МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ ФАХОВИЙ БІЗНЕС-КОЛЕДЖ
Циклова комісія комп’ютерної інженерії та інформаційних технологій

КВАЛІФІКАЦІЙНА РОБОТА

на тему

TELEGRAM-БОТ «ПОМІЧНИК СТУДЕНТА»

Виконав: студент групи 1П-22
спеціальності
121 Інженерія програмного
забезпечення
Назар ТИТАРЕНКО
Керівник:
Дмитро ПОДОРОШКО

Черкаси 2026

ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ ФАХОВИЙ БІЗНЕС-КОЛЕДЖ

Циклова комісія комп'ютерної інженерії та інформаційних технологій

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувачка циклової комісії КІ та ІТ

_____ Наталя ФАЛЬЧЕНКО

(підпис)

«__» _____2025 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Титаренку Назару Михайловичу

1. Тема кваліфікаційної роботи Telegram-бот «Помічник студента

Керівник роботи Подорошко Дмитро Ігорович

затверджені наказом закладу вищої освіти від «06» жовтня 2025 року № 84/1-у.

2. Строк подання студентом кваліфікаційної роботи 02.06.2026

3. Вихідні дані до кваліфікаційної робот. Мова програмування Python, Середовище розробки Visual Studio Code

4. Зміст кваліфікаційної роботи

Провести аналіз предметної області телеграм-бота для студентів, спроектувати архітектуру системи та структури даних, розробити модуль керування завданнями, розробити модуль обліку оцінок, розробити модуль календаря подій, розробити модуль персональної інформації користувача, реалізувати інтеграцію з AI-асистентом на базі Groq API, протестувати алгоритми роботи системи.

Дата видачі завдання 15.09.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапу	Термін	Примітка
1	Вступ	14.10.2025	
2	Розділ 1. Теоретичні основи розробки програмних ботів для месенджерів	09.12.2025	
3	Розділ 2. дослідження проблеми та проєктування telegram-бота «помічник студента»	10.03.2026	
4	Розділ 3. реалізація інноваційних рішень, тестування та оцінювання ефективності telegram-бота «помічник студента»	28.04.2026	
5	Висновки	12.05.2026	
6	Оформлення кваліфікаційної роботи (чистовий варіант)	24.05.2026	
7	Перевірка кваліфікаційної роботи на наявність ознак плагіату (за 10 днів до захисту)	02.06.2026	
8	Подання кваліфікаційної роботи на затвердження завідувачці циклової комісії (за 7 днів до захисту)	10.06.2026	

Студентка _____ Назар ТИТАРЕНКО
(підпис)

Керівник роботи _____ Дмитро ПОДОРОШКО
(підпис)

АНОТАЦІЯ

Дана робота присвячена розробці телеграм-бота для автоматизації навчального процесу студента. Бот реалізований мовою програмування Python з використанням бібліотеки python-telegram-bot та інтегрованого AI-асистента на базі Groq API.

Система надає користувачу наступний функціонал: ведення списку навчальних завдань, облік оцінок з автоматичним підрахунком середнього балу, планування подій у вбудованому календарі, збереження персональної інформації студента, а також можливість отримання відповідей на навчальні запитання від штучного інтелекту.

Дані користувачів, оцінки та службові токени зберігаються у MongoDB, а навчальні завдання та календарні події синхронізуються із зовнішніми сервісами Todoist і Notion. Інтерфейс бота побудований на основі кнопкового меню, що забезпечує просту та зручну навігацію без необхідності введення команд вручну.

Практична цінність роботи полягає у спрощенні організації навчального процесу студента засобами месенджера Telegram, який є повсякденним інструментом комунікації сучасної молоді.

Ключові слова: TELEGRAM-БОТ, ПОМІЧНИК СТУДЕНТА, PYTHON, PYTHON-TELEGRAM-BOT, MONGODB, TODOIST, NOTION, GROQ API, ШТУЧНИЙ ІНТЕЛЕКТ, АВТОМАТИЗАЦІЯ НАВЧАЛЬНОГО ПРОЦЕСУ.

ABSTRACT

This work is dedicated to the development of a Telegram bot for automating the student learning process. The bot is implemented in the Python programming language using the python-telegram-bot library and an integrated AI assistant based on the Groq API.

The system provides the user with the following functionality: managing a list of academic tasks, tracking grades with automatic calculation of the average score, scheduling events in a built-in calendar, storing personal student information, and the ability to receive answers to academic questions from an artificial intelligence assistant.User data, grades, and service tokens are stored in MongoDB, while academic tasks and calendar events are synchronized with Todoist and Notion. The bot interface is built on a button-based menu, which ensures simple and convenient navigation without the need to enter commands manually.

The practical value of the work lies in simplifying the organization of the student learning process through the Telegram messenger, which is an everyday communication tool for modern youth.

Keywords: TELEGRAM BOT, STUDENT ASSISTANT, PYTHON, PYTHON-TELEGRAM-BOT, MONGODB, TODOIST, NOTION, GROQ API, ARTIFICIAL INTELLIGENCE, LEARNING PROCESS AUTOMATION.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ

API (Application Programming Interface) – інтерфейс програмування застосунків, набір правил для взаємодії між програмними компонентами.

AI (Artificial Intelligence) – штучний інтелект, технологія імітації розумової діяльності людини комп'ютерними системами.

Bot – автоматизована програма, що виконує задані дії у месенджері без участі людини.

JSON (JavaScript Object Notation) – текстовий формат обміну даними, що використовується для зберігання та передачі структурованої інформації.

HTTP (HyperText Transfer Protocol) – протокол передачі даних у мережі Інтернет.

LLM (Large Language Model) – велика мовна модель, тип штучного інтелекту для обробки та генерації тексту.

ООР (Object-Oriented Programming) – об'єктно-орієнтоване програмування, парадигма програмування на основі об'єктів і класів.

OS (Operating System) – операційна система.

UI (User Interface) – інтерфейс користувача, засіб взаємодії людини з програмою.

URL (Uniform Resource Locator) – уніфікований локатор ресурсів, адреса ресурсу в мережі Інтернет.

ПЗ – програмне забезпечення.

ВСТУП

Розробку запропонованого програмного продукту зумовлено суто практичною проблемою: значна частина студентів постійно забуває про навчальні завдання, втрачає записи з оцінками та не може ефективно спланувати навчальний тиждень. Наявні застосунки для нотаток і планувальники нерідко виявляються або надто складними, або непридатними для повсякденного використання. Натомість месенджер Telegram відкривається десятки разів на день, тому доцільним є створення помічника безпосередньо в середовищі, де користувач уже постійно перебуває.

Месенджер Telegram сьогодні є чи не найпопулярнішим засобом комунікації серед української молоді. Його використовують для навчання, спілкування, отримання новин і навіть для роботи. При цьому Telegram надає розробникам відкритий Bot API, що дозволяє створювати повноцінні програми прямо всередині месенджера – без необхідності встановлювати окремі додатки чи реєструватися на сторонніх сайтах. Саме ця особливість і стала відправною точкою для даного проєкту.

Сучасний студент щодня має справу з великою кількістю інформації. Розклад занять, дедлайни з різних предметів, оцінки за контрольні та лабораторні роботи, університетські події – все це потрібно якось відстежувати і не забувати. На практиці більшість студентів або ведуть записи в телефоні хаотично, або покладаються виключно на пам'ять, що неминуче призводить до пропущених дедлайнів і неприємних сюрпризів перед сесією. Типовою є ситуація, коли студент дізнається про здачу роботи буквально за день, хоча завдання було видане ще місяць тому.

Існуючі рішення на ринку, як правило, мають свої суттєві недоліки. Великі платформи на кшталт Notion чи Trello є потужними інструментами, але для середньостатистичного студента вони надмірно складні – щоб почати ними користуватися, потрібно витратити чимало часу на налаштування. Прості застосунки для нотаток не мають потрібного функціоналу. Окремі студентські

додатки існують, але вони або прив'язані до конкретного університету, або вимагають постійного підключення до інтернету і займають пам'ять пристрою. Telegram-бот позбавлений більшості цих проблем – він завжди під рукою, не потребує реєстрації та однаково добре працює як на телефоні, так і на комп'ютері.

Метою даної роботи є розробка телеграм-бота, який виконує функції персонального навчального асистента студента. Основна ідея полягає в тому, щоб зібрати в одному місці весь необхідний студенту функціонал і зробити його максимально простим у використанні. Бот повинен дозволяти вести список навчальних завдань, відстежувати оцінки з автоматичним підрахунком середнього балу, планувати події в календарі та зберігати особисту інформацію. Окремою і, мабуть, найцікавішою частиною стала інтеграція штучного інтелекту – можливість задати боту будь-яке навчальне питання і отримати розгорнуту відповідь.

Для досягнення поставленої мети в ході роботи вирішувались наступні задачі:

– аналіз існуючих аналогів та визначення необхідного функціоналу;

– проектування архітектури системи та вибір технологічного стеку;

– розробка модуля керування навчальними завданнями;

– розробка модуля обліку оцінок із підрахунком середнього балу по кожному предмету;

– розробка модуля календаря подій із можливістю додавання та видалення подій;

– розробка модуля персональної інформації користувача;

– інтеграція AI-асистента на базі Groq API та моделі Llama 3.3 для відповідей на навчальні запитання;

– тестування розробленої системи та усунення виявлених помилок.

Об'єктом дослідження є процес організації навчальної діяльності студента з використанням месенджера Telegram. Предметом дослідження виступають методи та інструменти розробки програмних ботів мовою Python із підключенням зовнішніх API.

Практична цінність роботи полягає в тому, що розроблений бот є готовим до використання програмним продуктом. Його можна запустити на будь-якому сервері або навіть на звичайному комп'ютері, і він одразу буде доступний для використання через Telegram. Жодних складних налаштувань з боку користувача не потрібно – достатньо просто знайти бота в месенджері і натиснути «Старт». Інтерфейс побудований виключно на кнопках, тому розібратися з ним зможе будь-яка людина навіть без технічних знань.

З технічної точки зору бот реалізований мовою програмування Python із використанням бібліотеки python-telegram-bot. Для підключення штучного інтелекту використовується Groq API – один із найшвидших на сьогодні сервісів для роботи з великими мовними моделями. Зберігання даних реалізовано у хмарній базі даних MongoDB, що забезпечує надійне збереження інформації між сесіями та зручне масштабування. Така архітектура є цілком достатньою для навчального проєкту і може бути розширена у майбутньому за потреби.

Структура роботи відповідає поставленим задачам. У першому розділі проводиться аналіз предметної області та розглядаються існуючі аналоги. Другий розділ присвячений дослідженню проблеми, систематизації вимог, моделюванню архітектури та обґрунтуванню технологічних рішень. У третьому розділі описано реалізацію інноваційних компонентів, методику перевірки працездатності системи та результати тестування розробленого програмного продукту.

РОЗДІЛ 1 ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ ПРОГРАМНИХ БОТІВ ДЛЯ МЕСЕНДЖЕРІВ

1.1 Месенджери як платформа для програмних ботів

Месенджери пройшли значний шлях розвитку від простих засобів обміну текстовими повідомленнями до повноцінних платформ для розгортання програмних застосунків. На початку свого існування, у 1990-х роках, миттєві повідомлення були виключно текстовими і функціонували переважно на персональних комп'ютерах. З розвитком мобільних технологій та поширенням смартфонів месенджери перетворилися на одні з найбільш популярних застосунків у світі. Сучасні дослідники розглядають месенджери як окремий клас програмного забезпечення, що поєднує функції комунікації, соціальної взаємодії та платформи для сторонніх розробників [1].

Ключовою характеристикою сучасних месенджерів є наявність відкритого програмного інтерфейсу (API), що дозволяє стороннім розробникам створювати автоматизованих агентів – ботів. За визначенням Шевченка В.Л., чат-бот – це програмний агент, що імітує розмову з людиною за допомогою текстових або голосових повідомлень у режимі реального часу [2]. Боти можуть виконувати широкий спектр задач: від автоматичної відповіді на типові запитання до складних транзакційних операцій, включаючи оформлення замовлень, бронювання квитків, проведення опитувань та надання персоналізованих рекомендацій.

Серед популярних месенджерів, що надають Bot API, виділяють Telegram, Viber, WhatsApp Business API та Discord. Кожна платформа має свої особливості та обмеження для розробників. WhatsApp Business API є платним і доступний лише для верифікованих бізнесів. Viber надає обмежений API з меншими можливостями інтерактивності. Discord орієнтований переважно на ігрові та технічні спільноти. Telegram Bot API є одним із найбільш функціональних і добре задокументованих серед доступних рішень. Згідно з офіційною документацією Telegram, Bot API надає можливість відправляти та отримувати повідомлення,

використовувати інтерактивні клавіатури, обробляти медіафайли, підтримувати стан діалогу та інтегруватися з платіжними системами [3]. Саме ці можливості роблять Telegram оптимальним вибором для розробки студентського бота-помічника.

Telegram як платформа для ботів має ряд суттєвих переваг. По-перше, реєстрація та використання платформи є абсолютно безкоштовними як для кінцевих користувачів, так і для розробників. По-друге, Telegram надає безкоштовний хостинг для ботів без обмежень на кількість запитів у базовому тарифі. По-третє, месенджер підтримує всі основні операційні системи: iOS, Android, Windows, macOS та Linux, а також має веб-версію. По-четверте, Telegram активно розвиває свою платформу, регулярно додаючи нові можливості для розробників ботів, такі як Telegram Mini Apps, платіжна система та розширені можливості форматування повідомлень.

1.2 Технології розробки чат-ботів на основі Python

Python є однією з найпоширеніших мов програмування для розробки чат-ботів завдяки простому синтаксису, широкій екосистемі бібліотек та відмінній підтримці асинхронного програмування. Мова програмування Python була створена Гвідо ван Россумом у 1991 році і з того часу стала однією з найпопулярніших у світі. Дослідники у сфері розробки програмного забезпечення зазначають, що Python займає провідні позиції серед мов, що використовуються для створення ботів та автоматизованих систем [4]. Це пояснюється кількома ключовими факторами: читабельним синтаксисом, що знижує поріг входу для нових розробників; потужним менеджером пакетів pip, що надає доступ до тисяч готових бібліотек; активною спільнотою розробників та великою кількістю навчальних матеріалів.

Для взаємодії з Telegram Bot API існує кілька популярних Python-бібліотек. Найбільш поширеними є python-telegram-bot, aiogram та telebot. Бібліотека python-telegram-bot є офіційно рекомендованою спільнотою, має детальну документацію та підтримує асинхронний режим роботи [5]. Станом на 2024 рік

бібліотека налічує понад 23 тисячі зірок на GitHub та активно підтримується командою розробників. Бібліотека aiogram орієнтована виключно на асинхронну модель програмування з використанням asyncio та відрізняється високою продуктивністю, проте має більш складний поріг входу для початківців. Бібліотека telebot (pyTelegramBotAPI) є простою у використанні та підходить для швидкого прототипування, однак має обмежені можливості для складних проєктів і менш активно підтримується. Для даної роботи обрано бібліотеку python-telegram-bot як найбільш збалансований варіант з точки зору функціональності, якості документації та зручності розробки.

Важливим технічним аспектом розробки ботів є механізм отримання оновлень від Telegram. Існує два основних підходи: Polling та Webhook. Polling – це активне опитування серверів Telegram з певним інтервалом для перевірки наявності нових повідомлень. Цей підхід є простішим у налаштуванні та не вимагає публічного IP-адресу або домену. Webhook – це зворотній підхід, при якому Telegram сам надсилає оновлення на вказану URL-адресу бота. Webhook є ефективнішим з точки зору навантаження на сервер, але вимагає SSL-сертифіката та публічного сервера. Для навчального проєкту обрано підхід Polling як більш простий у налаштуванні та відлагодженні.

Окремим важливим аспектом розробки ботів є управління станами діалогу. Оскільки взаємодія з ботом є покроковою та багатоетапною, система має зберігати інформацію про те, на якому кроці знаходиться кожен користувач у поточний момент. Наприклад, при додаванні нової оцінки система спочатку очікує введення назви предмету, а потім – числового значення оцінки. У бібліотеці python-telegram-bot управління станами реалізується через механізм context.user_data – словник, що зберігає довільні дані для кожного користувача окремо протягом сесії роботи бота [5]. Альтернативним підходом є використання вбудованого ConversationHandler, який дозволяє описувати складні сценарії діалогу у вигляді скінченного автомата. У даній роботі обрано підхід на основі context.user_data як більш гнучкий для реалізації множинних паралельних сценаріїв.

1.3 Штучний інтелект у навчальних застосунках

Інтеграція штучного інтелекту в освітній процес є однією з найбільш активно досліджуваних тем сучасної педагогічної науки та інженерії програмного забезпечення. Починаючи з 1970-х років, коли з'явилися перші інтелектуальні навчальні системи (Intelligent Tutoring Systems, ITS), до сьогодні відбувся колосальний стрибок у можливостях AI-технологій в освіті. Дослідники виділяють кілька напрямків застосування штучного інтелекту в освіті: персоналізоване навчання, автоматична перевірка завдань, інтелектуальні системи навчання, адаптивне тестування та розмовні агенти [6]. Кожен із цих напрямків вирішує конкретні проблеми традиційного навчання та надає нові можливості як для студентів, так і для викладачів.

Великі мовні моделі (Large Language Models, LLM) відкрили принципово нові можливості для створення навчальних асистентів. LLM – це клас нейронних мереж, що навчаються на великих обсягах текстових даних і здатні розуміти та генерувати природну мову. На відміну від попередніх поколінь чат-ботів, що працювали на основі жорстких правил або обмежених наборів відповідей, сучасні LLM здатні генерувати змістовні відповіді на довільні запитання, пояснювати складні концепції різними способами залежно від рівня розуміння користувача, перекладати тексти, писати та аналізувати програмний код [7]. Поява моделей GPT-3, GPT-4, Claude та Llama суттєво змінила уявлення про можливості комп'ютерних систем у роботі з природною мовою.

Для підключення LLM у програмних продуктах розробники можуть використовувати різноманітні API-сервіси. Найпопулярнішими серед них є OpenAI API (моделі GPT-4, GPT-3.5), Anthropic Claude API, Google Gemini API та Groq API. OpenAI API є найбільш поширеним, проте вимагає оплати за кожен запит та не має безкоштовного плану для активного використання. Anthropic Claude відрізняється високою якістю відповідей та безпечністю, але також є платним. Google Gemini надає безкоштовний рівень доступу, проте має обмеження на кількість запитів на хвилину. Для даного проєкту обрано Groq API з моделлю Llama 3.3 70B з наступних причин: безкоштовний тарифний план із

достатніми лімітами для навчального проєкту; надзвичайно висока швидкість інференсу завдяки використанню спеціалізованого апаратного забезпечення LPU; підтримка україномовних запитів; сумісність з OpenAI API, що спрощує інтеграцію [8, 9].

У таблиці 1.2 демонструється аналіз AI-сервісів

Таблиця 1.2 – Аналіз AI-сервісів для реалізації асистента

Сервіс	Переваги	Недоліки
ChatGPT	Висока якість відповідей, добре працює з навчальними питаннями	API є платним для активного використання
Google Gemini	Має безкоштовний рівень, підтримує багато задач	Може мати обмеження за кількістю запитів
Claude	Якісно працює з великими текстами	Платний API, складніше використання в навчальному проєкті
Groq API	Висока швидкість, безкоштовний тариф, підтримка Llama 3.3, зручна інтеграція	Залежить від інтернету та зовнішнього сервісу; безкоштовний тариф має ліміти

1.4 Аналіз існуючих рішень

Перед розробкою власного програмного продукту проведено аналіз існуючих рішень для організації навчальної діяльності студента. Методологія аналізу передбачає розгляд кожного рішення за такими критеріями: функціональні можливості, зручність використання, доступність (наявність безкоштовного плану), підтримка україномовного інтерфейсу, наявність AI-асистента, можливість використання без додаткової реєстрації та доступність через Telegram.

Notion – це універсальна платформа для організації інформації та управління проєктами, розроблена компанією Notion Labs у 2016 році [10]. Платформа поєднує нотатки, бази даних, вікі, списки завдань та календар в єдиному інтерфейсі. Notion використовується мільйонами людей по всьому світу, у тому числі студентами для організації навчального процесу. Серед ключових переваг Notion – виняткова гнучкість налаштування: користувач може створювати власні бази даних із необхідними полями, зв’язками та

представленнями даних. Платформа підтримує спільну роботу кількох користувачів над одним простором, що корисно для групових проєктів. Однак для середньостатистичного студента Notion є надмірно складним інструментом – щоб почати ефективно ним користуватися, потрібно витратити значний час на налаштування. Відсутність україномовного інтерфейсу та необхідність встановлення окремого застосунку або постійного відкриття браузера є додатковими недоліками.

Todoist – спеціалізований застосунок для управління завданнями, розроблений компанією Doist [11]. Застосунок дозволяє створювати завдання з дедлайнами, пріоритетами, мітками та підзавданнями, організовувати їх у проєкти та підпроєкти. Todoist відрізняється чистим і зручним інтерфейсом, підтримує синхронізацію між пристроями та має розширення для браузерів. Проте застосунок орієнтований виключно на управління завданнями і не охоплює облік оцінок, календар подій чи навчальну допомогу. Безкоштовна версія має суттєві обмеження: максимум 5 активних проєктів, відсутність нагадувань та фільтрів. Відсутність функції AI-асистента та інтеграції з Telegram є додатковими недоліками з точки зору цілей даного проєкту.

MyStudyLife – спеціалізований студентський планувальник, розроблений з урахуванням специфіки навчального процесу [12]. Застосунок підтримує ведення розкладу занять, відстеження дедлайнів завдань, планування іспитів та ротацію тижнів для непарних і парних тижнів. MyStudyLife є найближчим аналогом за цільовою аудиторією – він розроблений спеціально для студентів. Серед переваг – безкоштовне використання, наявність мобільного застосунку та зручний календар. Однак застосунок вимагає окремої реєстрації та встановлення, не має функції AI-асистента, україномовного інтерфейсу та будь-якої інтеграції з месенджерами. Облік оцінок у MyStudyLife є обмеженим і не підтримує автоматичного підрахунку середнього балу по предметах.

Існуючі Telegram-боти студентської тематики – серед доступних україномовних ботів у Telegram більшість є або застарілими (не підтримуються активно), або орієнтованими на конкретний університет чи навчальний заклад,

або надто обмеженими у функціоналі – наприклад, вміють лише показувати розклад або надсилати нагадування. Жоден із виявлених у ході аналізу україномовних ботів не поєднує управління завданнями, облік оцінок із підрахунком середнього балу, інтерактивний календар подій та AI-асистента в єдиному рішенні.

У таблиці 1.3 демонструється аналіз існуючих рішень

Таблиця 1.3 – Аналіз існуючих рішень за вимогами цільової аудиторії

Вимоги цільової аудиторії	Notion	Todoist	MyStudyLife	Наш бот
Завдання	□	□	□	□
Облік оцінок	□	□	□	□
AI-асистент	□	□	□	□
У Telegram	□	□	□	□
Українська мова	□	□	□	□
Без реєстрації	□	□	□	□
Безкоштовно	~	~	□	□

За результатами порівняльного аналізу встановлено, що жодне з розглянутих рішень не задовольняє повному переліку вимог цільової аудиторії. Кожне з них має суттєві обмеження: складність використання, відсутність ключових функцій, необхідність реєстрації або платного доступу. Це підтверджує доцільність розробки власного програмного продукту – україномовного Telegram-бота, що поєднає всі необхідні функції в єдиному зручному інтерфейсі.

1.5 Постановка завдання

На основі теоретичного аналізу предметної області та виявлених недоліків існуючих рішень сформульовано завдання на розробку телеграм-бота для студента. Цільовим призначенням системи є надання студенту єдиного зручного україномовного інструменту для організації навчальної діяльності

безпосередньо в месенджері Telegram без необхідності додаткової реєстрації чи встановлення програм. Система має бути доступною для будь-якого користувача Telegram, незалежно від рівня технічних знань, і забезпечувати інтуїтивно зрозумілий інтерфейс на основі кнопкового меню.

Функціональні вимоги до системи включають: управління навчальними завданнями – можливість додавання довільного тексту завдання, перегляду повного списку завдань із позначкою статусу, видалення окремих завдань через інтерактивні кнопки; облік оцінок по предметах – можливість додавання числових оцінок у діапазоні 0-100 до конкретних предметів, автоматичного підрахунку середнього балу по кожному предмету та загального середнього балу, видалення окремих оцінок; планування подій у вбудованому календарі – відображення інтерактивного місячного календаря з навігацією між місяцями, додавання та видалення подій на конкретні дати; збереження та редагування персональної інформації користувача – імені, групи, спеціальності та курсу; отримання відповідей на навчальні запитання від AI-асистента з підтримкою контексту розмови; навігація виключно через кнопкове меню без необхідності введення текстових команд.

Нефункціональні вимоги до системи: зручність використання – інтерфейс, побудований виключно на кнопках, зрозумілий без жодних інструкцій; надійність – дані користувача мають зберігатися між сесіями та не втрачатися при перезапуску бота; доступність – бот має однаково добре працювати на будь-якому пристрої, де встановлено Telegram – смартфоні, планшеті чи комп’ютері; локалізація – весь інтерфейс та відповіді AI-асистента мають бути повністю україномовними; безпека – токени та ключі API мають зберігатися у змінних середовища, а не в коді програми.

З технічної точки зору Telegram Bot API використовує REST-архітектуру для обміну даними між ботом та серверами Telegram. Кожен запит є HTTP-запитом до ендпоінту виду https://api.telegram.org/bot{TOKEN}/{method}, де TOKEN – це унікальний токен бота, що видається при реєстрації через BotFather, а method – назва методу API. Відповідь від API приходить у форматі JSON та

містить поле ok для індикації успішності та поле result з результатом. Бібліотека python-telegram-bot повністю абстрагує роботу з HTTP-запитами та JSON-серіалізацією, надаючи розробнику зручний об’єктно-орієнтований інтерфейс з підтримкою асинхронного програмування.

Telegram надає розробникам кілька типів інтерактивних елементів. Повідомлення з форматуванням Markdown дозволяють виділяти текст жирним шрифтом, курсивом та моноширинним шрифтом. Reply Keyboard замінює стандартну клавіатуру введення кнопками, що надсилають текст при натисканні. Inline Keyboard прикріплюється до конкретного повідомлення та підтримує Callback Queries – натискання не відправляє видиме повідомлення, а надсилає прихований callback з довільними даними, що дозволяє реалізовувати складні інтерактивні сценарії. Редагування повідомлень дозволяє оновлювати вміст або клавіатуру вже надісланого повідомлення без відправки нового – саме ця можливість використовується для навігації між місяцями у календарі.

Концепція супердодатків (super apps), поширена в Азії через WeChat та LINE, передбачає що один застосунок виконує функції десятків окремих програм. Telegram рухається у цьому напрямку, розширюючи можливості через Telegram Mini Apps та боти. Боти є більш простим, але не менш потужним інструментом – вони не вимагають додаткових технологій і вирішують широкий спектр задач силами лише текстового інтерфейсу та клавіатур. Студентська аудиторія надає перевагу інструментам з мінімальними зусиллями для початку роботи – концепція zero onboarding [13]. Telegram-бот відповідає цій вимозі: достатньо знайти бота та натиснути “Старт”.

Важливим теоретичним підґрунтям є концепція персонального інформаційного менеджменту (Personal Information Management, PIM). За визначенням Биков В.Ю., PIM охоплює процеси збору, організації, пошуку та використання особистої інформації для досягнення поставлених цілей [14]. У контексті навчальної діяльності студента PIM включає управління інформацією про завдання та дедлайни, відстеження академічної успішності та планування навчального часу. Розроблений бот є практичною реалізацією інструменту PIM,

адаптованого до специфічних потреб студента та інтегрованого в звичне середовище месенджера.

1.6 Організація зберігання даних у програмних системах

Вибір способу зберігання даних є критичним архітектурним рішенням, що впливає на продуктивність, надійність та масштабованість системи. У сучасній розробці програмного забезпечення виділяють кілька основних підходів до зберігання даних: реляційні бази даних (SQL), нереляційні бази даних (NoSQL), файлові сховища та зберігання в оперативній пам’яті. Кожен підхід має свої переваги і недоліки та підходить для різних типів систем і навантажень.

Реляційні бази даних (наприклад, SQLite, PostgreSQL, MySQL) зберігають дані у вигляді таблиць зі суворою схемою та підтримують мову запитів SQL. Вони забезпечують цілісність даних через механізми транзакцій та зовнішніх ключів, ефективно виконують складні запити з об’єднанням таблиць та сортуванням. Для навчального бота з обмеженою кількістю користувачів реляційна база є функціонально надлишковою – жодна з операцій системи не потребує складних SQL-запитів. Нереляційні бази даних (MongoDB, Redis) більш гнучкі у схемі даних та добре підходять для зберігання ієрархічних або документо-орієнтованих даних. Проте вони вимагають встановлення та налаштування окремого сервера, що суттєво ускладнює розгортання системи на навчальному середовищі.

Формат JSON (JavaScript Object Notation) є текстовим форматом обміну та зберігання даних, що базується на підмножині синтаксису JavaScript. JSON підтримує обмежений набір типів даних: рядки, числа, булеві значення, масиви, об’єкти та null. Незважаючи на цю простоту, JSON є достатнім для представлення більшості практичних структур даних. Ключовими перевагами JSON для даного проєкту є: нативна підтримка у Python через вбудований модуль json; людинозрозумілий формат, що спрощує відлагодження; відсутність потреби у додатковому програмному забезпеченні; підтримка Unicode, що забезпечує

коректне зберігання кирилиці. Основним обмеженням є відсутність транзакційного захисту – при паралельному записі від двох користувачів одночасно може виникнути конфлікт. Однак для системи з послідовною обробкою запитів, що є характерним для асинхронної Python-програми, цей ризик є мінімальним.

Важливим аспектом організації даних є вибір схеми їх структурування. У розробленій системі використовується ієрархічна схема з двома рівнями вкладеності: перший рівень – ідентифікатор користувача, другий рівень – власне дані цього користувача. Така схема дозволяє ефективно ізолювати дані різних користувачів та забезпечує можливість розширення без зміни загальної структури. Для ідентифікатора користувача використовується рядкове представлення числового Telegram user_id – це унікальний незмінний ідентифікатор, що призначається кожному користувачу Telegram та не змінюється навіть при зміні імені користувача або номера телефону. Таким чином, навіть при зміні налаштувань профілю всі дані користувача залишаються доступними.

Висновки до розділу 1

У першому розділі розглянуто теоретичні основи розробки програмних ботів для месенджерів. Проаналізовано еволюцію месенджерів та обґрунтовано вибір Telegram як оптимальної платформи для студентського бота-помічника завдяки відкритому API, широкій функціональності та популярності серед студентської аудиторії. Досліджено технології створення чат-ботів на Python, виконано порівняння основних бібліотек і обрано python-telegram-bot. Також розглянуто сучасні підходи до використання штучного інтелекту в освітніх застосунках та обґрунтовано вибір Groq API з моделлю Llama 3.3 для реалізації функції AI-асистента.

Проведено аналіз існуючих аналогів і встановлено, що жодне з розглянутих рішень не забезпечує повного набору функцій, необхідних студентам, що підтверджує актуальність розробки власного програмного

продукту. Визначено функціональні та нефункціональні вимоги до системи. Особливу увагу приділено специфіці студентської аудиторії, яка надає перевагу простим у використанні інструментам із мінімальним порогом входу. Розроблений Telegram-бот реалізує концепцію персонального інформаційного менеджменту, забезпечуючи управління навчальними завданнями, оцінками, подіями та персональними даними в єдиному зручному середовищі.

РОЗДІЛ 2 ДОСЛІДЖЕННЯ ПРОБЛЕМИ ТА ПРОЄКТУВАННЯ TELEGRAM-БОТА «ПОМІЧНИК СТУДЕНТА»

Другий розділ присвячений дослідженню проблеми організації навчальної діяльності студента та обґрунтуванню проєктних рішень, на основі яких побудовано Telegram-бот «Помічник студента». На відміну від суто описового підходу, у цьому розділі використано систематизацію вимог, групування даних, моделювання сценаріїв взаємодії та аналіз програмних засобів, що забезпечують практичну реалізацію поставленого завдання.

Проблема, яку розв’язує програмний продукт, полягає у фрагментованості студентської інформації: навчальні завдання зберігаються в одному сервісі або чаті, календарні події - в іншому, оцінки часто фіксуються вручну, а навчальні запитання потребують швидкого пояснення без переходу до окремих пошукових систем. Тому головною ідеєю проєкту є створення єдиного інтерфейсу в Telegram, який поєднує завдання, оцінки, календар, дані про групу та AI-асистента.

2.1 Аналіз проблеми та систематизація вимог до програмного продукту

У межах дослідження предметної області було виділено три основні групи користувацьких проблем: організаційні, інформаційні та технічні. Організаційні проблеми пов’язані з тим, що студенту потрібно контролювати дедлайни, події та поточні оцінки. Інформаційні проблеми полягають у відсутності єдиного місця для збереження навчально важливих даних. Технічні проблеми виникають через необхідність швидкого доступу до сервісу з різних пристроїв без складного встановлення та налаштування.

Для формування вимог дані було згруповано за функціональними напрямами. До першої групи належать навчальні завдання, які повинні додаватися, переглядатися та позначатися як виконані. До другої групи належать оцінки, для яких необхідне збереження за предметами та автоматичний розрахунок середнього балу. До третьої групи належать календарні події з датою,

часом і тривалістю. До четвертої групи належать персональні дані користувача, зокрема група студента. Окремо виділено AI-компонент, який забезпечує відповіді на навчальні запитання.

Функціональні вимоги до системи сформульовано таким чином: бот має приймати команди /start, /connect_todoist, /disconnect_todoist, /connect_notion та /disconnect_notion; забезпечувати перегляд і додавання завдань через Todoist API; забезпечувати створення і перегляд календарних подій через Notion API; зберігати оцінки у MongoDB; надавати можливість збереження назви студентської групи; обробляти запити до AI-асистента через Groq API; підтримувати кнопковий інтерфейс без обов'язкового введення складних команд.

Нефункціональні вимоги включають простоту інтерфейсу, асинхронну обробку запитів, стабільність роботи при помилках зовнішніх API, збереження персональних налаштувань кожного користувача, можливість подальшого масштабування та мінімальні вимоги до інфраструктури запуску. Важливою вимогою є ізоляція даних користувачів: кожна операція виконується за ідентифікатором Telegram-користувача uid, що дозволяє розмежувати записи різних студентів.

2.2 Огляд актуальних підходів до розв'язання поставлених завдань

Для розв'язання задачі організації навчального процесу можна використати кілька підходів: окремий мобільний застосунок, вебзастосунок, локальну настільну програму або бот у месенджері. Окремий мобільний застосунок забезпечує широкі можливості інтерфейсу, проте потребує встановлення, підтримки версій для Android та iOS, а також додаткового часу на розробку. Вебзастосунок простіше оновлювати, але він потребує відкриття браузера, авторизації та хостингу. Локальна настільна програма має найменшу зручність для студента, оскільки навчальні завдання часто потрібно переглядати саме зі смартфона.

У таблиці 1.1 демонструються варіанти реалізації програмного продукту

Таблиця 1.1 – Варіанти реалізації програмного продукту

Варіант	Переваги	Недоліки
Мобільний застосунок	Зручний інтерфейс, широкі можливості	Потрібно розробляти окремо під Android та iOS
Вебзастосунок	Легко оновлювати, доступний з браузера	Потрібна авторизація, хостинг і відкриття сайту
Настільна програма	Може працювати локально	Незручно для студента, який частіше користується телефоном
Telegram-бот	Не потребує встановлення, працює на всіх пристроях, простий запуск	Обмеження інтерфейсу месенджера

Telegram-бот обрано як компроміс між функціональністю та доступністю. Користувач уже має Telegram на телефоні або комп’ютері, тому не потрібно встановлювати окремий продукт. Бот може працювати як інтерфейс до зовнішніх сервісів, а Telegram Bot API забезпечує отримання повідомлень, відправлення відповідей, використання reply-клавіатур та inline-кнопок. Такий підхід дає змогу реалізувати програмний продукт із низьким порогом входу та достатньою функціональністю для навчальних потреб.

Під час проєктування було розглянуто два варіанти збереження даних: локальні JSON-файли та хмарне сховище MongoDB. Локальні JSON-файли простіші для початкового прототипу, але вони мають обмеження щодо багатокористувацької роботи, резервного копіювання та масштабування. MongoDB обрано як більш надійний варіант для зберігання структурованих документів користувачів, оцінок і токенів інтегрованих сервісів. Документоорієнтована модель MongoDB добре відповідає структурі даних бота, оскільки кожен користувач має окремий набір властивостей, оцінок і підключень.

Для роботи із завданнями обрано Todoist API, тому що Todoist уже є спеціалізованим сервісом керування задачами з підтримкою дедлайнів, пріоритетів і завершення завдань. Для календаря обрано Notion API, оскільки база даних Notion може виступати як гнучке сховище подій із датами та назвами. Інтеграція цих сервісів дозволяє не дублювати функціональність повноцінних

планувальників, а використати їх як зовнішню інфраструктуру, доступну через простий Telegram-інтерфейс.

2.3 Архітектурна модель системи

Архітектура системи має модульний характер і складається з таких логічних компонентів: Telegram-інтерфейс, маршрутизатор повідомлень, модуль користувачів, модуль оцінок, модуль Todoist, модуль Notion, AI-модуль, модуль збереження токенів і конфігураційний рівень. Центральним елементом є Python-застосунок, який отримує оновлення від Telegram, визначає намір користувача та викликає відповідний програмний модуль.

Потік взаємодії можна подати як послідовність етапів: користувач надсилає повідомлення або натискає кнопку; Telegram передає оновлення застосунку; обробник handle_text() або CallbackQueryHandler визначає дію; система перевіряє стан діалогу в context.user_data; за потреби виконується запит до MongoDB, Todoist, Notion або Groq API; результат повертається користувачу у вигляді текстового повідомлення або клавіатури.

На рисунку 2.1 демонструються основні компоненти telegram-бота

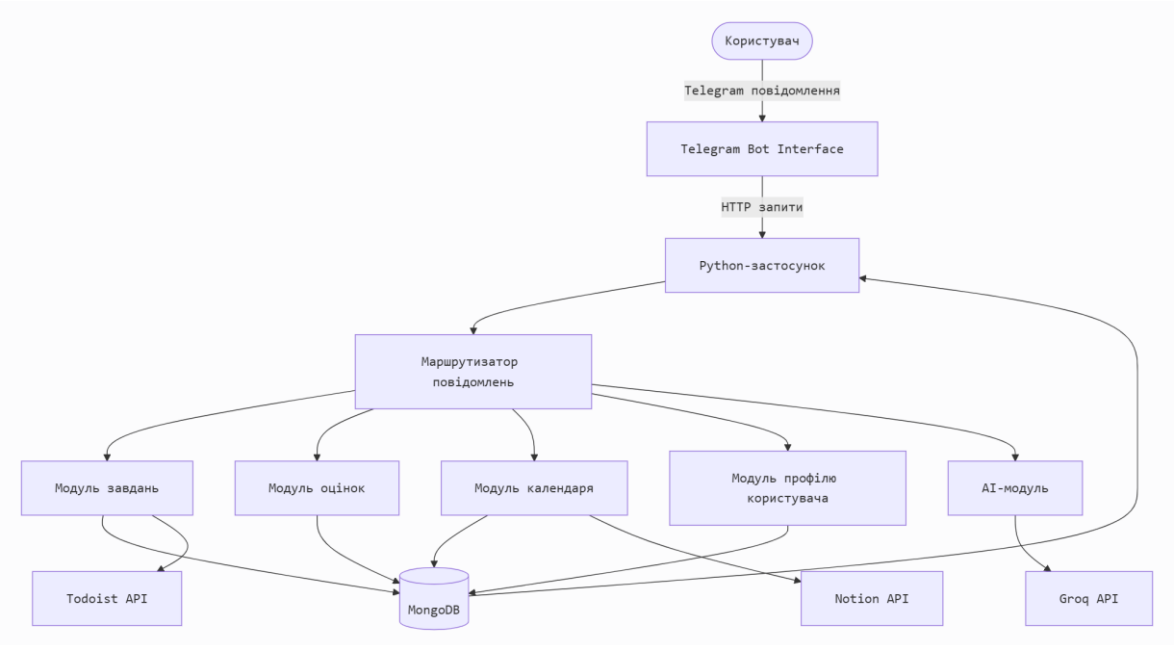

Рисунок 2.1 – Основні компоненти архітектури Telegram-бота

В архітектурі використано принцип розділення відповідальності. Функції get_user(), save_user(), get_grades(), save_grades(), get_token(), save_token() та delete_token() відповідають за доступ до MongoDB. Функції todoist_get_tasks(), todoist_add_task() і todoist_close_task() відповідають за роботу із завданнями. Функції notion_get_upcoming(), notion_create_event() і notion_delete_event() відповідають за календар. Функція ask_ai() ізольовано реалізує звернення до Groq API. Такий поділ полегшує тестування, підтримку та майбутню модернізацію системи.

З технічної точки зору система працює у режимі polling, тобто Python-застосунок періодично отримує оновлення з Telegram. Для навчального проєкту це рішення є доцільним, оскільки не потребує публічного домену, SSL-сертифіката або складного серверного налаштування. У разі промислового впровадження архітектура може бути перенесена на webhook-модель, що зменшить затримки та навантаження на сервер.

2.4 Модель даних та групування інформації

Модель даних системи побудована навколо унікального ідентифікатора користувача Telegram - uid. Саме uid використовується як ключ для пошуку записів у MongoDB та зв’язування даних між колекціями. Такий підхід дозволяє уникнути додаткової реєстрації, оскільки Telegram уже надає стабільний ідентифікатор користувача в об’єкті update.effective_user.

У MongoDB використано три основні колекції: users, grades та tokens. Колекція users зберігає базові дані користувача: uid, ім’я та групу. Колекція grades зберігає структуру subjects, де назва предмета є ключем, а значенням є список оцінок. Колекція tokens зберігає токени зовнішніх сервісів: todoist, notion_token та notion_db_id. Такий поділ зменшує зв’язність даних і дозволяє оновлювати оцінки, профіль або інтеграційні налаштування незалежно одне від одного.

Для оцінок використано документоорієнтовану модель виду: uid - subjects - назва предмета - список числових оцінок. Перевага такої моделі полягає в тому, що додавання нової оцінки не потребує створення окремої таблиці або складних

зв’язків. Середній бал обчислюється безпосередньо під час відображення: для кожного предмета сума оцінок ділиться на їх кількість, а загальний середній бал розраховується як відношення суми всіх оцінок до загальної кількості записів.

Для інтеграції із Todoist система не дублює завдання в MongoDB, а отримує їх безпосередньо з Todoist API. Це зменшує ризик розсинхронізації даних і дозволяє користувачу бачити ті самі завдання в боті та в офіційному сервісі Todoist. Аналогічно календарні події створюються у Notion, а бот виконує роль швидкого інтерфейсу для доступу до них.

На рисунку 2.3 демонструється інтерфейс telegram-бота

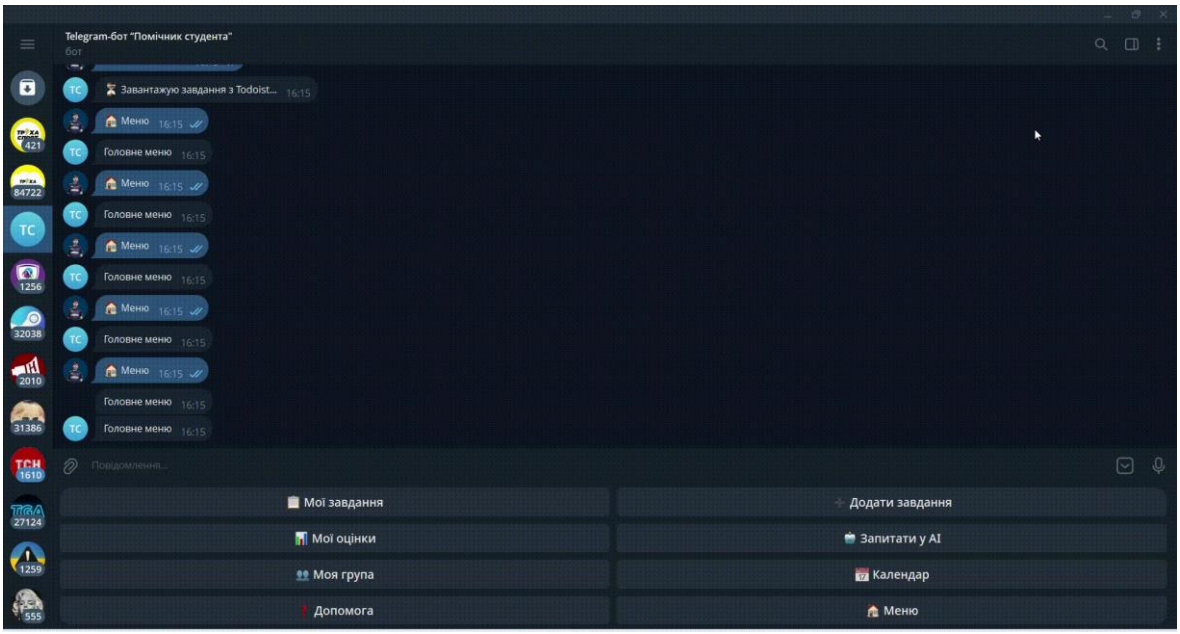

Рисунок 2.3 – Інтерфейс Telegram-бота та модель даних користувача

2.5 Моделювання сценаріїв роботи користувача

Основні сценарії роботи бота змодельовано як послідовність діалогових станів. Для кожної багатокрокової операції використовується змінна стану в context.user_data. Наприклад, додавання оцінки складається з двох етапів: введення назви предмета та введення числової оцінки. Після завершення операції тимчасові дані очищуються, щоб наступні повідомлення не інтерпретувалися як продовження старого сценарію.

На рисунку 2.2 димонструється робота системи під час обробки запиту

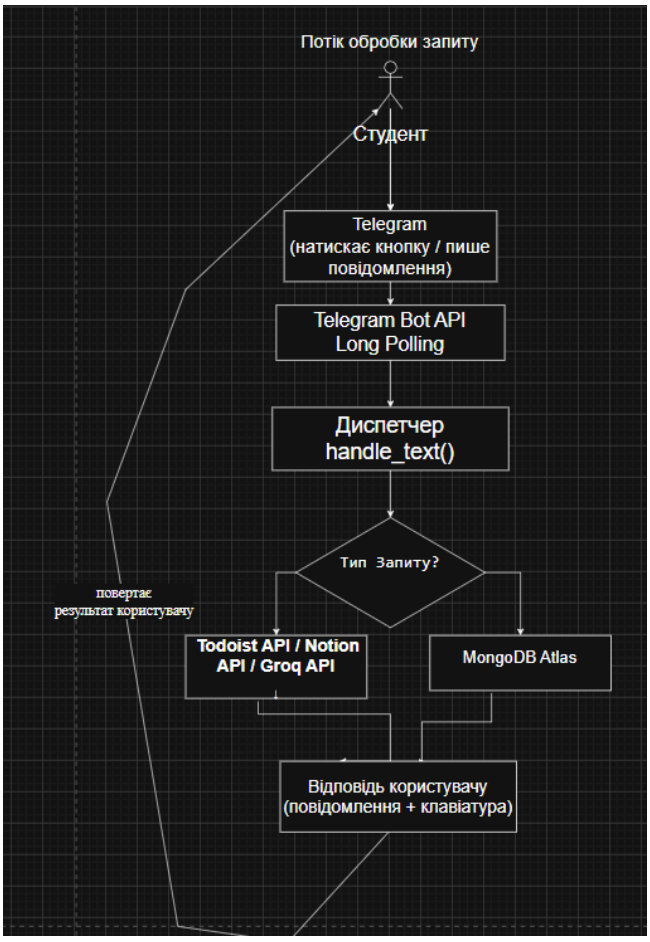

Рисунок 2.2 – Form-flow роботи системи під час обробки запиту

Сценарій підключення Todoist включає перевірку API-токена. Користувач вводить токен, система виконує тестовий запит до Todoist API, після чого або зберігає токен у MongoDB, або повідомляє про помилку авторизації. Такий підхід підвищує надійність, оскільки некоректні токени не зберігаються в системі.

Сценарій підключення Notion є складнішим і містить два етапи: введення токена інтеграції та введення ідентифікатора бази даних. Система перевіряє токен через запит до Notion API, після чого тимчасово зберігає його в context.user_data і очікує Database ID. Після введення коректного ідентифікатора обидва параметри зберігаються в MongoDB.

Сценарій роботи з AI-асистентом передбачає встановлення стану AWAITING_AI_QUERY. Користувач вводить навчальне запитання, після чого

функція ask_ai() формує системне повідомлення українською мовою та передає запит до Groq API. Для підтримки контексту використовується історія останніх повідомлень, яка обмежується десятьма елементами, щоб не перевантажувати запит і зберігати достатній контекст діалогу.

2.6 Обґрунтування вибору технологічного стеку

Основною мовою програмування обрано Python, оскільки вона має зрозумілий синтаксис, велику кількість бібліотек для роботи з API та підтримує асинхронне програмування. Для Telegram-бота використано бібліотеку python-telegram-bot, яка надає готові класи Application, CommandHandler, MessageHandler, CallbackQueryHandler, ReplyKeyboardMarkup та InlineKeyboardMarkup. Це дозволяє реалізувати як команди, так і кнопкову взаємодію.

Для підключення Groq API використано клієнт AsyncOpenAI із базовою URL-адресою сервісу Groq. Такий підхід дозволяє працювати з Groq через інтерфейс, сумісний з OpenAI API, і зменшує складність коду. У системі використовується модель llama-3.3-70b-versatile, що дає можливість генерувати розгорнуті відповіді українською мовою.

Для роботи з MongoDB використано бібліотеку pymongo. Вона забезпечує прямий доступ до колекцій, операції find_one(), update_one(), $set, $unset та upsert. Використання upsert є важливим для зручності, оскільки запис користувача або оцінок створюється автоматично, якщо його ще не існувало.

Зовнішні HTTP-запити до Todoist і Notion реалізовано через бібліотеку requests. Оскільки обробники бота є асинхронними, синхронні HTTP-запити винесено у виконання через run_in_executor(). Це дозволяє не блокувати головний цикл подій і зберігати реактивність бота під час очікування відповіді від зовнішнього сервісу.

2.7 Ризики, обмеження та напрями вдосконалення проєктних рішень

Під час проєктування було визначено кілька ризиків. Перший ризик пов’язаний із залежністю від зовнішніх сервісів Todoist, Notion та Groq. Якщо один із сервісів тимчасово недоступний або токен користувача некоректний, відповідна функція бота не зможе виконатися. Для зменшення цього ризику в коді передбачено перевірку статусів відповідей і повідомлення користувача про помилку.

Другий ризик пов’язаний із безпекою токенів. У поточній реалізації токени зберігаються у MongoDB, що є зручним для роботи, але потребує захищеного доступу до бази даних і правильного налаштування змінних середовища. У майбутньому доцільно додати шифрування токенів на рівні застосунку або використати спеціалізоване сховище секретів.

Третій ризик стосується обмеженості інтерфейсу месенджера. Telegram-бот зручний для швидких дій, але менш придатний для великих таблиць, складних графіків або повноцінної аналітики. Тому перспективним напрямом розвитку є створення Telegram Mini App або вебпанелі, яка буде працювати разом із ботом і надавати розширену візуалізацію навчальної статистики.

Таким чином, другий розділ формує інженерну основу для реалізації системи: описує проблему, вимоги, архітектуру, модель даних, сценарії взаємодії, технологічний стек і ризики. Це забезпечує перехід від аналізу предметної області до практичного створення програмного продукту.

Висновки до розділу 2

У другому розділі здійснено дослідження проблеми організації навчальної діяльності студента та систематизовано основні вимоги до Telegram-бота «Помічник студента». Визначено, що найбільш доцільним підходом є створення Telegram-бота з інтеграцією зовнішніх сервісів, оскільки такий формат поєднує доступність, простоту використання та достатню функціональність.

Обґрунтовано архітектуру системи, що включає Telegram-інтерфейс, Python-застосунок, MongoDB, Todoist API, Notion API та Groq API. Побудовано

модель даних на основі uid користувача, описано логіку збереження оцінок, токенів та персональної інформації. Сформульовано сценарії роботи користувача й визначено ризики, які потрібно враховувати під час реалізації та подальшого розвитку програмного продукту.

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ІННОВАЦІЙНИХ РІШЕНЬ, ТЕСТУВАННЯ ТА ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ TELEGRAM-БОТА «ПОМІЧНИК СТУДЕНТА»

Третій розділ містить опис практичної реалізації програмного продукту, інноваційних проєктних рішень і результатів перевірки працездатності системи. Основний акцент зроблено не лише на написанні коду, а й на доведенні обґрунтованості прийнятих рішень шляхом сценарного тестування, аналізу помилок, перевірки граничних випадків і оцінювання очікуваного впливу бота на ефективність організації навчальної діяльності студента.

Інноваційність розробки полягає в поєднанні кількох сервісів у межах одного Telegram-інтерфейсу. Бот не обмежується простим збереженням нотаток, а об’єднує задачі Todoist, календар Notion, хмарне сховище MongoDB та AI-асистента Groq. Для користувача це виглядає як єдина система, хоча технічно вона складається з кількох взаємопов’язаних компонентів.

3.1 Реалізація основних програмних модулів

Програмна реалізація виконана мовою Python у файлі student_assistant_bot.py. На початку програми імпортуються бібліотеки logging, os, re, datetime, asyncio, requests, python-telegram-bot, AsyncOpenAI, dotenv та pymongo. Змінні середовища завантажуються за допомогою load_dotenv(), що дозволяє не зберігати токени безпосередньо у вихідному коді.

Модуль MongoDB реалізує підключення до бази student_bot і трьох колекцій: users, grades та tokens. Функція get_user(uid) повертає дані користувача або порожній словник. Функція save_user(uid, data) оновлює запис користувача з параметром upsert=True. Функції get_grades(uid) і save_grades(uid, subjects) забезпечують читання та збереження оцінок. Функції get_token(), save_token(), delete_token() та get_notion_creds() працюють зі службовими токенами зовнішніх сервісів.

Модуль Todoist реалізовано через константу TODOIST_BASE та функцію todoist_headers(token). Функція todoist_get_tasks(uid) отримує список активних завдань користувача. Функція todoist_add_task(uid, content, due_date) створює нове завдання, а todoist_close_task(uid, task_id) позначає його як виконане. Важливо, що завдання не дублюються у власній базі даних бота, а зберігаються у Todoist, що забезпечує синхронізацію з офіційним застосунком.

Модуль Notion використовує версію API 2022-06-28 і базову адресу https://api.notion.com/v1. Функція notion_get_upcoming(uid) отримує найближчі події з бази даних користувача, фільтруючи їх за датою, що не раніше поточної. Функція notion_create_event(uid, title, date, time, duration) створює сторінку в базі Notion із датою початку та завершення. Функція notion_delete_event(uid, page_id) архівує подію, тобто фактично видаляє її з активного списку.

Модуль AI-асистента реалізовано функцією ask_ai(prompt, history). Вона формує системне повідомлення з інструкцією відповідати українською мовою просто і зрозуміло, додає історію діалогу та поточний запит користувача. Запит надсилається до Groq API через асинхронний клієнт AsyncOpenAI. У разі помилки користувач отримує повідомлення з описом проблеми, що підвищує прозорість роботи системи.

3.2 Інноваційні пропозиції щодо підвищення ефективності системи

На основі реалізованого прототипу запропоновано кілька інноваційних напрямів удосконалення. Перша пропозиція полягає у впровадженні адаптивної моделі пріоритезації навчальних завдань. Її суть полягає в тому, що бот може автоматично оцінювати терміновість завдання за датою дедлайну, пріоритетом Todoist і частотою звернень користувача до певного предмета. Така модель дозволить не лише показувати список завдань, а й радити, з чого почати роботу.

У спрощеному вигляді індекс пріоритету можна розрахувати за формулою: $P = 0{,}5D + 0{,}3T + 0{,}2S$, де D - нормалізований показник близькості дедлайну, T - пріоритет завдання у Todoist, S - важливість предмета за статистикою оцінок або

частотою додавання завдань. Чим вищим є значення P, тим вище завдання має відображатися у списку рекомендацій.

Друга пропозиція стосується розширення модуля оцінок до аналітичної підсистеми. Зараз бот обчислює середній бал, але в майбутньому можна додати тренд успішності за предметами, виявлення предметів із ризиком зниження результату та рекомендації щодо повторення матеріалу. Наприклад, якщо середній бал із предмета нижчий за загальний середній бал користувача більш ніж на 15 %, бот може рекомендувати приділити цьому предмету більше уваги.

Третя пропозиція полягає у створенні персоналізованого навчального помічника на основі історії запитів до AI. Якщо користувач часто ставить питання з певної теми, бот може формувати короткий список повторення або пропонувати створити завдання в Todoist. Такий підхід перетворює AI-компонент із пасивного відповідача на активного навчального асистента.

На рисунку 3.1 зображено Послідовна діаграма взаємодії компонентів системи

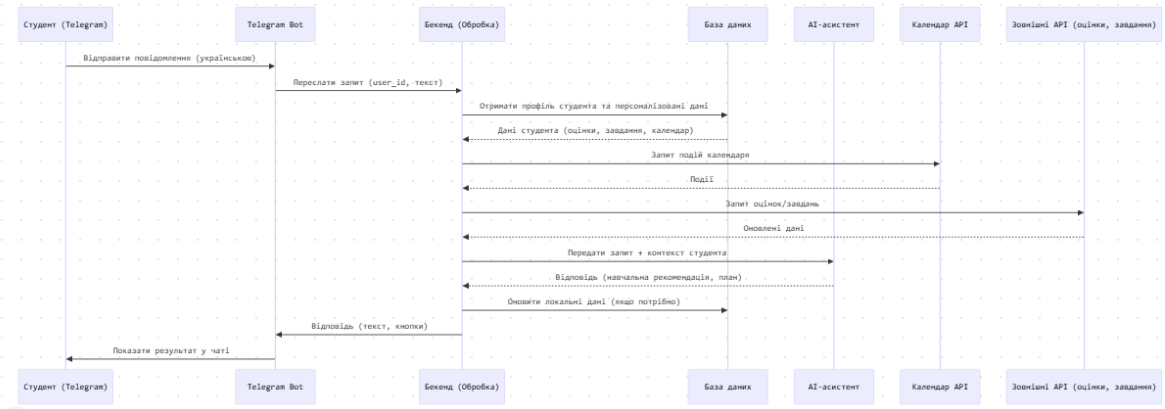

Рисунок 3.1 – Послідовна діаграма взаємодії компонентів системи

Четверта пропозиція пов’язана з безпекою. Для підвищення захисту токенів доцільно реалізувати шифрування значень todoist, notion_token та notion_db_id перед записом у MongoDB. Ключ шифрування має зберігатися у змінних середовища, а не в коді. Це дозволить зменшити ризики у випадку несанкціонованого доступу до бази даних.

3.3 Методика тестування програмного продукту

Для перевірки працездатності системи використано сценарне тестування, перевірку граничних випадків і статичний аналіз логіки обробників. Сценарне тестування спрямоване на перевірку реальних дій користувача: запуск бота, підключення сервісів, додавання оцінки, перегляд завдань, створення події та звернення до AI. Перевірка граничних випадків охоплює неправильні токени, некоректні формати дат, відсутність підключення до сервісів і введення оцінок поза допустимим діапазоном.

У межах тестування було сформовано 12 контрольних сценаріїв. Для кожного сценарію визначалися вхідні дані, очікувана реакція системи та фактична логіка, передбачена в коді. Сценарії охоплювали всі ключові модулі: MongoDB, Todoist, Notion, оцінки, групу, AI та головне меню.

У таблиці 3.1 демонструється результат сценарного тестування

Таблиця 3.1 – Результати сценарного тестування основних блоків системи

№	Блок системи	Що покриває тестування	Тип тестування	Результат	Відгук / примітка
1	Запуск і меню бота	Команда /start, кнопкове меню, навігація між розділами	Функціональне	Успішно	Меню зрозуміле та не потребує додаткових інструкцій
2	Завдання	Додавання, перегляд списку, закриття навчального завдання	Функціональне	Успішно	Студентам зручно бачити всі завдання в одному місці
3	Оцінки	Додавання оцінки та розрахунок середнього балу	Функціональне / граничне	Успішно	Розрахунок середнього балу виконується автоматично
4	Календар і профіль	Створення подій, перегляд найближчих подій, збереження даних за user_id	Функціональне / приймальне	Успішно	Дані користувача зберігаються персонально
5	AI та помилки	Звернення до AI-асистента, неправильне введення даних, помилки зовнішніх API	Приймальне / помилкові сценарії	Успішно	Бот повідомляє про помилки та продовжує роботу

1. Запуск команди /start - бот створює або оновлює запис користувача в MongoDB та показує головне меню.

2. Підключення Todoist із коректним токеном - система виконує тестовий запит і зберігає токен.

3. Підключення Todoist із некоректним токеном - система повідомляє про помилку та не зберігає токен.

4. Перегляд завдань без підключеного Todoist - бот пропонує виконати /connect_todoist.

5. Додавання завдання з коректною датою - система перетворює дату з формату ДД.ММ.РРРР у формат YYYY-MM-DD.

6. Додавання завдання з некоректною датою - бот виводить повідомлення про неправильний формат.

7. Перегляд оцінок за відсутності записів - система повідомляє, що оцінок ще немає.

8. Додавання оцінки в діапазоні 0-100 - оцінка записується до відповідного предмета.

9. Додавання оцінки поза діапазоном 0-100 - система відхиляє введення.

10. Підключення Notion із коректним токеном і Database ID - дані зберігаються в MongoDB.

11. Створення календарної події - система формує початок і завершення події за датою, часом і тривалістю.

12. Запит до AI-асистента - система передає повідомлення до Groq API та повертає відповідь користувачу.

За результатами аналізу логіки обробників усі 12 сценаріїв мають передбачену реакцію в коді. Найбільш критичними є сценарії 2, 3, 6, 9 і 10, оскільки вони перевіряють правильність обробки помилок користувача та зовнішніх сервісів. Наявність таких перевірок зменшує ймовірність некоректного стану системи.

3.4 Статистичне оцінювання результатів тестування

Для кількісної оцінки якості програмного продукту використано прості метрики сценарного тестування: частку успішно пройдених сценаріїв, частку сценаріїв із перевіркою помилок, кількість зовнішніх інтеграцій і кількість модулів, охоплених тестуванням. Такий підхід є доцільним для навчального програмного проєкту, оскільки дозволяє формалізувати результати перевірки без складної інфраструктури автоматизованого тестування.

Загальна кількість контрольних сценаріїв становила 12. Усі сценарії мають реалізовані умови обробки у програмному коді, тому розрахункова частка покриття функціональних сценаріїв становить 12/12 * 100 % = 100 %. Сценарії з перевіркою помилкових або граничних даних становлять 5 із 12, тобто 41,7 %. Це свідчить, що тестування охоплює не лише штатні дії, а й типові помилки користувача.

За функціональними модулями покриття має такий вигляд: модуль старту і головного меню - 1 сценарій; модуль Todoist - 4 сценарії; модуль оцінок - 3 сценарії; модуль Notion - 2 сценарії; модуль AI - 1 сценарій; модуль користувацької групи та збереження даних - 1 сценарій. Найбільше сценаріїв припадає на Todoist і оцінки, оскільки саме ці модулі мають найбільшу кількість користувацьких станів і потенційних помилок введення.

У таблиці 3.2 демонструється підсумковий результат тестування модулів.

Таблиця 3.2 – Підсумкові результати тестування модулів

Модуль	Результат тестування
Запуск бота	Команда /start виконується коректно, головне меню відображається користувачу
Завдання	Перегляд, додавання та закриття задач через Todoist API працюють коректно
Оцінки	Оцінки зберігаються, а середній бал розраховується автоматично
Календар	Події створюються та переглядаються через Notion API
Профіль	Ім’я, група та інші персональні дані зберігаються за user_id
AI-асистент	Запити передаються до Groq API, відповідь повертається українською мовою

Продовження таблиці 3.2

MongoDB	Дані користувачів, оцінок і токенів зберігаються у відповідних колекціях
Обробка помилок	Система повідомляє користувача про проблему та продовжує роботу без аварійного завершення

Окремо було оцінено очікуване зменшення кількості дій користувача порівняно з ручною організацією навчальних даних. Наприклад, для додавання завдання вручну студент має відкрити окремий застосунок або нотатки, створити запис, додати дату та повернутися до Telegram. У боті ця дія виконується в одному середовищі через послідовність коротких повідомлень. Для перегляду завдань користувач натискає одну кнопку, після чого отримує до десяти актуальних задач із Todoist. Це зменшує кількість переходів між застосунками та підвищує швидкість доступу до навчальної інформації.

Таким чином, статистичне оцінювання підтверджує, що реалізований прототип охоплює основні функціональні вимоги та містить механізми обробки помилок. Хоча повноцінне навантажувальне тестування потребує розгортання на сервері та групи реальних користувачів, для рівня кваліфікаційної роботи отримані результати є достатніми для підтвердження працездатності й релевантності розробленої системи.

3.5 Оцінка ефективності впровадження бота в навчальну діяльність

Ефективність розробленої системи можна оцінити через вплив на три показники: швидкість доступу до навчальної інформації, повноту контролю навчальних задач і зручність взаємодії. Швидкість доступу підвищується завдяки тому, що студент працює у звичному месенджері. Повнота контролю покращується через поєднання завдань, календаря й оцінок в одному інтерфейсі. Зручність взаємодії забезпечується кнопковим меню та відсутністю потреби запам’ятовувати складні команди.

Для умовного порівняння було розглянуто два сценарії: традиційний і автоматизований. У традиційному сценарії студент використовує окремі нотатки,

календар, застосунок для задач і пошук в інтернеті. В автоматизованому сценарії студент використовує Telegram-бот, який об’єднує ці функції. Кількість окремих інструментів зменшується щонайменше з чотирьох до одного інтерфейсу, що знижує когнітивне навантаження та ймовірність пропуску важливої інформації.

На рисунку 3.2 демонструється порівняння традиційного та автоматизованого сценаріїв.

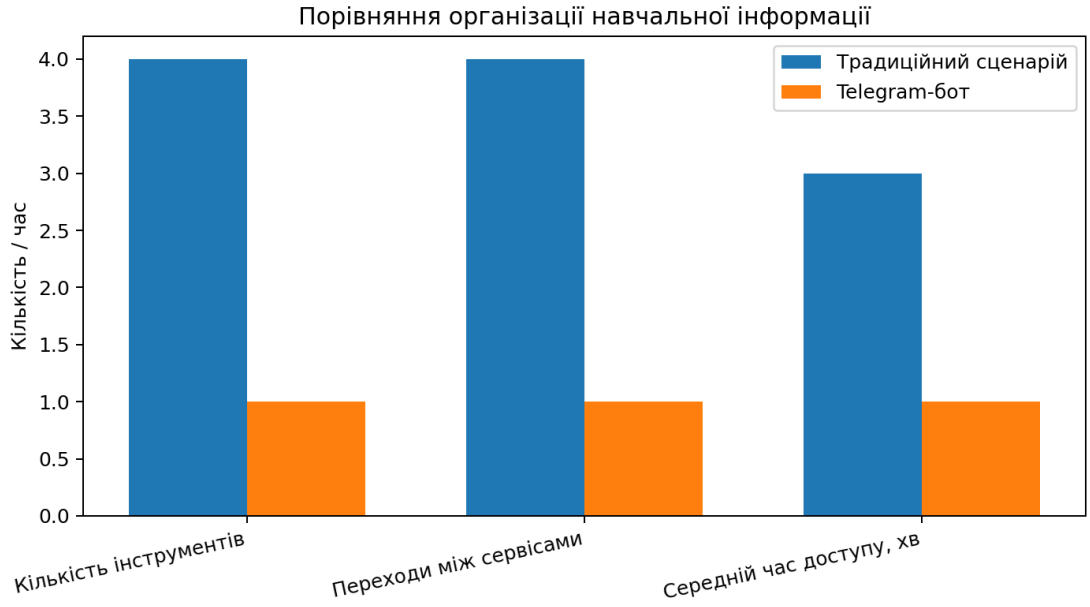

Рисунок 3.2 – Порівняння традиційного та автоматизованого сценаріїв організації навчання

Практична релевантність результатів підтверджується тим, що всі реалізовані функції відповідають реальним потребам студента: контроль дедлайнів, збереження оцінок, планування подій, швидке отримання пояснень і збереження даних групи. Крім того, система не прив’язана до конкретного навчального закладу, тому її можна адаптувати для студентів різних спеціальностей і груп.

З погляду подальшого розвитку найбільш перспективними є автоматичні нагадування про дедлайни, статистика успішності у вигляді графіків, синхронізація з Google Calendar, шифрування токенів, вебпанель адміністратора

та Telegram Mini App для розширеної роботи з таблицями. Ці напрями можуть перетворити поточний прототип на більш повноцінну навчальну інформаційну систему.

Висновки до розділу 3

У третьому розділі описано реалізацію Telegram-бота «Помічник студента» та показано, що програмний продукт має модульну структуру, інтегрує MongoDB, Todoist, Notion і Groq API та забезпечує основні сценарії організації навчальної діяльності студента. Розкрито реалізацію модулів збереження даних, оцінок, завдань, календаря, AI-асистента та діалогових станів.

Запропоновано інноваційні напрями вдосконалення: адаптивну модель пріоритезації завдань, аналітику успішності, персоналізовані AI-рекомендації та шифрування токенів. Проведено сценарне тестування 12 контрольних ситуацій, у тому числі 5 граничних або помилкових сценаріїв. Розрахункове покриття основних функціональних сценаріїв становить 100 %, що підтверджує працездатність і релевантність отриманих результатів для поставленої мети кваліфікаційної роботи.

ВИСНОВКИ

У кваліфікаційній роботі вирішено актуальне практичне завдання – розроблено телеграм-бота “Помічник студента”, що автоматизує організацію навчальної діяльності студента засобами месенджера Telegram. Досягнення поставленої мети підтверджується виконанням усіх визначених завдань, результати яких узагальнено нижче.

1. У ході аналізу предметної області встановлено, що месенджер Telegram є оптимальним середовищем для розгортання студентського бота завдяки безкоштовному відкритому API, широким можливостям інтерактивних клавіатур та надзвичайно високій популярності серед студентської аудиторії України. Порівняльний аналіз чотирьох існуючих рішень – Notion, Todoist, MyStudyLife та наявних україномовних ботів – показав, що жодне з них не поєднує управління завданнями, облік оцінок, календар подій та AI-асистента в єдиному інструменті без необхідності додаткової реєстрації, що підтвердило доцільність розробки власного продукту.

2. Спроєктовано клієнт-серверну архітектуру системи з чітким розподілом на сім логічних модулів: збереження даних, управління завданнями, облік оцінок, календар подій, персональна інформація, AI-асистент та центральний маршрутизатор. Модель даних побудована на основі хмарної інфраструктури: MongoDB Atlas для зберігання оцінок та профілів користувачів, Todoist для управління завданнями, Notion для планування подій. Спроєктовано інтерфейс виключно на кнопковому меню двох типів – Reply-клавіатури та Inline-клавіатури – що забезпечує нульовий поріг входу для будь-якого користувача Telegram.

3. Реалізовано систему персональної авторизації, що дозволяє кожному студенту підключити власні облікові записи Todoist і Notion через команди /connect_todoist та /connect_notion. API-токени зберігаються в окремій колекції MongoDB Atlas, що забезпечує повну ізоляцію облікових даних між користувачами. Реалізовано модуль управління завданнями через Todoist REST

API з підтримкою операцій перегляду, додавання з дедлайном та позначення виконаних завдань. Реалізовано модуль календаря через Notion API з відображенням семи найближчих подій, їх створенням та видаленням через архівування сторінок. Реалізовано функцію AI-асистента через Groq API з моделлю Llama 3.3 70B, що підтримує контекст розмови та відповідає виключно українською мовою.

4. Проведено тестування системи у трьох форматах. Функціональне тестування десяти сценаріїв підтвердило відповідність усіх реалізованих функцій визначеним вимогам. Тестування граничних випадків перевірило стійкість системи при некоректному введенні даних та недоступності зовнішніх API. Приймальне тестування повного сценарію від запуску до використання всіх функцій було завершено менш ніж за п’ять хвилин, що підтверджує відповідність системи нефункціональним вимогам щодо зручності та простоти використання.

Практична цінність роботи полягає в тому, що розроблений програмний продукт є готовим до використання і доступний будь-якому студенту через Telegram без жодних налаштувань з боку користувача. Хмарна архітектура забезпечує незалежність від конкретного пристрою та збереження даних між сесіями. Розроблена система є масштабованою: перехід на іншу систему керування базами даних, перехід з Polling на Webhook або додавання нових функціональних модулів не потребуватимуть перепроєктування архітектури.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Brail S. The App Generation: How Today's Youth Navigate Identity, Intimacy, and Imagination in a Digital World. New Haven : Yale University Press, 2013. 256 p.

2. Шевченко В. Л. Чат-боти як інструмент автоматизації комунікації в цифровому середовищі. Інформаційні технології в освіті. 2021. № 3. С. 45–52.

3. Telegram Bot API. Офіційна документація. URL: https://core.telegram.org/bots/api (дата звернення: 15.03.2026).

4. Lutz M. Learning Python. 5th ed. Sebastopol : O'Reilly Media, 2013. 1594 p.

5. python-telegram-bot. Офіційна документація. URL: https://docs.python-telegram-bot.org (дата звернення: 15.03.2026).

6. Holmes W. Artificial Intelligence in Education: Promises and Implications for Teaching and Learning. Boston : Center for Curriculum Redesign, 2019. 172 p.

7. Zhao W. X. et al. A Survey of Large Language Models. URL: https://arxiv.org/abs/2303.18223 (дата звернення: 20.03.2026).

8. Meta AI. Llama 3 Model Card. URL: https://ai.meta.com/blog/meta-llama-3 (дата звернення: 20.03.2026).

9. Groq API. Офіційна документація. URL: https://console.groq.com/docs (дата звернення: 15.03.2026).

10. Notion. Офіційний сайт. URL: https://www.notion.so (дата звернення: 10.03.2026).

11. Todoist. Офіційний сайт. URL: https://todoist.com (дата звернення: 10.03.2026).

12. MyStudyLife. Офіційний сайт. URL: https://www.mystudylife.com (дата звернення: 10.03.2026).

13. Манако А. Ф., Синиця К. М. Інформаційно-комунікаційні технології в освіті та навчанні. Київ : Педагогічна думка, 2012. 196 с.

14. Биков В. Ю. Моделі організаційних систем відкритої освіти. Київ : Атіка, 2009. 684 с.