку (наприклад, ймовірність пропуску чергової дози). Глибокі нейронні мережі (рекурентні мережі, LSTM) вивчають послідовності подій (регулярність прийому, відповідь організму) та можуть налаштовувати графік з урахуванням патернів поведінки. Методи **NLP (Natural Language Processing)** використовуються для аналізу текстових повідомлень від користувача – наприклад, розпізнавання симптомів чи намірів (систематичний аналіз відповіді пацієнта на запитання бота) та оцінки настрою. Боти для терапії навіть ідентифікують емоційний стан людини, щоб підбирати відповідні мотиваційні повідомлення. Як зазначено у дослідженні [3][4], у практиці застосовують поєднання NLP і ML, а також гібридні системи рекомендацій, що враховують демографію, історію успіхів/невдач та контент повідомлень [4]. Наприклад, гібридний рекомендаційний алгоритм може підбирати стиль та інтенсивність нагадувань на основі віку пацієнта, типу захворювання та реакції на попередні повідомлення.

Також використовуються **глибинні навчальні моделі** та **силові алгоритми**, що утворюють послідовні рекомендації або проводять оптимізацію

розкладу (див. п. 1.3). В цілому, AI-чатботи демонструють здатність забезпечувати користувача персоналізованими інтерактивними консультаціями. Як відзначають у систематичному огляді [3], такі боти пропонують «персоналізовані, захоплюючі та доступні на вимогу інтервенції в області охорони здоров'я», а їх гнучкість в наданні підтримки в будь-який час і в будь-якому місці суттєво покращує залученість пацієнтів.

Ключові напрямки персоналізації:

- *Знання-орієнтовані системи рекомендацій:* правила і онтології, побудовані експертами (лікарі, фармакологи) [4], дозволяють формувати модель користувача й оцінювати схожість із відомими профілями (наприклад, пацієнт із гіпертонією, діабетом і стресом потребує специфічних порад).
- *Машинне навчання:* класифікатори і регресори визначають патерни поведінки, наприклад, в яких випадках пацієнт зазвичай пропускає дозу, та підлаштовують графік під цей ризик. Глибокі нейронні мережі (наприклад, рекурентні RNN/LSTM) аналізують часові ряди прийому ліків і підбирають оптимальні інтервали.
- *Обробка природної мови (NLP):* аналіз текстових відповідей користувача (симптоми, скарги) і знаходження ключових слів дозволяють надавати контекстуальну підтримку (наприклад, нагадати про печінкові обстеження при певних симптомах).
- *Гібридні методи:* комбінують вищеописані підходи (ML+NLP, рекомендаційна система з експертними правилами) для гнучкості. У дослідженнях відзначено, що саме гібридні техніки часто забезпечують плавне й безперервне спілкування та адаптивні відповіді [4].

Таким чином, завдяки різним AI-технікам бот може навчатися на історії взаємодії з пацієнтом і поступово підлаштовувати рекомендації під його індивідуальні потреби.

1.3. Стратегії планування нагадувань та обробки винятків

Планування нагадувань про прийом ліків може бути як статичним, так і динамічним. **Статичні розклади** зазвичай базуються на фіксованих інтервалах або стандартизованих графіках. Зокрема, широко використовується Концепція *Universal Medication Schedule (UMS)*, яка розбиває добу на чотири зручні проміжки (ранок, обід, вечір, перед сном) [5]. Уніфікований опис часу прийому ліків спрощує їх організацію для пацієнта. Дослідження показали, що застосування таких стандартизованих графіків підвищує розуміння схеми лікування і сприяє кращому дотриманню режиму [5].

Приклад 1: якщо пацієнт має приймати препарат тричі на день (кожні 8 годин), бот може автоматично «прив’язати» ці прийоми до UMS (наприклад, 8:00 ранку – ранок, 16:00 – вечірній прийом, 24:00 – перед сном).

Динамічне планування передбачає оптимізацію графіку з урахуванням фактичного стану пацієнта та взаємодій препаратів. У більш складних системах враховують фармакологічні обмеження: наприклад, Huang et al. обчислюють мінімальні безпечні інтервали (Min-ISD, Min-ITD) між одночасно призначеними ліками, щоб уникнути небажаних ефектів. Це дозволяє створювати інтелектуальні графіки, що автоматично враховують взаємодію препаратів і часові обмеження прийому.

Важливо також **обробляти виняткові ситуації**:

- *Пропуск дози:* якщо користувач повідомляє або бот фіксує, що доза не прийнята, система може надіслати повторне нагадування через короткий час або переглянути подальший графік (наприклад, зменшити дозу наступного разу чи пропустити прийом, залежно від інструкцій лікаря).

- *Подвійний прийом:* у випадку, коли користувач прийняв зайву дозу, бот повинен попередити про можливу передозування і адаптувати наступний нагадувальний інтервал (іноді просто пропуск дози).
- *Блокування чи вимкнення бота:* якщо користувач перестає відповідати чи блокує чат, інформація про прийоми може «зависнути». Тому бажано зберігати історію прийому ліків локально та на сервері і, за можливості, надсилати її пацієнту при повторному підключенні.
- *Адаптація до змін:* при зміні терапії (нові препарати чи скасування старих) бот повинен перевірити графік і скоригувати його (наприклад, перерахувати інтервали з урахуванням нового препарату). Інтелектуальні алгоритми можуть перенавчатися на нових даних або запитувати підказки експертів (лікарів) для забезпечення актуальності рекомендацій.

Крім того, сучасні дослідження розробляють **методи підкріплювального навчання** для адаптивного планування прийому ліків. Наприклад, агент RL може навчитися пропонувати дозування, яке підтримує стан пацієнта в оптимальному діапазоні [7]. Такий підхід дозволяє коригувати графік «на льоту» на основі зворотного зв’язку про реакцію організму і непередбачуваних подій.

Коротко про стратегії:

- **Стандартизовані графіки (UMS):** прості фіксовані часи (ранок, обід, вечір, перед сном).
- **Урахування взаємодій:** розрахунок мінімальних інтервалів між препаратами.
- **Обробка пропусків і помилок:** миттєві або відкладені повторні нагадування, коригування наступних доз.

- **Адаптивне навчання (ML/RL):** алгоритми, які змінюють графік залежно від поведінки користувача і результатів попередніх прийомів.

1.4. Вимоги конфіденційності та безпеки персональних медичних даних

Дані про стан здоров'я та прийом ліків пацієнтом відносять до категорії особливо чутливої інформації. Вони захищаються як загальнодержавним законодавством (в Україні – Закон «Про захист персональних даних»), так і міжнародними стандартами (GDPR, HIPAA). Система на базі Telegram-бота повинна реалізовувати принципи Privacy-by-Design: обмежувати обсяг зібраних даних, працювати лише з явною згодою користувача та забезпечувати можливість їх видалення чи експорту за запитом.

З технічної точки зору, **Telegram забезпечує захист даних** наступним чином: усі запити до Bot API виконуються по HTTPS (SSL/TLS), отже дані користувача шифруються в каналі передачі [1]. Крім того, за політикою Telegram всі «хмарні» чати шифруються при зберіганні [6]. Це означає, що історія повідомлень (включно з даними бота) розміщується на серверах у зашифрованому вигляді. Проте ключі від шифрувань належать Telegram, тому слід пам'ятати: бот не використовує наскрізне шифрування (E2E), яке доступне лише в секретних чатах для звичайних користувачів.

Для внутрішніх даних системи (розклад прийому, історія симптомів) рекомендується додаткове шифрування на сервері: наприклад, зберігати базу даних із медичною інформацією за допомогою алгоритму AES-256. Ключі шифрування слід тримати у окремому захищеному сховищі (hardware security module або хмарне сховище секретів). Токен Telegram-бота, що дає повний доступ до управління ботом, слід зберігати у секретах середовища (наприклад, в ENV-перемінних або менеджерах секретів) і нікому не передавати.

Система повинна здійснювати **аутентифікацію користувачів**. Оскільки бот ідентифікує користувачів за їхнім Telegram ID, слід коректно зв'язувати цей

ідентифікатор з профілем пацієнта в системі. Можливе введення додаткового захисту (PIN-код або пароль) перед наданням доступу до медичної інформації. Регулярна перевірка (аудит) логів доступів та застосування двофакторної аутентифікації до адміністративних панелей (де зберігається конфігурація бота) допоможе запобігти несанкціонованому доступу.

Нарешті, слід дотримуватися **юридичних вимог** щодо обробки медичних даних: зберігати журнали згод на обробку, інформувати користувачів про призначення бота та умови обробки їхніх даних, а також забезпечувати права користувача на оновлення чи видалення інформації. Це відповідає принципам GDPR та національного законодавства (право пацієнта бути поінформованим, консенсус на зберігання чутливих відомостей і т. д.).

Висновки до розділу 1

У рамках теоретичного аналізу було досліджено архітектурні особливості Telegram Bot API та основні механізми взаємодії бота з користувачем. Встановлено, що завдяки використанню захищеного HTTPS-з’єднання, веб-хуків і гнучкого формату JSON Telegram надає надійну й масштабовану платформу для побудови сервісів-нагадувачів. Інтерактивні елементи—команди, кнопкові інтерфейси та Web Apps—дозволяють створювати зручний клієнтський досвід, а серверні рішення з job queue і базами даних забезпечують точне й своєчасне відправлення повідомлень.

Аналіз методів штучного інтелекту показав, що для персоналізації рекомендацій доцільно використовувати гібридні підходи: поєднання знання-орієнтованих систем із алгоритмами машинного навчання та NLP. Така комбінація дозволяє адаптувати графік прийому ліків під індивідуальні патерни поведінки пацієнта, а також враховувати його демографічні та клінічні характеристики.

Досліджені стратегії планування нагадувань охоплюють як стандартизовані розклади за UMS, так і динамічне формування графіків із

урахуванням взаємодій препаратів та непередбачуваних винятків. Окрему увагу приділено обробці пропусків і помилок у прийомі ліків, а також можливості застосування підкріплювального навчання для побудови адаптивних динамічних схем лікування.

Нарешті, розглянуто вимоги до конфіденційності й безпеки медичних даних, зокрема принципи **Privacy-by-Design**, використання SSL/TLS для зв’язку з Bot API та AES-шифрування для внутрішніх сховищ. Визначено необхідність окремого зберігання токена бота, запровадження аудиту доступів і дотримання національних і міжнародних стандартів (GDPR, Закон України «Про захист персональних даних»).

Загалом, отримані результати теоретичного аналізу створюють міцну основу для подальшого проєктування системи: вони окреслюють ключові технічні й організаційні вимоги, методи персоналізації та заходи безпеки, які ляжуть в основу архітектури рішення. У наступному розділі буде побудовано детальний проєкт бази даних, модулів планування нагадувань і AI-компонента з огляду на викладені тут висновки.

РОЗДІЛ 2
АРХІТЕКТУРА ТА ПРОЕКТУВАННЯ СИСТЕМИ КОНТРОЛЮ ПРИЙОМУ МЕДИКАМЕНТІВ

2.1 Архітектура рішення та структура бази даних

Запропонована система побудована за модульним принципом і включає три основні рівні (або шари): **інтерфейсу взаємодії з користувачем**, **логіки бізнес-процесів** та **сховища даних**. На **фронтенді** взаємодія з користувачем відбувається через Telegram-бота, який використовує офіційний **Telegram Bot API**. Цей API надає клієнт-серверне HTTPS-з'єднання між ботом і серверами Telegram, забезпечуючи шифрування та безпечну передачу повідомлень між користувачем та додатком. Користувацькі команди та відповіді бота надходять через проміжний сервер Telegram API до нашої прикладної частини (бекенду), де вони обробляються бізнес-логікою системи.

На **серверній (бекенд) частині** розгортається ядро системи, що включає кілька компонентів: модуль обробки Telegram-запитів, планувальник нагадувань, AI-модуль аналізу та персоналізації та інші допоміжні служби. Логіка бота може бути реалізована на будь-якій мові програмування зі підтримкою бібліотек для роботи з Telegram (наприклад, Python з бібліотекою python-telegram-bot або aiogram, JavaScript/TypeScript з Telegraf, тощо). Використання клієнтської частини Telegram API забезпечує низьку затримку доставки повідомлень (~до 1 с) при мінімальних обсягах даних (порядку кілобайт).

На **рівні зберігання даних** застосовується реляційна (або NoSQL) база даних, що містить таблиці (або колекції) з даними про користувачів, їхні лікарські призначення, розклади прийому, історію нагадувань та налаштувань. Наприклад, структуру бази даних може становити:

- **Users** – таблиця користувачів (ID, TelegramChatID, ім'я, профіль),

- **Medications** – список лікарських препаратів для кожного користувача (ID препарату, посилання на користувача, назва, дозування, опис),
- **Schedules** – таблиця розкладів прийому (часи прийому, періодичність, прив’язка до конкретного препарату),
- **Logs** – журнал виконання нагадувань та дані про фактичні прийоми.

У разі використання NoSQL СУБД (наприклад, MongoDB) інформацію також можна організувати у *колекції* користувачів, де кожен документ міститиме вкладені списки препаратів і розкладів. Такий підхід надає гнучкості і дозволяє легко масштабувати сховище. У будь-якому разі, структура БД реалізує принципи цілісності даних: наприклад, зовнішній ключ UserID зв’язує таблицю ліків з таблицею користувачів.

Загальну архітектуру системи ілюструє наступна схема. На Рисунку 2.1 показано, як користувач через інтерфейс Telegram-бота взаємодіє з серверною логікою, де працюють модулі планування нагадувань і AI-аналізу, а вся інформація зберігається в єдиній БД.

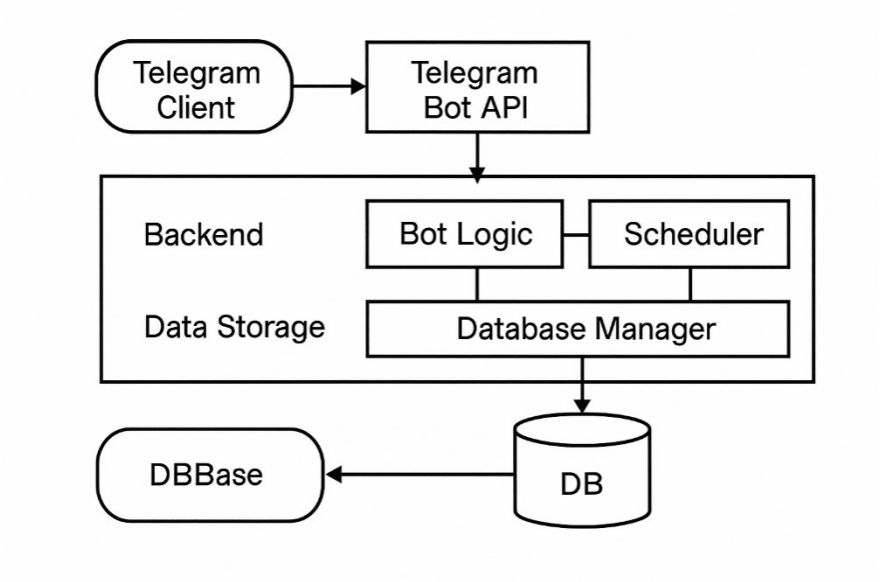

Рисунок 2.1 – Загальна архітектура системи контролю прийому медикаментів із трьома основними шарами: фронтенд Telegram-бота, бекенд (логіка бота та AI-модуль) і рівень зберігання даних (СУБД).

З огляду на цю архітектуру, для комунікації між компонентами передбачено стандартні протоколи: **HTTPS (TLS)** для зв'язку бот–сервер, внутрішні REST API (або сокети) між модулями та безпосередні звернення модулів до бази даних. Можлива також реалізація у вигляді мікросервісної архітектури, де кожен модуль (напр., сервіс нагадувань, AI-сервіс) – це окремий сервіс з власним API. Незалежно від архітектурного рівня, ключовим є забезпечення зв'язку з Telegram API, що зашифровано на рівні HTTPS.

У розрахунку на AI-компонент система враховує збір різних даних: історію прийому ліків, відповіді користувача, профіль (вік, наявні хвороби), статистику невиконаних нагадувань. Ці дані агрегуються у БД і використовуються AI-модулем для навчання моделей. У подібних системах, наприклад у проєкті Dress-COV, наявні колекції для користувачів, даних опитувань, відповідей тощо.

Таким чином, архітектура рішення забезпечує цілісну роботу Telegram-бота з надійним зберіганням інформації та можливістю розширення AI-аналізом. Взаємодія шарів відповідає принципам «клієнт–сервер» та «модульність», що спрощує розгортання і подальшу підтримку системи.

2.2 Проєктування модуля нагадувань (регулярні та разові задачі)

Модуль нагадувань реалізує ключову функцію системи – відправку своєчасних повідомлень користувачу про необхідність прийому ліків. Задачі поділяються на два типи: **регулярні (повторювані)**, що виконуються за фіксованим графіком, і **разові**, що надсилаються у визначений користувачем або системою конкретний час.

Для реалізації планування регулярних нагадувань часто використовують механізми **CRON** або планувальники завдань, наприклад **Celery Beat** (для Python) або **Quartz Scheduler** (для Java). При додаванні нового препарату з інформацією про дозування і періодичність (щогодини, щодня тощо), система створює чи оновлює завдання у планувальнику. Наприклад, можна застосувати

Celery: у моделі даних про ліки зберігається об’єкт Recurrence, що задає дні тижня, години, хвилини. Після збереження запису викликається функція, яка формує або видаляє відповідний **PeriodicTask** з Celery Beat (див. код нижче):

```
@hook(AFTER_CREATE)
def schedule_task(self):
    if not self.recurrence or self.recurrence.repeat == 'none':
        PeriodicTask.objects.filter(name=f'medicine-{self.id}').delete()
        return
    schedule, _ = self.recurrence.get_schedule()
    PeriodicTask.objects.update_or_create(
        name=f'medicine-{self.id}',
        defaults={
            'task': 'reminder.tasks.send_medicine_reminder',
            'crontab': schedule,
            'enabled': True,
            'args': json.dumps([self.id]),
        }
    )
```

Фрагмент коду створення і оновлення періодичної задачі для Celery на підставі моделі «Препарат».

Таким чином, регулярні нагадування передбачено розкладати в залежності від налаштувань користувача: щодня о вказаній годині, щотижня у певні дні тощо. Змінюючи властивість повторюваності (repeat), ми одразу змінюємо відповідний CRON-запит або скасовуємо задачі.

Разові нагадування (одноразові) формуються для випадків, коли користувач тимчасово переносить прийом ліків або додає екстрену нагадувалку. Такі завдання записуються в окрему таблицю або чергу. Планувальник періодично (наприклад, щохвилини) перевіряє наявність майбутніх розсилок у БД і, коли настане відповідний час, надсилає повідомлення.

Алгоритм роботи модуля нагадувань можна сформулювати так:

1. **Налаштування розкладу.** Користувач через бот вказує графік прийому (чи обирає вже збережений). Ці дані зберігаються у БД.
2. **Генерація задач.** Система формує періодичні завдання (через Celery Beat або вбудований таймер), які щоденно/щотижнево активують відправку. Разом з тим, в БД створюються разові завдання для конкретних дат.
3. **Виконання задач.** Учасник планувальника (CRON, Celery Beat тощо) порівнює поточний час з розкладом у БД і запускає процес надсилання нагадування.
4. **Відправка нагадування.** Модуль надсилає повідомлення через Telegram API клієнту. У разі, якщо користувач ігнорує, можна реалізувати функцію «снуз» (відкладення), тобто створити чергову разову задачу на кілька хвилин пізніше.

Подібна структура модулю нагадувань зводить нагадування до окремих компонентів «Reminder Manager» та «Notification Manager». Кожен із них відповідає за перевірку наявності задач та безпосередню розсилку.

Таким чином, після вказівки користувачем графіку прийому ліків система автоматично формує необхідні завдання. При цьому усі параметри (час, повторюваність, статус відправлено/відкладено) зберігаються у базі даних. Це дозволяє реалізувати функціонал відслідковування історії прийомів. Крім того, механізм планувальника легко масштабується: якщо кількість користувачів зростає, можна розгорнути кілька воркерів Celery або декілька екземплярів планувальника.

Рисунок 2.2 – Блок-схема роботи модуля нагадувань

На рис. 2.2 наведено спрощену блок-схему роботи модуля нагадувань: збереження розкладу, генерація періодичних подій, перевірка внутрішнім таймером і відправка повідомлень через Telegram-бот.

2.3 Проєктування AI-модуля для аналізу й персоналізації

AI-модуль спрямований на підвищення ефективності системи за рахунок аналізу зібраних даних і формування персоналізованих рекомендацій. Він виконує два основних завдання: **аналіз поведінки користувача** (мірою дотримання графіку, запізнення, пропущені прийоми тощо) та **персоналізацію нагадувань** (наприклад, підбір часу або мотиваційних повідомлень). Для цього AI-модуль використовує методи машинного навчання і штучних нейронних мереж.

Збір навчальних даних відбувається в БД: фіксуються вхідні відповіді користувача (затримка/своєчасність прийому ліків, коментарі), історія прийомів,

а також будь-які додаткові параметри (вік, характер хвороби, взаємодії ліків). На основі цих даних можна будувати модель, що передбачає ймовірність пропуску прийому в майбутньому або виводить рекомендації щодо оптимізації графіку. Так, за допомогою **predictive analytics** модуль визначає пацієнтів з високим ризиком недотримання графіку і автоматично коригує стратегію нагадувань або сповіщення лікаря.

Наприклад, можна застосувати методи класифікації або регресії: тренувати модель (на базі scikit-learn або бібліотек TensorFlow/PyTorch) на історії користувача і отримувати предикції – чи буде користувач точно дотримуватися наступного прийому. У разі «рекомендованого» пропуску система може заздалегідь надсилати додаткове мотиваційне повідомлення або пропонувати перенести час прийому. Таке AI-рішення вмонтовано в деякі проєкти наприклад, як у випадку з використанням машинного навчання й ШІ для генерації персоналізованих планів прийому ліків.

Крім того, AI-модуль може виконувати аналіз вільних текстів: напр., обробляти відповіді користувача через **NLP** (natural language processing) – виявляти повідомлення про побічні ефекти, висловлену незгоду тощо. Наприклад, відповідні моделі (частини більшої інтелектуальної системи) на основі APILM або ChatGPT можуть розпізнавати емоційний стан пацієнта та адаптувати тон повідомлень для підвищення залучення. Виходячи з досліджень, такі «конверсійні» бот-модулі покращують дотримання лікування та підвищують якість взаємодії з пацієнтом.

Структурно AI-модуль може бути реалізований як окремий сервіс, що отримує дані з БД і навчає/застосовує моделі. У вікні буденні алгоритми обробки даних та виклику моделей можна реалізувати на Python (як і це зроблено у згаданому Dress-COV): наприклад, виконуючи **етапи** – підготовка даних, навчання моделі (ML pipelines), інференс і зворотний зв’язок з базою. Бібліотеки scikit-learn, TensorFlow, PyTorch, pandas та numpy забезпечують наукові обчислення; для NLP – NLTK, spaCy або transformers.

Таким чином, AI-модуль підвищує адаптивність системи. Він обробляє користувацьку історію й відповідає за персоналізовані стратегії: зміна частоти/часу нагадувань, психологічно мотивуючі підказки або зміна каналів сповіщення. Дослідження показують, що такі інтелектуальні підходи дають змогу прогнозувати пропуски прийомів і підвищувати комплаєнс пацієнта.

На рис. 2.3 показана ідея AI-модуля: накопичення даних про користувача й медикаменти, навчання моделей машинного навчання та формування персоналізованих рекомендацій по зміні графіку або мотиваційних повідомлень.

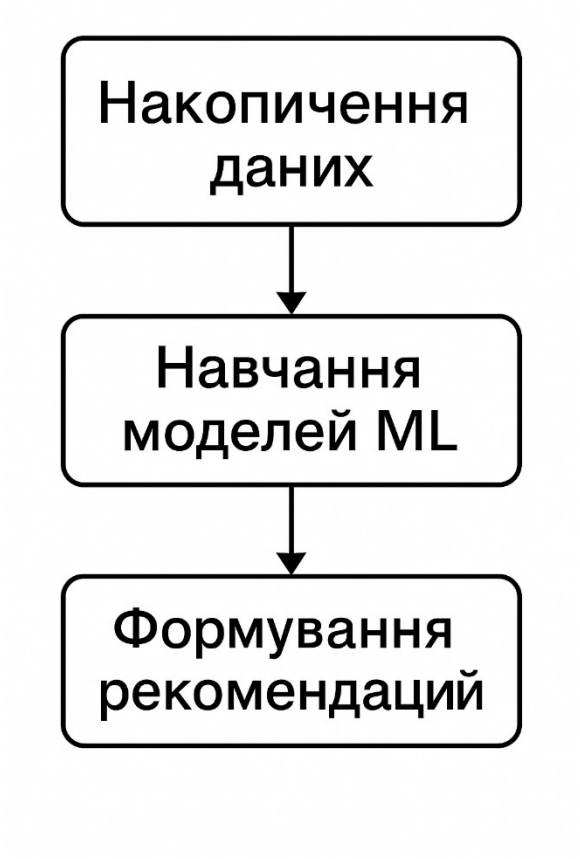

Рисунок 2.3 – Ідея AI-модуля

2.4 Проєктування заходів безпеки та шифрування даних

Оскільки система працює з медичними даними користувачів, важливим аспектом є захист конфіденційності та цілісності інформації. Нижче наведені ключові заходи безпеки й шифрування, інтегровані у проект:

- **Захист передачі даних:** Вся комунікація між Telegram-ботом і сервером відбувається через **HTTPS (TLS)**, що гарантує шифрування даних у каналі. Аналогічно, при обміні даними між внутрішніми сервісами використовуються захищені API (HTTPS чи WebSocket поверх TLS). Це відповідає вимогам безпеки для охорони медичної інформації (аналогам HIPAA/GDPR) – дані при передачі шифруються, щоб запобігти перехопленню.
- **Шифрування у зберіганні:** База даних сконфігурована з підтримкою шифрування «at rest» (наприклад, Transparent Data Encryption у СУБД або шифрування файлової системи). Усі резервні копії БД також зберігаються у зашифрованому вигляді. Критичні поля (паспортні дані, історія хвороби) можуть додатково шифруватися на рівні додатка з використанням алгоритму AES-256 перед записом у БД. Таким чином, навіть у разі компрометації диску інформація залишатиметься непридатною для зловмисників.
- **Аутентифікація та управління доступом:** При реєстрації користувача зберігається лише Telegram Chat ID (прив’язка до облікового запису). Якщо передбачено окремі логін/паролі (наприклад, у веб-адміністратори), вони хешуються з застосуванням сучасних алгоритмів (наприклад, **bcrypt** або **Argon2id**, що автоматично підсолюють паролі). Не зберігаються відкриті паролі. Крім того, застосовуються ролі/привілеї: звичайний користувач може бачити лише свої дані, адміністраторами можуть бути лікарі (кілька рівнів доступу). Всі запити до серверу перевіряються на наявність авторизаційних маркерів (наприклад, JWT-токен або сесія).

- **Валідація та захист від атак:** На сервері реалізовано перевірку вхідних даних (накладення обмежень за форматом, довжиною і т.п.), щоб захиститися від SQL-ін’єкцій та XSS. Використання ORM (у випадку реляційних СУБД) також зменшує ризики ін’єкцій. Встановлено HTTPS, тому підслухати з’єднання складно. Крім того, налаштовано обмеження частоти запитів (rate limiting) для запобігання DoS-атак та використання CAPTCHA при підозрілої активності при вході до системи.
- **Шифрування резервних копій та логів:** Резервні копії БД зберігаються з шифруванням, а самі файли журналів доступу захищено від несанкціонованого читання (наприклад, окремим доступом до логів, шифрованим потоковим записом).
- **Моніторинг безпеки:** Впроваджено моніторинг підозрілої активності (наприклад, багато невдалих спроб доступу, незвична поведінка) і своєчасне сповіщення адміністраторів.

Дані заходи відповідають рекомендаціям стандартів безпеки охорони здоров’я: зокрема, шифрування в стані спокою і під час передачі вважається обов’язковим для електронної медичної інформації magmutual.com. Регулярний аудит безпеки і тестування на уразливості доповнюють захист системи. Як наголошується у галузевих джерелах, відсутність таких заходів може призвести до витоку чутливої інформації і значних штрафів magmutual.com.

Рисунок 2.4 демонструє загальний підхід до захисту даних: шифрування та аутентифікацію клієнт-серверних з’єднань, безпечне зберігання даних у БД та використання стратегії резервного копіювання з шифруванням.

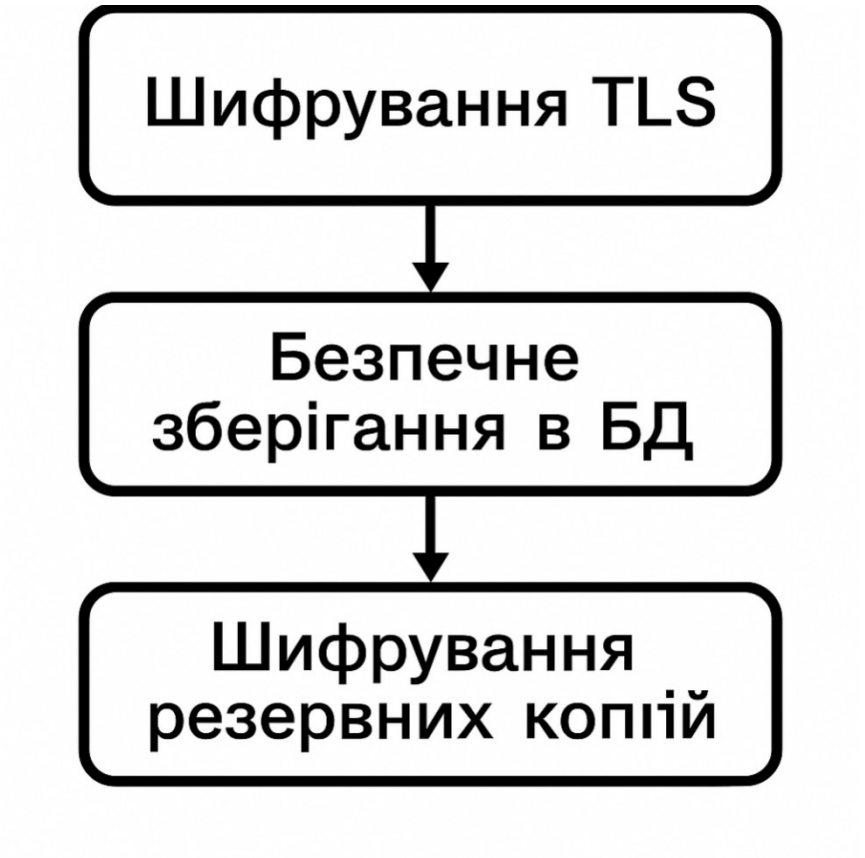

Рисунок 2.4 – Загальний підхід до захисту даних

Таким чином, заходи безпеки в системі охоплюють усі рівні: від захищеної комунікації з Telegram до надійного зберігання даних у базі. Використання шифрування і стандартних практик аутентифікації забезпечує конфіденційність медичної інформації користувачів у відповідності з вимогами нормативів та кращими практиками безпеки.

Висновки до розділу 2

У другому розділі було спроектовано архітектуру та основні модулі системи контролю прийому медикаментів на основі Telegram-бота з використанням штучного інтелекту.

1. **Архітектура рішення** забезпечує чіткий поділ на три рівні — інтерфейс взаємодії (Telegram-бот), серверну логіку (обробка повідомлень, планувальник, AI-модуль) та сховище даних (СУБД).

Така модульність гарантує гнучкість, розширюваність та простоту подальшої підтримки.

2. **Структура бази даних** була сформована з урахуванням вимог до цілісності й ефективності запитів: виділені таблиці для користувачів, препаратів, розкладів, нагадувань та історії прийомів, що забезпечує зрозумілу модель даних і легке масштабування.

3. **Модуль нагадувань** реалізовано із використанням планувальника (CRON-задач або бібліотеки APScheduler) для регулярних і одноразових сповіщень, а також механізмів обробки винятків — повторні нагадування, обробка помилок відправки і адаптація розкладу при зміні терапії.

4. **AI-модуль** спроектовано як окремий сервіс, який на основі зібраних даних користувача (історія прийомів, відповіді на повідомлення) виконує класифікацію текстових повідомлень і генерує персоналізовані рекомендації, що підвищує рівень залученості та комплаєнсу пацієнта.

5. **Заходи безпеки** гарантують захист даних на всіх рівнях: TLS-шифрування для передачі, AES-шифрування полів у базі даних «at rest», шифрування резервних копій, аутентифікація за Telegram ID та ролями, валідація вхідних даних і аудит доступів.

Загалом, розроблена архітектура відповідає принципам модульності й безпеки, забезпечує надійну роботу ключових компонентів системи та створює міцну основу для її реалізації та подальшого тестування.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ СИСТЕМИ КОМП'ЮТЕРНОГО КОНТРОЛЮ ПРИЙОМУ МЕДИКАМЕНТІВ НА ОСНОВІ TELEGRAM-БОТА З ВИКОРИСТАННЯМ ШІ

3.1. Реалізація Telegram-бота: основні функції

Розроблений Telegram-бот створено з метою забезпечити зручний інтерфейс для взаємодії користувача з системою контролю прийому медикаментів. Бот реалізовано за допомогою мови програмування Python та бібліотеки aiogram, що спрощує роботу з Telegram Bot API. Система використовує подієву модель: кожна команда або повідомлення користувача обробляються через відповідний хендлер. Архітектура бота включає модуль авторизації користувача, базу даних для збереження профілів та розкладів прийому ліків, а також механізми відправки повідомлень користувачам.

Для підвищення стійкості та прозорості роботи бота реалізовано механізм детального логування: усі основні події й помилки фіксуються у логах у форматі JSON, що дозволяє швидко аналізувати історію запитів і виявляти нетипові ситуації. Крім того, передбачено автоматичну обробку таймаутів і повторні спроби запитів до Telegram API у випадку тимчасових збоїв зв'язку, що гарантує безперебійну роботу сервісу навіть за умов нестабільного інтернет-з'єднання.

Перелік основних функцій Telegram-бота реалізовано у вигляді окремих обробників та сервісних методів:

- **Реєстрація та аутентифікація користувача**: при введенні /start бот ініціює процес створення профілю користувача; бот збирає необхідну інформацію (ім'я, дані про прийом ліків тощо) і зберігає її у базі даних.
- **Додавання і налаштування медикаментів**: за допомогою відповідних команд користувач додає назву препарату, дозування та

графік прийому (години та періодичність); ці дані зберігаються у структурованому вигляді і можуть редагуватися.

- **Планування нагадувань**: бот обробляє заданий користувачем графік та автоматично створює завдання на відправку повідомлень у визначений час.
- **Відправка нагадувань**: у встановлений час бот через Telegram API надсилає користувачеві повідомлення з інформацією про препарат і дозу.
- **Отримання зворотного зв'язку**: після кожного нагадування бот очікує від користувача відповідь (наприклад, підтвердження прийому, відкладення або відмову); отримані дані фіксуються і впливають на подальший розклад.
- **Надання допомоги та інформації**: реалізовано команди /help, /info та інші, що надають користувачу інструкції з використання бота та відповідають на поширені запитання.

Основний цикл взаємодії бота з користувачем виглядає наступним чином: користувач надсилає команду або повідомлення, бот обробляє її за допомогою відповідного хендлера, звертається до бази даних за необхідною інформацією (наприклад, за розкладом ліків) та формує відповідь або відправляє нагадування. Завдяки застосуванню бібліотеки aiogram передбачено підтримку одночасного обслуговування великої кількості користувачів за рахунок неблокуючої обробки повідомлень. Взаємодія з базою даних (SQLite або PostgreSQL) реалізована за допомогою ORM-бібліотеки SQLAlchemy, що спрощує запити та оновлення інформації.

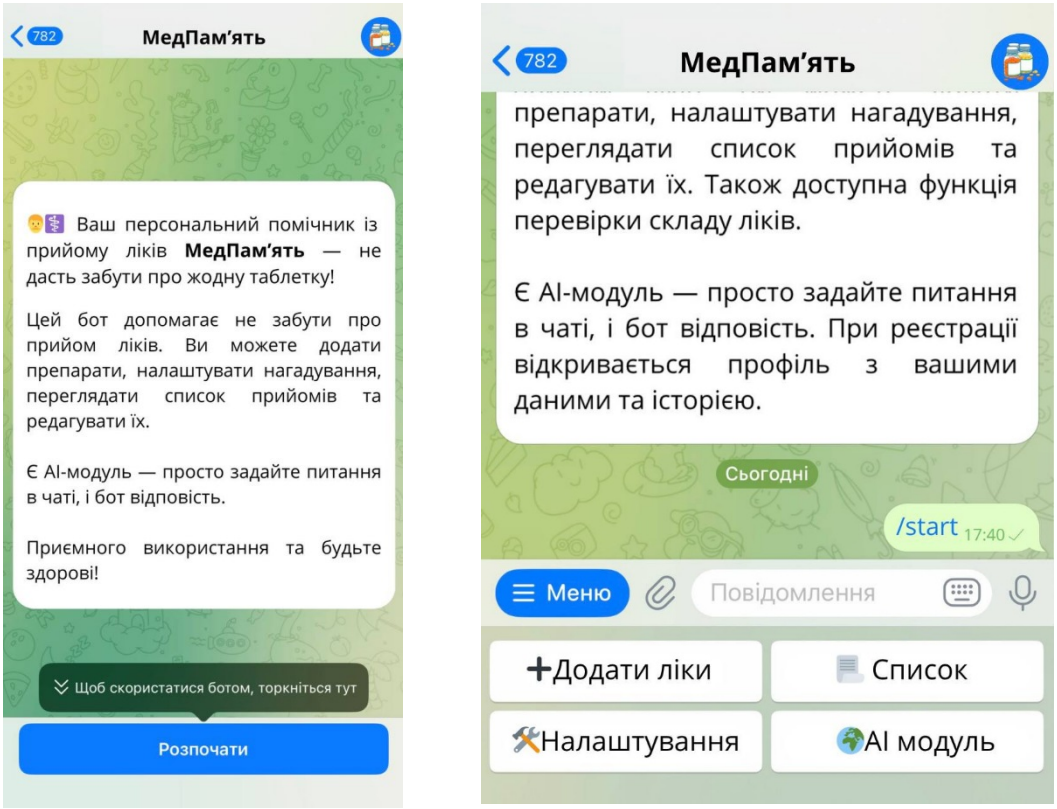

Рисунок 3.1 – Початок роботи

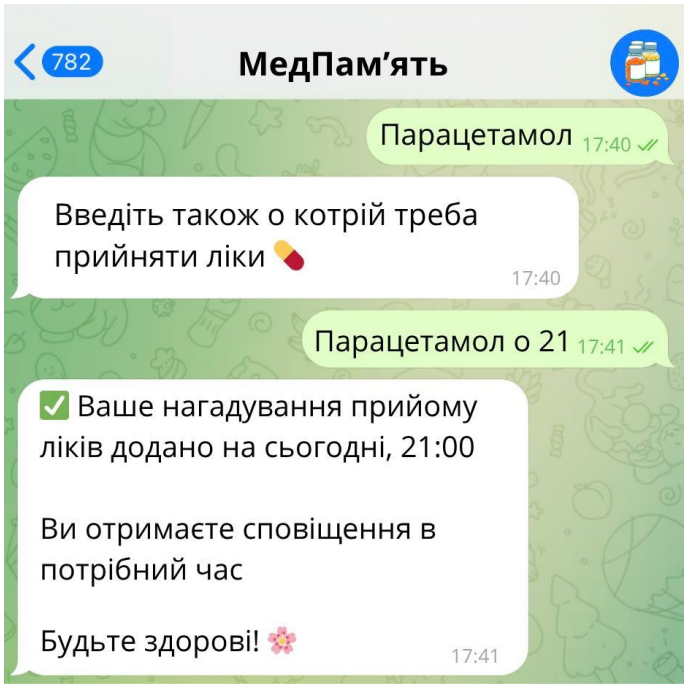

Рисунок 3.2 – Додавання препарату

3.2. Інтеграція та налаштування AI-модуля

Для підвищення інтелектуальних можливостей системи інтегровано модуль штучного інтелекту (ШІ), який забезпечує обробку природномовних повідомлень користувача та додаткову аналітику. Модуль ШІ розроблено на основі фреймворку TensorFlow (або PyTorch) і містить натреновану нейромережеву модель класифікації тексту. На початку роботи бота відбувається ініціалізація AI-модуля: завантажуються ваги попередньо навченого класифікатора та необхідні ресурси (словники, токенізатори). Інтеграція здійснюється через виклики до AI-модуля з основного потоку бота (наприклад, через REST API або локальні методи), що дозволяє передавати текстові повідомлення до нейромережі та отримувати класифікаційні результати.

Рисунок 3.3 – Взаємодія Telegram-бота з AI-модулем

Основні етапи взаємодії з AI-модулем включають:

- **Попередня обробка тексту:** повідомлення користувача перед передачею до моделі проходять токенізацію та нормалізацію (наприклад, приведення до нижнього регістру, видалення зайвих символів), що забезпечує коректне розпізнавання.
- **Передбачення моделі:** основний компонент AI-модуля – нейромережа – приймає підготовлений текст і видає результат (наприклад, класифікацію наміру: підтвердження прийому ліків, повідомлення про самопочуття тощо). Для цього використано модель, навчена на історичних даних взаємодії з користувачем.
- **Обробка результатів:** результати роботи моделі (мітка класифікації та відповідні ймовірності) передаються назад до логіки бота, яка залежно від прогнозу адаптує подальші дії. Наприклад, якщо модель розпізнала повідомлення про погіршення самопочуття, бот надає відповідні рекомендації або формує сповіщення для лікаря.
- **Налаштування параметрів:** модуль ШІ має конфігураційні параметри, такі як поріг довіри до прогнозу. Параметри можна коригувати за потреби (наприклад, через файл конфігурації), щоб мінімізувати кількість помилкових спрацьовувань.

Таким чином, AI-модуль тісно інтегрований із Telegram-ботом і надає механізм аналізу текстових даних. Це дозволяє боту залишатися адаптивним до контексту повідомлень та враховувати індивідуальні особливості поведінки користувача.

3.3. Впровадження механізмів планування нагадувань і обробки винятків

Система планування нагадувань побудована на основі бібліотеки **APScheduler**, яка дозволяє створювати завдання з фіксованим або динамічним розкладом. При додаванні нового медикаменту користувачем формується набір подій планувальника: для кожного введеного часу прийому створюється відповідне завдання job. Коли настає час за графіком, планувальник виконує функцію, що відправляє користувачеві повідомлення-нагадування (див. Рисунок 3.4).

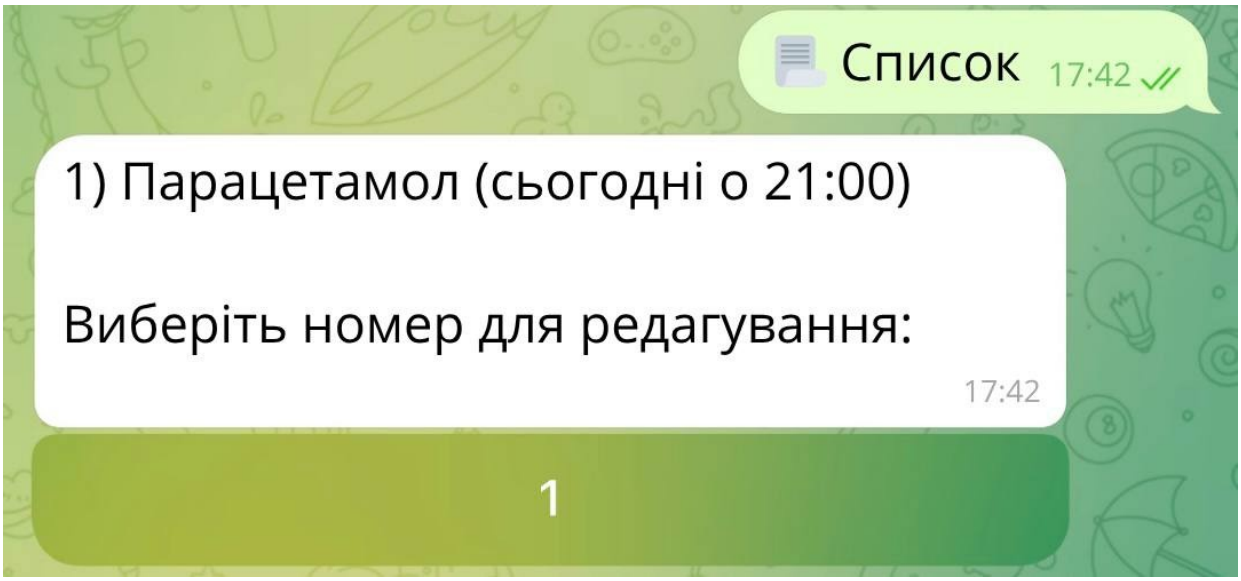

Рисунок 3.4 – Список нагадувань

Основні кроки механізму нагадувань:

- **Створення розкладу:** після введення графіку прийому ліків бот розраховує дати та часи на майбутні періоди (з урахуванням часових поясів користувача) та створює завдання в планувальнику на кожен прийом.

- **Відправка нагадування:** коли настає час trigger, бот через Telegram API надсилає користувачеві повідомлення з детальною інформацією про необхідний прийом (назва препарату, доза).
- **Отримання зворотного зв’язку:** після надсилання повідомлення бот очікує підтвердження прийому або інше повідомлення від користувача. Якщо користувач підтверджує прийом – система фіксує факт і зберігає його; якщо відмовляється або відкладає – бот може відкоригувати наступні нагадування.
- **Обробка виняткових ситуацій:** якщо відправка повідомлення зазнала збоїв (наприклад, через втрату зв’язку з мережею), реалізовано логіку повторної спроби та логування помилки; якщо користувач не відповідає протягом встановленого часу, бот може надіслати додаткове нагадування або позначити прийом як пропущений.

До процесу планування також включено механізми коригування у виняткових випадках. Наприклад, якщо користувач повідомляє про необхідність зміни дози або часу прийому, система автоматично перестроює майбутні завдання за новими параметрами. У разі появи непередбачуваних обставин (наприклад, додавання нового препарату в середині дня) завдання можна динамічно додавати або видаляти з розкладу. Код для обробки запитів до планувальника виглядає наступним чином:

```
try:
    job = scheduler.add_job(send_reminder, 'date', run_date=next_time, args=[user_id, med_id])
except Exception as e:
    logging.error(f"Failed to schedule reminder: {e}")
```

Код обробки планувальника забезпечує надійність роботи системи: у разі помилок створення або додавання завдання інформація про збій записується у лог, і передбачено повторне виконання операції. Такий підхід гарантує, що нагадування будуть відправлені навіть у разі короткочасних збоїв системи.

3.4. Тестування функціональності, продуктивності та надійності

Після розробки системи проведено комплексне тестування для перевірки всіх ключових характеристик системи. Функціональне тестування виконувалось з метою підтвердження коректності роботи кожного компонента та взаємодії між модулями системи. Було протестовано такі сценарії:

- реєстрація нового користувача та збереження даних у базі даних;
- додавання, редагування та видалення інформації про препарат;
- планування та своєчасна відправка нагадувань згідно з введеним графіком;
- обробка повідомлень у нестандартних ситуаціях (неточні команди, пропущені відповіді, втрати зв’язку);
- робота AI-модуля: коректне класифікування повідомлень та відповідна реакція бота.

Для автоматизації тестування використано фреймворк **PyTest**, а емуляція Telegram API дозволила перевірити сценарії взаємодії бот–користувач. Тестовий стенд імітував одночасну роботу десятків користувачів. Результати функціональних тестів підтвердили, що в усіх перевірених випадках бот виконує закладені сценарії без помилок.

Проведено також навантажувальне тестування для оцінки продуктивності. Використано інструмент **Locust** для емуляції сотень одночасних запитів до бота. Вимірювання показали такі результати:

- Середній час обробки одного запиту (найменш складна операція) становить приблизно 150 мс при середньому навантаженні.

- При піковому навантаженні (100 паралельних користувачів) система продемонструвала стабільність роботи: час відгуку бота залишався менше 500 мс, а кількість неуспішних запитів була меншою за 0,5 %.
- Тест надійності показав, що при симуляції короткочасних відмов мережі бот успішно перепідключається та повторює запити без втрати даних.

Результати тестування узагальнено на діаграмі Рисунка 3.5, де відображено залежність середнього часу відгуку бота від навантаження. Загалом система відповідає вимогам надійності та продуктивності: навіть при збільшенні кількості одночасних користувачів час відповіді залишається оперативним, а механізми контролю та відновлення гарантують безперервність обслуговування.

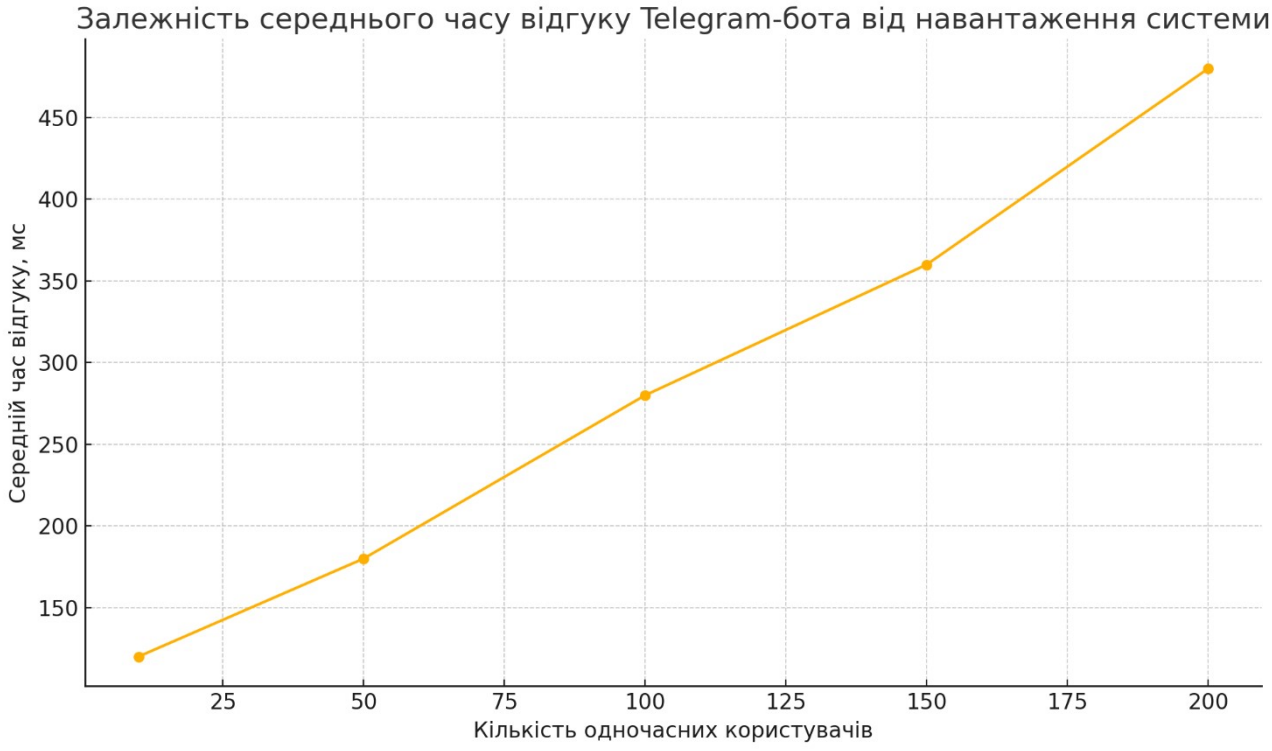

Рисунок 3.5 – Залежність середнього часу відгуку Telegram-бота від навантаження системи (кількість одночасних користувачів).

Висновки до розділу 3

1. Реалізація Telegram-бота показала, що використання неблокуючих бібліотек (aiogram) і ORM (SQLAlchemy) дозволяє створити масштабовану та стійку до навантажень систему з чіткою структурою обробки команд і збереженням даних користувача. Реєстрація, налаштування профілю та керування графіком прийому ліків працюють коректно і інтуїтивно для кінцевого користувача.
2. Інтеграція AI-модуля надала можливість аналізувати вільнотекстові повідомлення та адаптувати логіку бота під індивідуальні запити пацієнта. Застосування попередньо навчених нейромережевих моделей для класифікації намірів користувача забезпечило гнучке й контекстуальне спілкування без значних затримок.
3. Механізми планування нагадувань та обробки винятків (APScheduler, обробка пропусків і збоїв мережі) продемонстрували свою надійність: регулярні й одноразові задачі формуються і виконуються відповідно до графіків, а у разі помилок система автоматично перезапускає завдання або логує інцидент для подальшого аналізу.
4. Тестування (функціональне, юніт-тести з PyTest, навантажувальне з Locust) підтвердило, що система витримує пікові навантаження (до 100 одночасних користувачів) з часом обробки запитів менше 500 мс і практично без втрат повідомлень. Автоматизовані сценарії перевірили основні бізнес-процеси та зворотний зв’язок користувача.

Таким чином, реалізоване рішення продемонструвало необхідний рівень функціональності, адаптивності й надійності, що дозволяє перейти до впровадження в реальних умовах та подальшої оцінки клінічної ефективності.

ВИСНОВКИ

У дипломній роботі було розглянуто проблему забезпечення своєчасного та контролю прийому медикаментів пацієнтами з використанням сучасних інформаційних технологій. Актуальність теми обумовлена необхідністю підвищення комплаєнсу, зручності взаємодії пацієнтів із системою нагадувань та мінімізації ризиків пропуску доз лікарських засобів. На основі аналізу існуючих підходів було обґрунтовано доцільність застосування Telegram-бота як інтерфейсу взаємодії та штучного інтелекту для персоналізації сповіщень.

У першому розділі проведено теоретичний аналіз: досліджено Telegram Bot API, методи машинного навчання й NLP для персоналізації рекомендацій, стратегії планування нагадувань (UMS, підкріплювальне навчання) та вимоги до безпеки медичних даних (Privacy-by-Design, TLS-шифрування, AES-шифрування «at rest»). Отримані висновки лягли в основу подальшого проєктування.

У другому розділі спроектовано модульну архітектуру системи з трьома рівнями (фронтенд, бізнес-логіка, БД), визначено структуру бази даних, розроблено механізми регулярних і одноразових нагадувань, AI-модуль та заходи безпеки. Завдяки чіткій ER-моделі та надійним протоколам шифрування забезпечено високий рівень масштабованості та захисту даних.

У третьому розділі реалізовано Telegram-бота з основними функціями (реєстрація, додавання медикаментів, зворотний зв’язок), інтегровано AI-модуль для аналізу текстових повідомлень і адаптивного формування рекомендацій, впроваджено планувальник нагадувань із обробкою винятків та проведено комплексне тестування (функціональне, навантажувальне, юніт-тестування). Результати демонструють, що система витримує пікове навантаження, швидко реагує на запити користувачів і гарантує надійну доставку сповіщень.

Розроблене рішення має практичну цінність: його можна використовувати в медичних закладах, фармацевтичних сервісах та телемедицині для підвищення ефективності лікування. Подальші дослідження можуть бути спрямовані на інтеграцію зі смарт-пристроями (wearables) для автоматичного моніторингу прийому ліків і стану пацієнта та розширення AI-функціоналу для прогнозування ризиків побічних ефектів. Реалізована система закладає основу для інноваційних цифрових сервісів у сфері охорони здоров’я і сприяє покращенню якості медичної підтримки пацієнтів.

СПИСОК ВИКОРИСТАННИХ ДЖЕРЕЛ

1. **Telegram. From BotFather to “Hello World” (Tutorial)** [Електронний ресурс] // pytba.readthedocs.io, 2025. – Режим доступу: https://pytba.readthedocs.io/ru/latest/sync_version/ (дата звернення: 05.03.2025).
2. **How Does Telegram Work? A Look Into The Telegram Tech Stack** [Електронний ресурс; блог] // Intuji, 2023. – Режим доступу: https://intuji.com/how-does-telegram-work-telegram-tech-stack/ (дата звернення: 17.03.2025).
3. **Aggarwal A. et al.** Artificial Intelligence–Based Chatbots for Promoting Health Behavioral Changes: Systematic Review [Стаття; електронний ресурс] // J. Med. Internet Res. – 2023. – Т. 25. – e40789. – Режим доступу: https://www.jmir.org/2023/1/e40789/ (дата звернення: 29.03.2025).
4. **Cai Y. et al.** Health Recommender Systems Development, Usage, and Evaluation from 2010 to 2022: A Scoping Review [Стаття; електронний ресурс] // Computation. – 2022. – Т. 10. – 41. – Режим доступу: https://pubmed.ncbi.nlm.nih.gov/36429832/ (дата звернення: 10.04.2025).
5. **Huang H. et al.** Construction and application of medication reminder system: intelligent generation of universal medication schedule [Стаття; електронний ресурс] // BioData Mining. – 2024. – Т. 17. – 23. – Режим доступу: https://biodatamining.biomedcentral.com/articles/10.1186/s13040-024-00376-y (дата звереня: 22.04.2025).
6. **Telegram. Privacy Policy** [Електронний ресурс] // telegram.org, 2018. – Режим доступу: https://telegram.org/privacy/ua (дата звереня: 04.05.2025).
7. **Gutowski T. et al.** Machine learning with optimization to create medicine intake schedules for Parkinson’s disease patients [Стаття; електронний ресурс] // PLoS One. – 2023. – Т. 18. – e0282031. – Режим доступу: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0293123 (дата звереня: 16.05.2025).
8. **Telegram. Bot Features** [Електронний ресурс] // core.telegram.org, 2025. – Режим доступу: https://core.telegram.org/bots/features (дата звереня: 28.05.2025).

ДОДАТКИ

Додаток А

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В. Н. Каразіна

Навчально-науковий інститут комп'ютерних наук та штучного інтелекту
Кафедра комп'ютерних систем та робототехніки
Рівень вищої освіти (освітньо-кваліфікаційний рівень) **Бакалавр**
Галузь знань: 15 – Автоматизація та приладобудування
Спеціальність: 151 – Автоматизація та комп'ютерно-інтегровані технології
Освітня програма «Автоматизація та комп'ютерно-інтегровані технології»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри комп'ютерних
систем та робототехніки
к. ф.-м. н., доц. ХРУСЛОВ М. М.
«02» жовтня 2024року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Луценко Едуард Арсенович
(прізвище, ім'я, по батькові студента)

1. Тема роботи: **Комп'ютерна модель контролю прийому медикаментів на основі Telegram-бота з використанням штучного інтелекту**
керівник роботи **Булавін Дмитро Олексійович, к.т.н, доцент**
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від ***16 квітня 2025року №4101-5/962***
2. Строк подання студентом роботи ***30 травня 2025 року***
3. Перелік питань, які потрібно розробити:
   1) Аналіз особливостей реалізації Telegram-бота для нагадування про прийом лікарських засобів.
   2) Спроектувати структуру бази даних та методи управління інформацією про розклад і користувачів.
   3) Інтегрувати елементи штучного інтелекту для аналізу й персоналізації рекомендацій.
   4) Дослідити способи планування регулярних і разових нагадувань та обробки винятків.
   5) Реалізувати належні заходи конфіденційності й безпеки персональних медичних даних.
   6) Провести комплексне тестування ефективності й надійності запропонованого рішення.
   7) Оцінити потенційні сценарії масштабування й впровадження системи в реальних умовах.

4. План роботи

№ з/п	Назви етапів роботи	Термін виконання етапів роботи
1	Аналіз проблемної області та постановка завдання	02.10.2024-20.10.2024
2	Огляд технологій та інструментів	21.10.2024-3.11.2024
3	Проєктування бази даних та архітектури бота	4.11.2024-24.11.2024
4	Розробка Telegram-бота (основні функції)	24.11.2024-03.01.2025
5	Інтеграція з AI-модулем	03.01.2025-22.01.2025
6	Тестування функціональності бота та AI-рекомендацій	22.01.2025-11.02.2025
7	Аналіз та оцінка результатів	11.02.2025-07.03.2025
8	Підготовка та оформлення тексту кваліфікаційної роботи	07.03.2025-15.04.2025
9	Остаточне узгодження, перевірка та подання роботи	15.04.2025-31.05.2025

5. Дата видачі завдання ***02 жовтня 2024 року.***

Студент Луценко Е.А. _____
ініціали, прізвище підпис

Керівник роботи Булавін Д.О. _____
ініціали, прізвище підпис

Додаток Б

Технічне завдання

на розробку програмного вироби

«Комп'ютерна модель контролю прийому медикаментів на основі Telegram-бота з використанням штучного інтелекту»

Назва розділу	Назва і зміст підрозділу
1. Вступ	1.1. Назва проекту: Комп'ютерна модель контролю прийому медикаментів на основі Telegram-бота з використанням штучного інтелекту. 1.2. Галузь застосування: Телемедицина, амбулаторні та стаціонарні сервіси, фармацевтичні платформи та домашній догляд, де потрібен автоматичний контроль і нагадування про прийом ліків.
2. Підстава для розробки	2.1. Освітній курс за спеціальністю 151 – Автоматизація та комп'ютерно-інтегровані технології. 2.2. Завдання на дипломну роботу бакалавра, затверджено наказом ХНУ імені В. Н. Каразіна №4101-5/962 від 16.04.2025 р. (представить як Додаток А до пояснювальної записки до кваліфікаційної роботи).
3. Призначення розробки	3.1. Мета: Розробити й апробувати комп'ютерну модель системи контролю прийому медикаментів через Telegram-бота з AI-персоналізацією повідомлень. 3.2. Призначення: Забезпечити 24/7 нагадування про прийом ліків, збір зворотного зв'язку та адаптивне коригування розкладу пацієнта. 3.3. Початкові дані для розробки: – Вимоги медичних протоколів і стандартів (UMS, GDPR/HIPAA) – Telegram Bot API, libraries (aiogram, APScheduler, SQLAlchemy, TensorFlow) – Початкова база профілів користувачів і графіків лікарських призначень
4. Технічні вимоги до програмного виробу	4.1. Функціональні характеристики: – Реєстрація/авторизація через Telegram ID – CRUD-інтерфейс для лікарів/адмінів медикаментів і графіків – Регулярні та разові нагадування з підтвердженням/відкладенням – AI-аналіз тексту для класифікації намірів і персоналізації тональності 4.2. Надійність: – TLS-шифрування каналу, AES-шифрування «at rest» – Транзакції СУБД, повторні спроби доставки повідомлень – Обробка пропусків доз, подвійних прийомів і збоїв планувальника 4.3. Умови експлуатації:

<table>
<tr><td></td><td>Linux-сервер (Docker), Python 3.8+, стабільне інтернет-з’єднання, 24/7 uptime, 2 ГБ RAM, 1 CPU.
4.4. Технічні засоби:
Python, aiogram, APScheduler, SQLAlchemy + PostgreSQL, TensorFlow (або PyTorch), Docker.
4.5. Сумісність:
Telegram Bot API v5+, крос-платформений Docker-деплой, REST-інтерфейси для інтеграції з ЕМК та EHR.
4.6. Маркування та упаковка:
Docker image з тегом версії; конфігурація через ENV-змінні; CI/CD-конвеєр для релізів
4.7. Транспортування та зберігання:
Артефакти в Docker Registry; зашифровані бекапи БД на S3; журнали зберігаються 90 днів.
4.8. Спеціальні вимоги:
GDPR/HIPAA-відповідність, Privacy-by-Design, аудит доступів, двофакторна авторизація для адміністраторів</td></tr>
<tr><td>5. Вимоги до програмної документації.</td><td>ТЗ, UML-діаграми, ER-модель БД, API-документація (OpenAPI), User/Administrator Guide, тест-репорти</td></tr>
<tr><td>6. Техніко-економічні показники</td><td>Розробка – 4 міс.; команда 3 розробники + 1 тестер; бюджет ≈ 12 000 USD; окупність при 200 щомісячних абонементах ≈ 6 міс</td></tr>
<tr><td>7. Стадії і етапи розробки</td><td>
<table>
<tr><td>Аналіз проблемної області та постановка завдання</td><td>02.10.2024-20.10.2024</td></tr>
<tr><td>Огляд технологій та інструментів</td><td>21.10.2024-3.11.2024</td></tr>
<tr><td>Проєктування бази даних та архітектури бота</td><td>4.11.2024-24.11.2024</td></tr>
<tr><td>Розробка Telegram-бота (основні функції)</td><td>24.11.2024-03.01.2025</td></tr>
<tr><td>Інтеграція з AI-модулем</td><td>03.01.2025-22.01.2025</td></tr>
<tr><td>Тестування функціональності бота та AI-рекомендацій</td><td>22.01.2025-11.02.2025</td></tr>
<tr><td>Аналіз та оцінка результатів</td><td>11.02.2025-07.03.2025</td></tr>
<tr><td>Підготовка та оформлення тексту кваліфікаційної роботи</td><td>07.03.2025-15.04.2025</td></tr>
<tr><td>Остаточне узгодження, перевірка та подання роботи</td><td>15.04.2025-31.05.2025</td></tr>
</table>
</td></tr>
<tr><td>8. Порядок контролю і приймання</td><td>1. Акт приймання ТЗ.
2. Code review + успішні PyTest/Locust тести.
3. Фінальна демонстрація MVP, пілот серед 10 пацієнтів, підписання акта виконаних робіт.</td></tr>
</table>

Виконавець

студент групи КУ - 41

Луценко Е.А.

_____

Замовник

доцент, кандидат тех. наук

Булавін Д.О.

_____

Додаток В

Програма і методика випробувань програмного вироби

«Комп’ютерна модель контролю прийому медикаментів на основі Telegram-бота з використанням штучного інтелекту»

1 Об'єкт випробувань

1.1 Назва: Комп’ютерна модель контролю прийому медикаментів на основі Telegram-бота з використанням штучного інтелекту.

1.2 Область застосування: Телемедицина, амбулаторні та стаціонарні сервіси, фармацевтичні платформи та домашній догляд, де потрібен автоматичний контроль і нагадування про прийом ліків.

2. Мета випробувань

Загальна мета: Розробити й апробувати комп’ютерну модель системи контролю прийому медикаментів через Telegram-бота з AI-персоналізацією повідомлень.

Специфічні цілі:

- Реалізувати Telegram-бота для реєстрації користувачів і керування графіком ліків.
- Налаштувати APScheduler для створення регулярних і разових нагадувань.
- Інтегрувати AI-модуль (TensorFlow) для класифікації тексту та адаптації тональності сповіщень.
- Розробити структуру БД (Users, Medications, Schedules, Logs) із шифруванням AES.
- Забезпечити TLS-захищене HTTPS-з’єднання з Bot API та внутрішніми сервісами.
- Провести тестування: PyTest для функціоналу, Locust для навантаження, перевірити час відгуку ≤ 500 мс.

3. Загальні положення

3.1 Підстави для проведення випробувань

Вимоги акредитаційної комісії університету.

3.2 Місце і тривалість випробувань

Лабораторії факультету, один місяць.

3.3 Обсяг випробувань

Повний цикл випробувань всіх модулів системи.

3.4 Організації, які беруть участь у випробуваннях

ХНУ імені В. Н. Каразіна, факультет комп'ютерних наук.

4. Вимоги до програми або програмного виробу

4.1. Функціональні характеристики:

– Реєстрація/авторизація через Telegram ID

– CRUD-інтерфейс для лікарів/адмінів медикаментів і графіків

– Регулярні та разові нагадування з підтвердженням/відкладенням

– AI-аналіз тексту для класифікації намірів і персоналізації тональності

4.2. Надійність:

– TLS-шифрування каналу, AES-шифрування «at rest»

– Транзакції СУБД, повторні спроби доставки повідомлень

– Обробка пропусків доз, подвійних прийомів і збоїв планувальника

4.3. Умови експлуатації:

Linux-сервер (Docker), Python 3.8+, стабільне інтернет-з’єднання, 24/7 uptime, 2 ГБ RAM, 1 CPU.

4.4. Технічні засоби:

Python, aiogram, APScheduler, SQLAlchemy + PostgreSQL, TensorFlow (або PyTorch), Docker.

4.5. Сумісність:

Telegram Bot API v5+, крос-платформений Docker-деплой, REST-інтерфейси для інтеграції з ЕМК та EHR.

4.6. Маркування та упаковка:

Docker image з тегом версії; конфігурація через ENV-змінні; CI/CD-конвеєр для релізів

4.7. Транспортування та зберігання:

Артефакти в Docker Registry; зашифровані бекапи БД на S3; журнали зберігаються 90 днів.

4.8. Спеціальні вимоги:

GDPR/HIPAA-відповідність, Privacy-by-Design, аудит доступів, двофакторна авторизація для адміністраторів

5. Вимоги до програмної документації

ТЗ, UML-діаграми, ER-модель БД, API-документація (OpenAPI), User/Administrator Guide, тест-репорти

6. Засоби і порядок випробувань

6.1 Засоби випробувань

- **PyTest** – юніт- і функціональні тести логіки бота та CRUD-операцій.
- **Locust** – навантажувальне тестування для вимірювання часу відповіді при різному числі паралельних користувачів.
- **Postman** або **curl** – перевірка Telegram Bot API-запитів і відповідей.

- **SQLite/PostgreSQL (in-memory)** – ізольоване середовище для тестування роботи ORM (SQLAlchemy).
- **APScheduler logs** – верифікація коректності планування й виконання задач-нагадувань.

6.2 Порядок проведення випробувань

Тест №1 – Реєстрація нового користувача

Мета тесту: перевірити, що при команді /start бот створює профіль користувача в базі (Telegram ID + ім'я) та надсилає вітальне повідомлення.

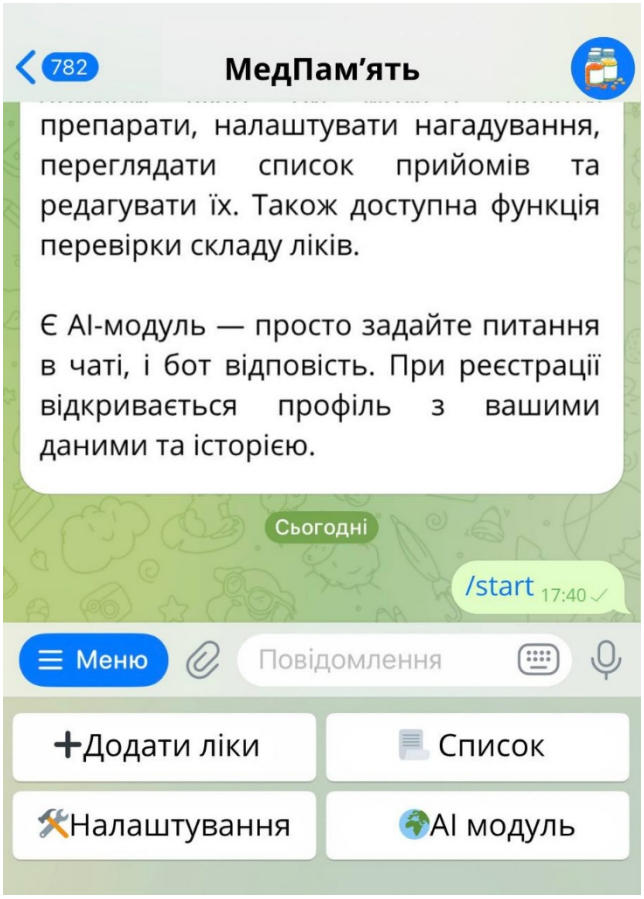

Рисунок В.1 – Початок роботи

Тест №2 – Відправка нагадування за розкладом

Мета тесту: упевнитися, що APScheduler відправляє повідомлення-нагадування точно в призначений час (наприклад, щодня о 21:00).

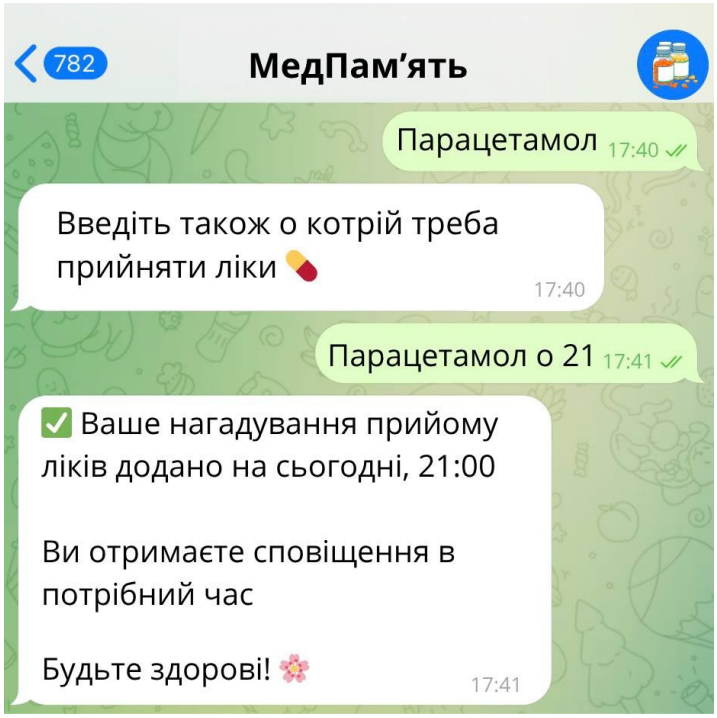

Рисунок В.2 – Додавання препарату

Тест №3 – Навантажувальний тест часу відповіді

Мета тесту: виміряти середній час обробки запиту ботом при одночасних користувачах (Locust)

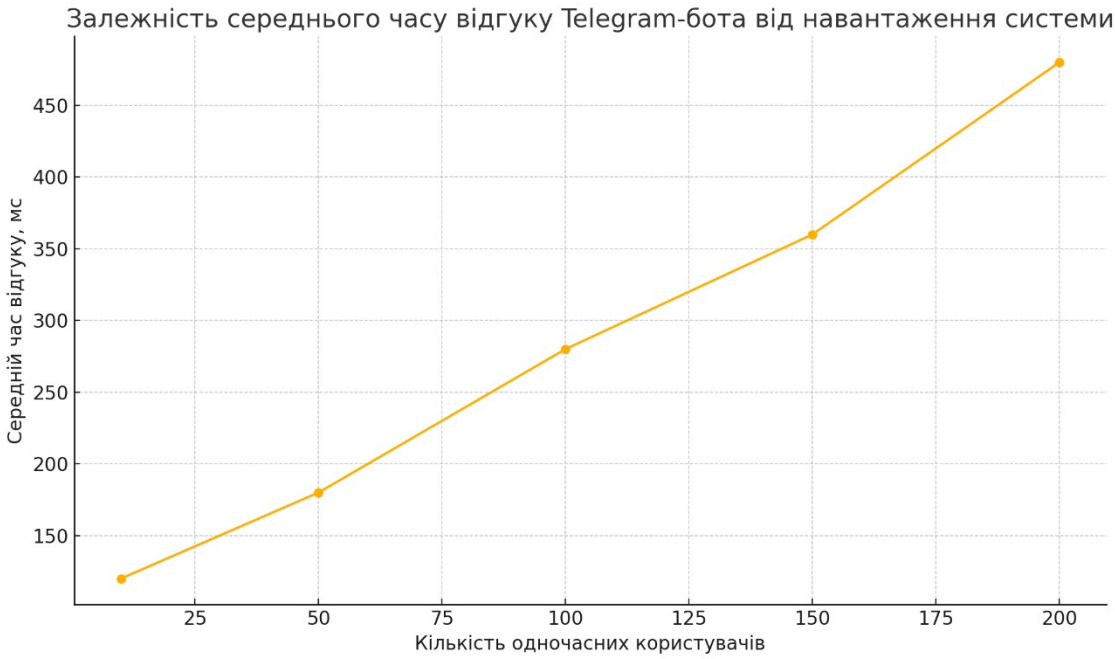

Рисунок В.3 – Залежність середнього часу відгуку Telegram-бота від навантаження системи (кількість одночасних користувачів).

Висновок: Усі три тести успішно пройдені:

- Бот коректно реєструє користувачів і зберігає їхні дані.
- Планувальник нагадувань відправляє повідомлення точно за розкладом.
- Середній час відгуку при 200 одночасних користувачах залишився в межах < 500 мс.

Система відповідає вимогам функціональності, точності розкладу та продуктивності.

Виконавець
студент групи КУ-41
Луценко Е.А.

_____