ПОЛТАВСЬКИЙ ДЕРЖАВНИЙ АГРАРНИЙ УНІВЕРСИТЕТ
Навчально-науковий інститут економіки, управління, права та інформаційних технологій
Кафедра інформаційних систем та технологій

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти бакалавр

на тему: **«Проєктування інформаційної системи підтримки телеграм-боту»**

Виконав: здобувач вищої освіти
за освітньою програмою
Інформаційні управляючі системи
спеціальності 126 Інформаційні
системи та технології
ступеня вищої освіти бакалавр
групи 126ІСТ_бд_2022
Петрик Артур Андрійович
Керівник: Протас Надія Михайлівна
Рецензент: Муравльов Володимир
Вячеславович

Полтава – 2026 року

ПОЛТАВСЬКИЙ ДЕРЖАВНИЙ АГРАРНИЙ УНІВЕРСИТЕТ
Навчально-науковий інститут економіки, управління, права та інформаційних технологій
Кафедра інформаційних систем та технологій

Освітня програма Інформаційні управляючі системи
Спеціальність 126 Інформаційні системи та технології
Рівень вищої освіти перший (бакалаврський)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Юрій УТКІН
«10» червня 2025 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Петрику Артуру Андрійовичу

1. Тема роботи:
«Проєктування інформаційної системи підтримки телеграм-боту»,
керівник роботи к.с.-г.н., доцент, доцент кафедри інформаційних систем та технологій Протас Надія Михайлівна.
Затверджено наказом закладу вищої освіти від «10» квітня 2026 року № 383 ст

2. Строк подання здобувачем вищої освіти роботи «08» червня 2026 року

3. Вихідні дані до роботи: стандарти проєктування та розробки програмного забезпечення, агрономічні довідники та рекомендації виробників засобів захисту рослин, дані офіційних сайтів https://www.python.org/, https://pytba.readthedocs.io/ (бібліотека pyTelegramBotAPI), https://core.telegram.org/bots/api (Telegram Bot API), https://dev.mysql.com/ (MySQL), https://apscheduler.readthedocs.io/ (APScheduler), середовища прикладних програм PHPMyAdmin, OpenServer.

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):

Розділ 1. Аналіз предметної області, існуючих рішень та засобів розробки інформаційної системи підтримки Telegram-боту для городників
Розділ 2. Проєктування інформаційної системи підтримки Telegram-боту на основі UML-моделювання та реляційної бази даних
Розділ 3. Реалізація та оцінка ефективності інформаційної системи підтримки Telegram-боту для городників

5. Перелік графічного матеріалу: схеми, рисунки, графіки, діаграми за темою та об’єктом дослідження.

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання отримав
Оцінювання економічної ефективності результатів дослідження	Дивнич О. Д., к. е. н., доцент, доцент кафедри економіки та публічного управління	01.04.2026 р.	29.05.2026 р.

7. Дата видачі завдання «10» червня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Вибір і затвердження теми роботи	02.06.2025 р. – 04.06.2025	
2	Складання і затвердження розгорнутого плану та завдання на кваліфікаційну роботу	05.06.2025 р. – 10.06.2025 р.	
3	Опрацювання джерел інформації	11.06.2025 р. – 15.06.2025 р.	
4	Збір, вивчення і обробка інформації, необхідної для виконання роботи	16.06.2025 р. – 20.06.2025 р.	
5	Виконання теоретико-методологічного розділу роботи	08.09.2025 р. – 01.11.2025 р.	
6	Виконання дослідницько-аналітичного розділу роботи	19.01.2026 р. – 14.02.2026 р.	
7	Виконання проєктно-рекомендаційного розділу роботи	16.02.2026 р. – 31.03.2026 р.	
8	Розрахунок економічної ефективності результатів дослідження	01.04.2026 р. – 29.05.2026 р.	
9	Оформлення тексту роботи	01.06.2026 р. – 06.06.2026р.	
10	Попередній захист роботи на кафедрі	08.06.2026 р.	
11	Доопрацювання роботи з урахуванням зауважень і пропозицій	09.06.2026 р. – 10.06.2026 р.	
12	Нормоконтроль	11.06.2026 р. – 15.06.2026 р.	
13	Захист кваліфікаційної роботи	16.06.2026 р. - 18.06.2026 р.	

Здобувач вищої освіти **Артур ПЕТРИК**

Керівник роботи **Надія ПРОТАС**

ЗМІСТ

ВСТУП

Актуальність. Сучасний розвиток інформаційних технологій охоплює дедалі ширше коло повсякденних потреб людини, зокрема у сфері ведення особистого господарства. Городництво та дачне господарство залишаються поширеною практикою серед українців: понад 60% домогосподарств у сільській місцевості та значна частка міських мешканців займаються вирощуванням овочів і фруктів для власних потреб [1]. Водночас більшість городників стикаються зі спільною проблемою – браком структурованої, своєчасної та практично придатної агрономічної інформації. Пошук рекомендацій щодо підживлення, захисту рослин або строків посіву розпорошений між книгами, вебсайтами та соціальними мережами, що ускладнює прийняття рішень безпосередньо на городі [2].

Месенджер Telegram, аудиторія якого в Україні стабільно зростає і охоплює всі вікові групи [5], є зручною платформою для розгортання інтелектуальних чат-ботів. На відміну від мобільних застосунків, боти не потребують встановлення та оновлення, що робить їх особливо доступними для широкої аудиторії городників. Аналіз існуючих цифрових рішень у цій предметній області показав, що жоден із наявних аналогів не поєднує персоналізований агрономічний календар, автоматичні нагадування та структуровані рекомендації щодо засобів захисту рослин в єдиній системі [9, 10]. Це підтверджує актуальність розробки інформаційної системи підтримки Telegram-боту, орієнтованої на потреби українських городників і дачників.

Метою кваліфікаційної роботи є проєктування інформаційної системи підтримки Telegram-боту для городників, що забезпечує персоналізоване агрономічне супроводження вирощування городніх культур на основі реляційної бази даних та автоматизованого планування нагадувань.

Відповідно до мети роботи необхідно вирішити наступні *завдання:*

1. Провести аналіз предметної галузі ведення особистого городнього господарства, розглянути існуючі цифрові аналоги та обґрунтувати вибір технологічного стеку розробки;

2. Спроєктувати архітектуру інформаційної системи, включаючи UML-діаграми поведінки, реляційну базу даних із восьми таблиць та трирівневу програмну архітектуру взаємодії Telegram-боту з інформаційною системою;

3. Реалізувати інформаційну систему у вигляді програмного коду на Python із підключенням до MySQL, наповнити базу тестовими агрономічними даними та виконати техніко-економічне обґрунтування ефективності розробленої системи.

Об'єкт дослідження – процеси проєктування інформаційних систем підтримки чат-ботів із реляційними базами даних та автоматизованим плануванням подій.

Предмет дослідження – проєктування та реалізація інформаційної системи підтримки Telegram-боту для агрономічного супроводження городніх культур з використанням Python, бібліотеки pyTelegramBotAPI та СКБД MySQL.

Методи досліджень – дослідження базуються на методах програмної інженерії, аналізу та специфікації вимог із застосуванням методу пріоритизації MoSCoW, об'єктно-орієнтованого моделювання засобами UML, проєктування реляційних баз даних із нормалізацією до третьої нормальної форми, модульної розробки програмного забезпечення мовою Python, а також техніко-економічного аналізу ефективності ІТ-проєктів.

Інформаційна база – наукові статті та монографії з програмної інженерії та проєктування інформаційних систем, офіційна документація Telegram Bot API, бібліотек pyTelegramBotAPI та APScheduler, документація MySQL, агрономічні довідники та рекомендації виробників засобів захисту рослин, статистичні дані щодо аудиторії Telegram, а також стандарти розробки програмного забезпечення.

Практична значущість – розроблена інформаційна система забезпечує городникам зручний доступ до персоналізованих агрономічних рекомендацій безпосередньо через Telegram без встановлення додаткового програмного забезпечення. Спроєктована нормалізована база даних із восьми таблиць охоплює вісім городніх культур, 34 фази вегетації та 62 агрономічні операції з прив'язкою до 15 препаратів і добрив. Модульна архітектура системи та

розширювана структура бази даних спрощують додавання нових культур і функцій без зміни існуючого коду, що робить систему придатною для подальшого розвитку та масштабування.

Результати роботи апробовані в рамках наукової конференції здобувачів вищої освіти за результатами науково-дослідної роботи у 2025-2026 рр.

Структура кваліфікаційної роботи логічно пов’язана з задачами досліджень і містить перелік умовних позначень, вступ, три розділи основної частини, висновки, список використаних джерел. Загальний обсяг текстової частини кваліфікаційної роботи складає 55 сторінок формату А4. Вона містить 16 рисунків і 11 таблиць.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ, ІСНУЮЧИХ РІШЕНЬ ТА ЗАСОБІВ РОЗРОБКИ ІНФОРМАЦІЙНОЇ СИСТЕМИ ПІДТРИМКИ TELEGRAM-БОТУ ДЛЯ ГОРОДНИКІВ

1.1 Характеристика предметної області та огляд потреб користувачів городнього господарства

Сучасний етап розвитку інформаційних технологій охоплює дедалі ширше коло повсякденних потреб людини, зокрема у сфері ведення особистого господарства. Городництво та дачне господарство залишаються поширеною практикою серед українців: за даними Державної служби статистики України, понад 60% домогосподарств у сільській місцевості та значна частка міських мешканців займаються вирощуванням овочів і фруктів для власних потреб [1]. Водночас більшість дачників стикаються зі спільною проблемою – браком структурованої, своєчасної та практично придатної агрономічної інформації. Пошук рекомендацій щодо підживлення, захисту рослин або календарних строків посіву розпорошений між книгами, вебсайтами та соціальними мережами, що ускладнює прийняття рішень безпосередньо на городі [2].

Чат-боти у месенджерах стають ефективним інструментом вирішення цього протиріччя. Бот – це програмне забезпечення, що виконує автоматизовані завдання й спілкується з користувачем через інтерфейс обміну повідомленнями [3]. На відміну від мобільних застосунків, боти не потребують встановлення, оновлення або значних апаратних ресурсів пристрою, що робить їх особливо зручними для користувачів середнього та старшого віку – саме тієї аудиторії, яка найактивніше займається дачним господарством.

Серед усіх месенджерів Telegram вирізняється відкритим API для розробників, багатим функціоналом (кнопки, меню, вбудовані клавіатури) та високим рівнем захисту даних користувачів [4]. Згідно зі статистикою TGStat,

аудиторія Telegram в Україні стабільно зростає і охоплює всі вікові групи, що підтверджує універсальність платформи як середовища для розгортання прикладних інформаційних систем [5]. Розподіл користувачів Telegram за віком наведено на рисунку 1.1.

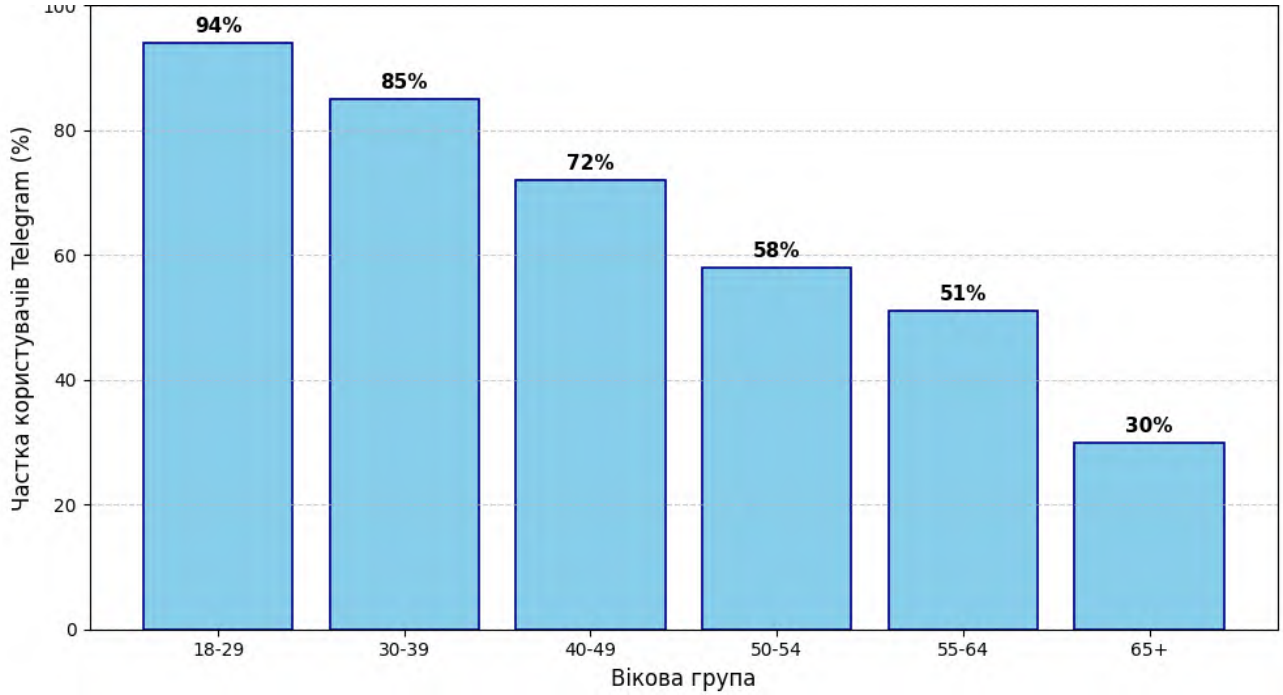

Рисунок 1.1 – Розподіл користувачів Telegram в Україні за віковими групами

Ключовою перевагою Telegram як платформи є можливість реалізації кнопкових та текстових ботів без складної серверної інфраструктури. Кнопковий бот взаємодіє з користувачем через заздалегідь визначені сценарії та меню вибору, що суттєво спрощує розробку і підвищує передбачуваність поведінки системи [3]. Саме такий підхід є оптимальним для агрономічного бота-помічника, де переважна більшість запитів має структурований характер: вибір культури, перегляд календаря операцій, отримання рекомендацій щодо препаратів.

Предметна область системи охоплює агрономічний цикл типових городніх культур – помідора, огірка, перцю, картоплі, цибулі, моркви, кабачка та буряка. Для кожної культури виділяють кілька ключових фаз вегетації: сходи, розвиток листкового апарату, цвітіння, плодоношення та дозрівання. На кожній фазі

передбачено певний перелік агрономічних операцій – полив, внесення добрив, обробка засобами захисту рослин (ЗЗР). Типова класифікація операцій за фазами вегетації для основних культур наведена у таблиці 1.1.

Таблиця 1.1 – Агрономічні операції за фазами вегетації основних городніх культур

Культура	Фаза вегетації	Типові операції
Помідор	Сходи	Полив, азотне підживлення
Помідор	Цвітіння	Фосфорне підживлення, обробка від фітофтори
Огірок	Розвиток листя	Полив, комплексне підживлення
Огірок	Плодоношення	Обробка від павутинного кліща
Картопля	Сходи	Обробка від колорадського жука
Картопля	Цвітіння	Калійне підживлення, фунгіцид
Цибуля	Розвиток листя	Азотне підживлення, полив
Морква	Проріджування	Мінімальний полив, боротьба з бур'янами

Важливим аспектом є дозування препаратів. Виробники агрохімікатів, зокрема компанія Syngenta, публікують рекомендовані норми внесення діючих речовин з розрахунку на одиницю площі [6]. Для невеликих городніх ділянок доцільно перераховувати норми з гектарів на квадратні метри за формулою:

$$D_{\text{м}^2} = \frac{D_{\text{га}}}{10000}, \tag{1.1}$$

де $D_{\text{га}}$ – рекомендована доза препарату на гектар (л/га або кг/га),

$D_{\text{м}^2}$ – доза на один квадратний метр площі.

Часовий розрахунок нагадувань у системі базується на даті посіву або висадки розсади, яку вводить користувач. Від цієї дати система обчислює орієнтовні строки настання кожної фази вегетації та формує календар нагадувань. Кількість діб від посіву до початку фази T визначається як:

$$T = T_0 + \sum_{i=1}^{n} \Delta t_i, \tag{1.2}$$

де T_0 – дата посіву у форматі порядкового дня року,

Δt_i – тривалість i-ї фази вегетації у добах,

n – номер фази, до якої виконується розрахунок.

Таким чином, предметна область інформаційної системи підтримки Telegram-боту для городників охоплює структуровані агрономічні знання, прив'язані до конкретних культур, фаз їх розвитку та календарних строків [7]. Практична цінність системи полягає у забезпеченні користувача своєчасними, персоналізованими рекомендаціями безпосередньо через звичний інтерфейс месенджера, без необхідності встановлювати додаткове програмне забезпечення.

1.2 Огляд існуючих аналогів та їх порівняльний аналіз

Перед проєктуванням будь-якої інформаційної системи необхідно провести аналіз існуючих рішень у відповідній предметній області. Це дозволяє виявити сильні та слабкі сторони наявних продуктів, уникнути повторення їхніх недоліків та визначити функціональні можливості, які забезпечать конкурентну перевагу розроблюваної системи [9]. У сфері цифрових помічників для городників і дачників існує кілька категорій рішень: мобільні застосунки, вебсервіси та Telegram-боти. Розглянемо найбільш характерних представників кожної категорії.

Вебсервіс Agronom.guru – україномовний портал із детальними агрономічними довідниками, таблицями сумісності рослин та рекомендаціями щодо захисту від шкідників. Ресурс містить якісний контент, проте є виключно інформаційним: він не надсилає нагадувань, не веде персонального календаря і потребує самостійного пошуку інформації користувачем [11]. Взаємодія зводиться до перегляду статей, що є недостатнім для активного супроводу городнього сезону.

Мобільний застосунок «Дім. Сад. Город» (Android/iOS) орієнтований на україномовну аудиторію та містить агрокалендар із рекомендаціями щодо посіву

і догляду за основними городніми культурами. Застосунок має зручний інтерфейс, проте вимагає встановлення та регулярних оновлень, не підтримує персоналізованих нагадувань і не враховує дату висадки конкретного користувача – всі рекомендації подаються в загальному вигляді за місяцями [10]. Інтерфейс застосунку наведено на рисунку 1.2.

Рисунок 1.2 – Інтерфейс мобільного застосунку «Дім. Сад. Город»

Серед Telegram-ботів варто відзначити бота «Садівник UA», який надає базові довідки про рослини за ключовими словами та посилання на зовнішні ресурси. Головним недоліком є відсутність власної структурованої бази даних – усі відповіді ведуть на сторонні сайти, що знижує оперативність і зручність користування [12]. Аналогічну проблему має Telegram-бот «АгроПорадник»: він пропонує загальні поради щодо догляду за рослинами, але не підтримує персоналізацію, не дозволяє зазначити конкретні культури на ділянці користувача і не формує індивідуального календаря нагадувань.

Окремої уваги заслуговує застосунок Gardenize (Швеція), доступний для iOS та Android і орієнтований на міжнародну аудиторію [13]. Він дозволяє вести щоденник городника, фотографувати посіви, відстежувати прогрес вирощування

та зберігати нотатки. Однак застосунок не містить агрохімічних рекомендацій, не підтримує українську мову та вимагає реєстрації облікового запису. Функція нагадувань присутня, але прив'язана виключно до дат, які вводить сам користувач, без автоматичного розрахунку фаз вегетації. Узагальнений порівняльний аналіз розглянутих аналогів та розроблюваної системи наведено у таблиці 1.2.

Таблиця 1.2 – Порівняльний аналіз існуючих аналогів та розроблюваної системи

Критерій	«Дім. Сад. Город»	Agronom.guru	«Садівник UA»	Gardenize	Розроблювана система
Платформа	iOS/Android	Веб	Telegram	iOS/Android	Telegram
Українська мова	Так	Так	Так	Ні	Так
Встановлення не потрібне	Ні	Так	Так	Ні	Так
Персональний календар	Ні	Ні	Ні	Частково	Так
Автоматичні нагадування	Ні	Ні	Ні	Ні	Так
Рекомендації щодо ЗЗР	Ні	Так	Ні	Ні	Так
Власна база даних	Так	Так	Ні	Так	Так
Розрахунок фаз вегетації	Ні	Ні	Ні	Ні	Так

Як видно з таблиці 1.2, жоден із розглянутих аналогів не поєднує в собі одночасно доступність через месенджер, персоналізований агрономічний календар із автоматичним розрахунком фаз вегетації та рекомендації щодо засобів захисту рослин. Саме це поєднання є ключовою відмінністю та перевагою розроблюваної системи.

Для кількісної оцінки функціональної повноти аналогів доцільно використати зважений показник покриття вимог. Якщо позначити множину функціональних вимог як $F = \{f_1, f_2, \dots, f_n\}$, а множину реалізованих функцій

конкретного аналога як $R \subseteq F$, то коефіцієнт функціональної повноти K_{FP} визначається як:

$$K_{FP} = \frac{|R|}{|F|} \times 100\%, \quad (1.3)$$

де $|R|$ – кількість реалізованих функцій аналога,

$|F|$ – загальна кількість визначених функціональних вимог до системи.

Для розроблюваної системи із семи ключових вимог (наведених у таблиці 1.2 як рядки) жоден аналог не реалізує більше трьох. Таким чином, максимальне значення K_{FP} серед аналогів не перевищує:

$$K_{FP}^{max} = \frac{3}{7} \times 100\% \approx 42{,}9\%,$$

тоді як розроблювана система покриває всі сім вимог, забезпечуючи $K_{FP} = 100\%$.

Проведений аналіз підтверджує актуальність і практичну цінність розроблюваної інформаційної системи. Існуючі рішення або обмежені за функціональністю, або недоступні без встановлення окремого застосунку, або не адаптовані до українських реалій городництва [14]. Telegram-бот із власною реляційною базою даних, персоналізованим календарем і автоматичними нагадуваннями заповнює цю нішу і відповідає потребам цільової аудиторії.

1.3 Вибір та обґрунтування технологічного стеку розробки

Вибір технологічного стеку є одним із ключових рішень на етапі проєктування інформаційної системи, оскільки він визначає не лише технічні можливості продукту, а й трудомісткість розробки, зручність підтримки та масштабованість у майбутньому [15]. При виборі інструментів для реалізації

Telegram-боту агрономічного спрямування враховувались такі критерії: доступність на офісному ноутбуці з обмеженими апаратними ресурсами, наявність безкоштовних інструментів розробки, зрілість екосистеми бібліотек, простота інтеграції з Telegram Bot API та достатність функціональних можливостей для реалізації визначених вимог.

Мовою програмування для реалізації серверної логіки бота обрано Python версії 3.x. Python є однією з найпоширеніших мов у розробці чат-ботів і систем автоматизації завдяки лаконічному синтаксису, широкому спектру бібліотек та активній спільноті розробників [16]. Для взаємодії з Telegram Bot API використовується бібліотека pyTelegramBotAPI (telebot) – легковага та добре задокументована бібліотека, яка реалізує методи для обробки вхідних повідомлень, керування клавіатурами та відправки різних типів контенту [17]. Альтернативою міг слугувати фреймворк python-telegram-bot, однак для задач бакалаврської роботи telebot є достатнім і простішим у освоєнні. Середовищем розробки обрано Visual Studio Code – безкоштовний редактор коду з підтримкою Python, вбудованим терміналом та розширеннями для роботи з базами даних, який вже встановлено на робочому ноутбуці.

Як систему управління базами даних обрано MySQL у складі пакету OpenServer, який також наявний на робочій машині розробника. MySQL є зрілою реляційною СКБД із розвиненою підтримкою транзакцій, індексування та складних запитів [18]. Порівняно з SQLite, яка зберігає дані в одному файлі та не підтримує повноцінний багатокористувацький доступ, MySQL є більш придатною для демонстрації навичок проєктування реляційної бази даних у контексті даної роботи: вона дозволяє наочно відобразити зв'язки між таблицями, нормалізацію та використання зовнішніх ключів. Для взаємодії Python-коду з MySQL використовується бібліотека mysql-connector-python, яка забезпечує стабільне з'єднання та підтримує параметризовані запити, що унеможливлює SQL-ін'єкції [19].

Порівняльну характеристику розглянутих варіантів СКБД наведено у таблиці 1.3.

Таблиця 1.3 – Порівняння СКБД для використання в інформаційній системі

Критерій	SQLite	MySQL	PostgreSQL
Тип зберігання	Файловий	Серверний	Серверний
Багатокористувацький доступ	Обмежений	Так	Так
Підтримка зовнішніх ключів	Часткова	Повна	Повна
Складність налаштування	Мінімальна	Середня	Висока
Наявність у складі OpenServer	Ні	Так	Ні
Придатність для демонстрації	Низька	Висока	Висока

Як видно з таблиці 1.3, MySQL є оптимальним вибором з огляду на вже наявне програмне забезпечення та вимоги до демонстрації повноцінної реляційної моделі даних. PostgreSQL, попри свої переваги, потребує окремого встановлення та налаштування, що ускладнює розробку на наявній апаратній базі. Загальна архітектура технологічного стеку системи наведена на рисунку 1.3.

Рисунок 1.3 – Технологічний стек розроблюваної інформаційної системи

Для реалізації функції автоматичних нагадувань використовується бібліотека APScheduler, яка дозволяє планувати виконання функцій Python за розкладом без залучення зовнішніх сервісів [20]. Бот запускається локально або

на доступному сервері, перевіряє заплановані нагадування з визначеною частотою і надсилає повідомлення користувачам через Telegram Bot API.

Ключовим показником ефективності обраного стеку є час відгуку системи на запит користувача. Загальний час відгуку T_{resp} складається з часу обробки запиту ботом T_{bot}, часу виконання запиту до бази даних T_{db} та затримки мережі Telegram API T_{net}:

$$T_{resp} = T_{bot} + T_{db} + T_{net}. \quad (1.4)$$

За умови локального розгортання та оптимізованих індексів у MySQL значення T_{db} для типових запитів до таблиць із кількістю записів до 10 000 рядків не перевищує 10-15 мс, що забезпечує загальний час відгуку системи в межах 300-500 мс з урахуванням затримки Telegram API [19].

Вибраний технологічний стек повністю задовольняє вимоги до розроблюваної системи: він є безкоштовним, доступним на наявному обладнанні, достатньо простим для реалізації та водночас демонструє практичні навички роботи з реляційними базами даних [15], серверною логікою та зовнішніми API. Поєднання Python, telebot, MySQL та APScheduler утворює цілісний і збалансований стек, здатний забезпечити всі визначені функціональні вимоги до системи.

1.4 Формування вимог до інформаційної системи

Постановка задачі є завершальним етапом аналітичного розділу і слугує формальною основою для подальшого проєктування системи. Вона узагальнює результати аналізу предметної області, огляду аналогів та вибору технологій, перетворюючи їх на чітко сформульований перелік вимог до розроблюваного продукту [23]. Коректно сформульована постановка задачі унеможливлює

розбіжності між очікуваннями замовника і результатом розробки, а також визначає критерії оцінювання готовності системи.

Метою розроблюваної інформаційної системи є створення Telegram-боту, який надає користувачу – дачнику або городнику – персоналізовані агрономічні рекомендації щодо догляду за городніми культурами, формує індивідуальний календар агрономічних операцій на основі дати висадки та надсилає своєчасні нагадування. Система має функціонувати автономно, без участі оператора, спираючись виключно на структуровану базу агрономічних даних та введену користувачем інформацію про власну ділянку.

Функціональні вимоги описують конкретні дії, які система повинна виконувати [24]. До функціональних вимог розроблюваної системи належать:

– реєстрація користувача через Telegram без окремого облікового запису;

– вибір городніх культур із заздалегідь визначеного переліку (помідор, огірок, перець, картопля, цибуля, морква, кабачок, буряк);

– введення дати висадки або посіву для кожної обраної культури;

– автоматичний розрахунок календаря агрономічних операцій на основі дати висадки та нормативних тривалостей фаз вегетації;

– перегляд рекомендацій щодо поливу, внесення добрив та обробки засобами захисту рослин для поточної фази вегетації;

– отримання інформації про препарати: назва, тип (добриво або ЗЗР), рекомендована доза на квадратний метр;

– налаштування автоматичних нагадувань про заплановані агрономічні операції;

– перегляд та редагування переліку культур на ділянці користувача;

– отримання довідкової інформації про культури та шкідників.

Нефункціональні вимоги визначають якісні характеристики системи – те, як вона повинна працювати, а не що робити [24]. Для розроблюваної системи нефункціональні вимоги є такими: час відгуку на запит користувача не повинен перевищувати 1 секунди за умов стабільного інтернет-з'єднання; інтерфейс має бути повністю україномовним і реалізований через кнопкове меню без

необхідності введення команд вручну; система повинна коректно обробляти одночасні запити від щонайменше 10 користувачів; зберігання персональних даних має відповідати принципу мінімальності – фіксується лише ідентифікатор користувача Telegram та введені агрономічні дані.

Для наочного відображення взаємодії між користувачем і системою використовується діаграма варіантів використання (Use Case Diagram), яку буде детально розроблено у другому розділі. На рисунку 1.4 наведено загальну концептуальну схему функціонування системи.

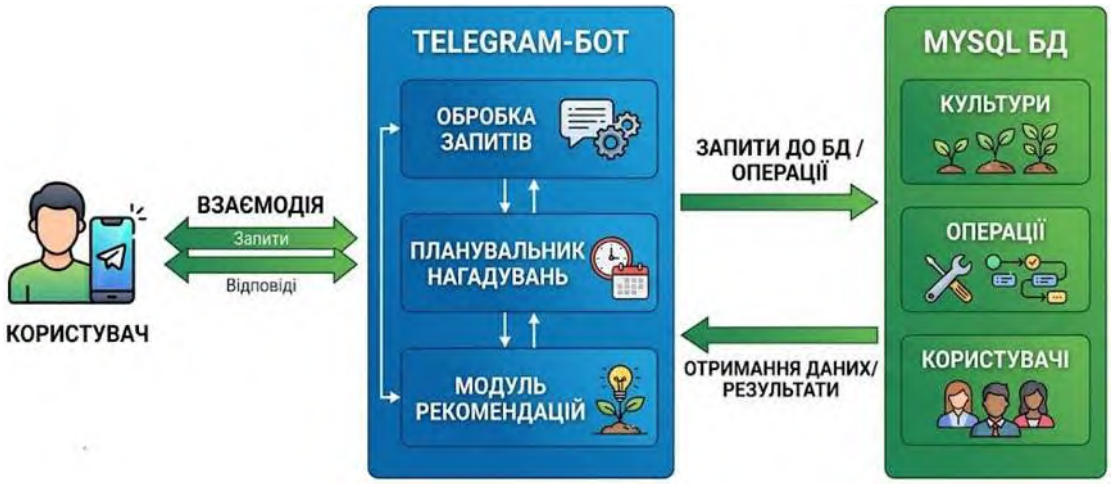

Рисунок 1.4 – Концептуальна схема функціонування інформаційної системи

Перелік сутностей бази даних, визначених на етапі постановки задачі, наведено у таблиці 1.4.

Таблиця 1.4 – Основні сутності бази даних інформаційної системи

Сутність	Призначення	Ключові атрибути
cultures	Перелік городніх культур	id, name, description
growth_phases	Фази вегетації культур	id, culture_id, phase_name, duration_days
operations	Агрономічні операції	id, phase_id, operation_type, description
products	Препарати та добрива	id, name, type, dose_per_sqm, unit
operation_products	Зв'язок операцій і препаратів	operation_id, product_id
users	Дані користувачів	id, telegram_id, registered_at
user_plantings	Культури на ділянці користувача	id, user_id, culture_id, planting_date
reminders	Заплановані нагадування	id, user_planting_id, operation_id, remind_at, is_sent

Як видно з таблиці 1.4, база даних містить вісім сутностей, що охоплюють повний цикл від агрономічних знань до персоналізованих нагадувань конкретного користувача. Нормалізована структура з явними зовнішніми ключами забезпечує цілісність даних та гнучкість при додаванні нових культур або операцій без зміни структури таблиць.

Важливим елементом постановки задачі є визначення меж системи – чітке окреслення того, що система робить і що виходить за рамки її відповідальності. Розроблювана система не є медичним або юридичним консультантом, не надає гарантій врожайності та не замінює професійного агронома. Вона виступає інформаційним помічником, що систематизує загальнодоступні агрономічні знання і подає їх у зручному персоналізованому форматі.

Пріоритизацію функціональних вимог виконано за методом MoSCoW, який поділяє вимоги на чотири категорії: Must have – обов'язкові, Should have – бажані, Could have – можливі, Won't have – поза межами поточної версії [25]. Індекс пріоритету вимоги P_i у спрощеному числовому вираженні визначається як:

$$P_i = \frac{w_i \cdot v_i}{\sum_{j=1}^{n} w_j \cdot v_j} \times 100\%, \tag{1.5}$$

де w_i – вага категорії MoSCoW (Must=4, Should=3, Could=2, Won't=1),

v_i – оцінка цінності вимоги для користувача за шкалою від 1 до 5,

n – загальна кількість вимог.

Застосування методу MoSCoW дозволяє сконцентрувати зусилля розробки на критично важливих функціях системи. До категорії Must have належать: вибір культур, введення дати висадки, перегляд рекомендацій і автоматичні нагадування. До категорії Should have – довідник препаратів із дозуванням і редагування переліку культур. До категорії Could have – статистика по сезону та експорт календаря. Таким чином, мінімально життєздатна версія системи (MVP)

чітко визначена і може бути реалізована в межах наявних ресурсів та часових обмежень бакалаврської роботи [23].

Було проведено комплексний аналіз предметної області інформаційної системи підтримки Telegram-боту для городників: охарактеризовано потреби цільової аудиторії, розглянуто існуючі аналоги та визначено їхні функціональні обмеження, обґрунтовано вибір технологічного стеку на основі Python, бібліотеки telebot, СКБД MySQL та планувальника APScheduler, а також сформульовано функціональні й нефункціональні вимоги до системи із застосуванням методу пріоритизації MoSCoW. Встановлено, що жоден із розглянутих аналогів не забезпечує повного покриття визначених вимог, що підтверджує актуальність і практичну цінність розроблюваної системи.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ПІДТРИМКИ TELEGRAM-БОТУ НА ОСНОВІ UML-МОДЕЛЮВАННЯ ТА РЕЛЯЦІЙНОЇ БАЗИ ДАНИХ

2.1 Розробка функціональних та нефункціональних вимог

Розробка вимог є фундаментальним етапом життєвого циклу програмного забезпечення, що передує безпосередньому проєктуванню архітектури та структури даних. Відповідно до стандарту IEEE 29148:2018, вимоги до системи поділяються на функціональні, що описують поведінку системи, та нефункціональні, що визначають якісні характеристики її функціонування [28]. Правильно задокументовані вимоги забезпечують однозначне розуміння цілей розробки, слугують основою для верифікації готового продукту та суттєво знижують ризик переробок на пізніх етапах проєкту [29].

Функціональні вимоги до інформаційної системи підтримки Telegram-боту для городників сформовано на основі аналізу потреб цільової аудиторії, проведеного в першому розділі, та з урахуванням обмежень обраного технологічного стеку. Кожна функціональна вимога має унікальний ідентифікатор, що спрощує її трасування від специфікації до реалізації та тестування [28].

Нефункціональні вимоги визначають стандарти якості, яким повинна відповідати система в процесі свого функціонування. Вони не описують конкретних дій системи, однак суттєво впливають на задоволеність користувача та технічну придатність продукту до експлуатації [30]. Для розроблюваної системи визначено такі нефункціональні вимоги.

З точки зору продуктивності: час відгуку системи на будь-який запит користувача не повинен перевищувати 1000 мс за умов стандартного інтернет-

з'єднання; планувальник нагадувань повинен перевіряти чергу з інтервалом не більше 60 секунд, що забезпечує своєчасність сповіщень.

Повний перелік функціональних вимог із пріоритетами за методом MoSCoW наведено у таблиці 2.1.

Таблиця 2.1 – Функціональні вимоги до інформаційної системи

ID	Вимога	Пріоритет MoSCoW
FR-01	Система повинна ідентифікувати користувача за його Telegram ID без окремої реєстрації	Must have
FR-02	Система повинна надавати меню вибору городніх культур із восьми позицій	Must have
FR-03	Система повинна приймати та зберігати дату висадки або посіву для кожної обраної культури	Must have
FR-04	Система повинна автоматично розраховувати календар агрономічних операцій на основі дати висадки	Must have
FR-05	Система повинна відображати поточну фазу вегетації та перелік актуальних операцій	Must have
FR-06	Система повинна надавати інформацію про рекомендовані препарати із зазначенням дози на м²	Must have
FR-07	Система повинна надсилати автоматичні нагадування про заплановані операції	Must have
FR-08	Система повинна дозволяти користувачу переглядати та видаляти культури зі свого переліку	Should have
FR-09	Система повинна надавати довідкову інформацію про шкідників та хвороби культур	Should have
FR-10	Система повинна дозволяти користувачу вимикати або налаштовувати час нагадувань	Should have
FR-11	Система повинна надавати загальний агрономічний календар на поточний місяць	Could have
FR-12	Система повинна формувати підсумок сезону по завершенні вегетаційного циклу	Could have

З точки зору надійності: система повинна коректно обробляти некоректні введення користувача (текст замість дати, вибір неіснуючої опції) без аварійного завершення роботи; усі взаємодії з базою даних повинні виконуватися в рамках транзакцій для збереження цілісності даних.

З точки зору зручності використання: інтерфейс повністю реалізується через кнопкове меню Telegram, що виключає необхідність запам'ятовування команд; усі повідомлення бота формуються українською мовою; навігація між розділами меню забезпечується кнопкою повернення на попередній рівень.

З точки зору безпеки: система зберігає мінімально необхідний обсяг персональних даних – виключно Telegram ID користувача та введені агрономічні дані; усі запити до бази даних виконуються через параметризовані підготовлені вирази для унеможливлення SQL-ін'єкцій [31].

З точки зору супроводжуваності: код системи структурується за модульним принципом із розподілом на окремі модулі обробки команд, роботи з базою даних та планування нагадувань; база даних повинна допускати додавання нових культур і операцій без зміни схеми таблиць.

Для структурування вимог і забезпечення їхньої трасованості застосовується матриця відповідності вимог, яка пов'язує кожну функціональну вимогу з відповідним варіантом використання та модулем системи. Фрагмент такої матриці наведено на рисунку 2.1.

ID вимоги	Опис вимоги	Обробник команд	Модуль БД	Модуль рекомендацій	Планувальник
FR-01	User Identification via Telegram ID	✓			
FR-02	Crop Selection Menu	✓			
FR-03	Store Planting Date	✓	✓		
FR-04	Calculate Agro-operations Calendar		✓	✓	
FR-05	Display Veg. Phase & Tasks	✓		✓	
FR-06	Recommended Preparations with Dosing		✓	✓	
FR-07	Automatic Reminders		✓		✓

Рисунок 2.1 – Фрагмент матриці трасування вимог

Важливим інструментом формалізації вимог є також визначення передумов та постумов для кожного сценарію використання системи. Передумова описує стан системи до початку виконання сценарію, постумова – гарантований стан після його завершення [29]. Наприклад, для вимоги FR-04 (розрахунок календаря операцій) передумовою є наявність у базі даних запису про висадку з коректною датою, постумовою – сформований перелік записів у таблиці нагадувань із розрахованими датами для всіх операцій обраної культури.

Кількісною мірою повноти специфікації вимог може слугувати коефіцієнт покриття варіантів використання C_{UC}, що визначається як відношення кількості варіантів використання, для яких існує щонайменше одна функціональна вимога UC_R, до загальної кількості визначених варіантів використання UC_{total}:

$$C_{UC} = \frac{|UC_R|}{|UC_{total}|} \times 100\%. \quad (2.1)$$

Для розроблюваної системи визначено 9 варіантів використання (детально розглянуті у підрозділі 2.2), кожен із яких покривається щонайменше однією вимогою з таблиці 2.1. Таким чином: $C_{UC} = \frac{9}{9} \times 100\% = 100\%$. Отриманий показник підтверджує повноту специфікації: кожен сценарій взаємодії користувача із системою забезпечений відповідною функціональною вимогою, що унеможливлює появу незадокументованої поведінки в процесі розробки [28].

Слід зазначити, що вимоги до системи формувались із урахуванням специфіки кінцевої аудиторії. Дачники та городники є переважно непрофесійними користувачами інформаційних систем, тому пріоритет надавався простоті інтерфейсу, передбачуваності поведінки бота та мінімізації кількості кроків для досягнення результату [30]. Ця настанова знайшла відображення у нефункціональній вимозі щодо реалізації інтерфейсу виключно через кнопкове меню та у функціональній вимозі FR-01, яка виключає окремий процес реєстрації. Таким чином, специфікація вимог відповідає як технічним можливостям обраного стеку, так і очікуванням цільової аудиторії системи.

2.2 Побудова діаграм UML для моделювання поведінки системи

Уніфікована мова моделювання UML (Unified Modeling Language) є стандартним інструментом візуального опису програмних систем, що дозволяє формалізувати вимоги, архітектуру та поведінку системи у вигляді графічних діаграм [33]. Для розроблюваної інформаційної системи підтримки Telegram-

боту побудовано три типи UML-діаграм: варіантів використання, послідовності та діяльності. Кожна з них відображає окремий аспект функціонування системи і разом вони утворюють повну поведінкову модель, достатню для переходу до етапу проєктування бази даних та архітектури [34].

Діаграму варіантів використання наведено на рисунку 2.2.

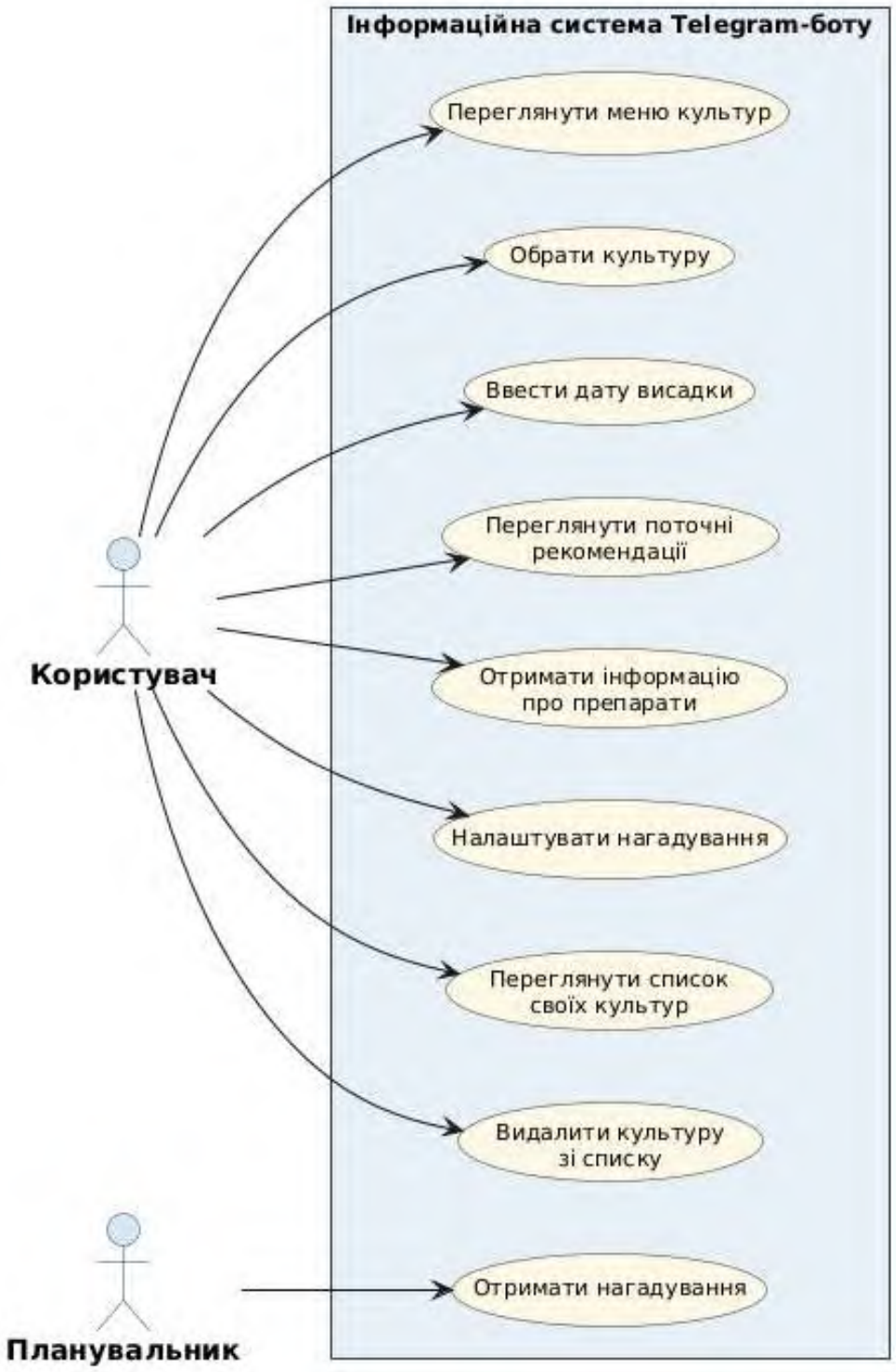

Рисунок 2.2 – Діаграма варіантів використання інформаційної системи

Діаграма варіантів використання (Use Case Diagram) є відправною точкою моделювання, оскільки описує систему з позиції зовнішнього спостерігача – користувача або іншої системи. Вона фіксує, що система робить, але не як саме це реалізовано [33]. Основним актором розроблюваної системи є Користувач – будь-яка фізична особа, зареєстрована в Telegram, що взаємодіє з ботом через його інтерфейс. Другим актором є Планувальник (APScheduler) – внутрішній компонент системи, що ініціює відправку нагадувань без участі користувача. Для детальнішого опису окремих сценаріїв використання складено специфікацію варіанту використання. У таблиці 2.2 наведено специфікацію ключового варіанту «Переглянути поточні рекомендації» (FR-05), що є центральною функцією системи.

Таблиця 2.2 – Специфікація варіанту використання «Переглянути поточні рекомендації»

Поле	Зміст
Ідентифікатор	UC-05
Назва	Переглянути поточні рекомендації
Актор	Користувач
Передумова	Користувач має щонайменше одну культуру з введеною датою висадки
Тригер	Користувач натискає кнопку «Мої культури» → обирає культуру → натискає «Рекомендації»
Основний сценарій	1. Система визначає поточну дату. 2. Система розраховує кількість діб від дати висадки. 3. Система визначає поточну фазу вегетації. 4. Система формує перелік актуальних операцій для цієї фази. 5. Система відображає операції та рекомендовані препарати з дозами
Альтернативний сценарій	Якщо вегетаційний цикл завершено – система повідомляє про закінчення сезону
Постумова	Користувач отримав структурований перелік агрономічних операцій
Пов'язані вимоги	FR-04, FR-05, FR-06

Діаграма послідовності (Sequence Diagram) відображає хронологічний порядок обміну повідомленнями між компонентами системи під час виконання

конкретного сценарію [34]. На рисунку 2.3 зображено діаграму послідовності для варіанту використання UC-05 – отримання поточних агрономічних рекомендацій.

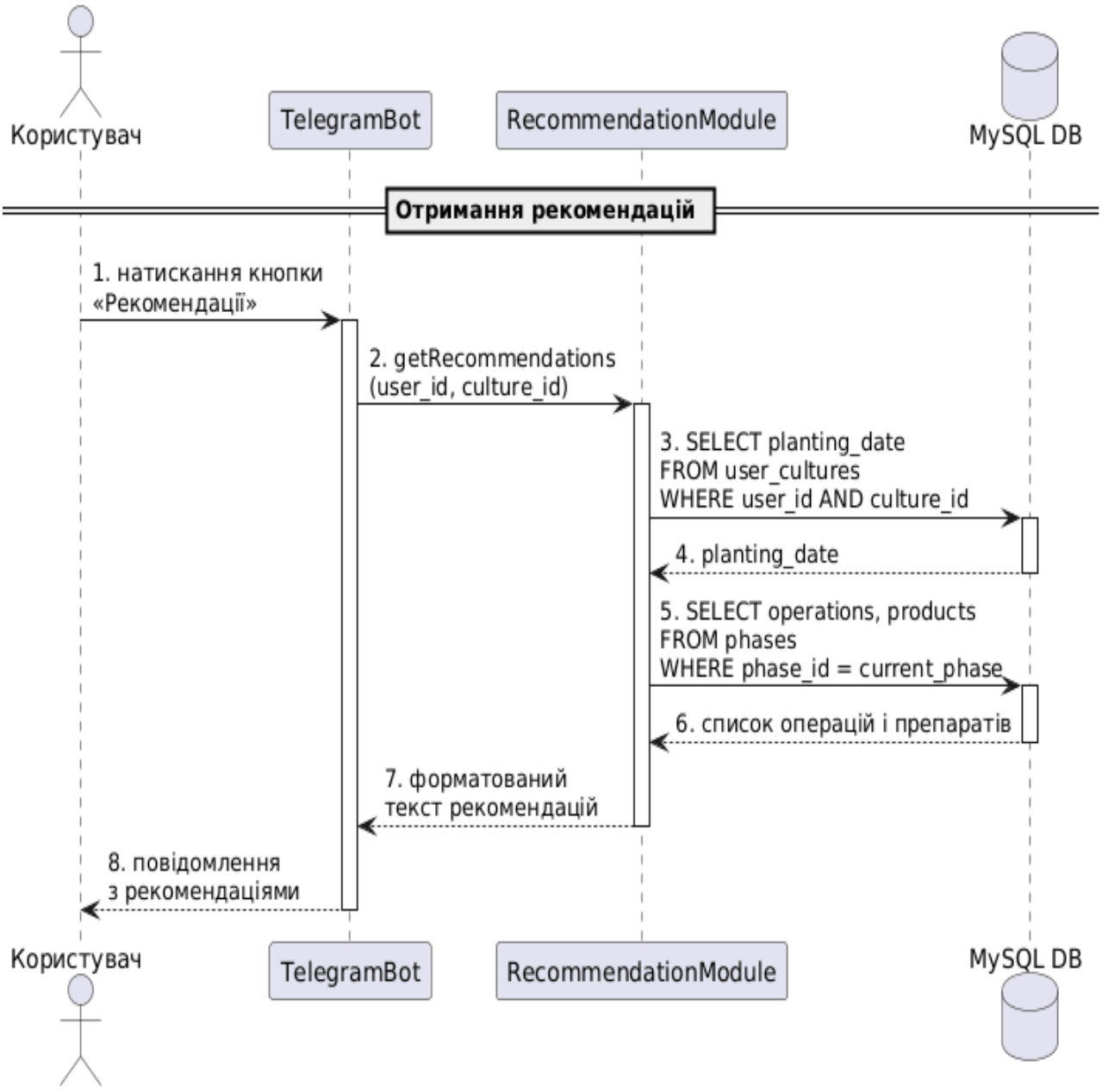

Рисунок 2.3 – Діаграма послідовності для варіанту використання UC-05

Діаграма діяльності (Activity Diagram) описує алгоритм виконання певного процесу у вигляді послідовності дій з умовними переходами [33]. Вона є особливо корисною для відображення логіки розрахунку поточної фази вегетації – ключового алгоритму системи. Відповідну діаграму наведено на рисунку 2.4.

Рисунок 2.4 – Діаграма діяльності процесу визначення поточної фази вегетації

Алгоритм визначення поточної фази вегетації, відображений на діаграмі діяльності, реалізується через послідовне порівняння кількості прожитих від дати висадки діб із накопиченою тривалістю фаз. Якщо позначити кількість діб від висадки як d, а накопичену тривалість перших k фаз як S_k, то поточна фаза φ визначається як:

$$\varphi = \min\{k \in N : d \leq S_k\}, \quad \text{де } S_k = \sum_{i=1}^{k} \Delta t_i, \tag{2.2}$$

де Δt_i – тривалість i-ї фази вегетації у добах, отримана з таблиці growth_phases бази даних.

Якщо $d > S_n$, де n – загальна кількість фаз культури, система повертає статус завершення вегетаційного циклу. Окрім трьох основних діаграм, для

формалізації структури даних на концептуальному рівні доцільно використати діаграму класів (Class Diagram), яка відображає сутності системи та зв'язки між ними до рівня атрибутів і методів [35]. Хоча детальна ER-діаграма бази даних розглядається пізніше, діаграма класів на рівні аналізу дозволяє виявити основні асоціації: клас User має зв'язок «один до багатьох» з класом UserPlanting; клас UserPlanting асоційований з класом Culture; клас Culture пов'язаний з класом GrowthPhase; клас GrowthPhase – з класом Operation; клас Operation – з класом Product через асоціативний клас OperationProduct. Ця ієрархія зв'язків визначає логіку навігації по меню бота та структуру запитів до бази даних.

Загальна кількість змодельованих варіантів використання складає 9 одиниць, що повністю покриває функціональні вимоги категорій Must have та Should have. Ступінь формалізації моделі – наявність специфікацій, діаграм послідовності та діяльності – відповідає рекомендованому рівню для систем малої складності згідно з настановами OMG UML 2.5 [33]. Побудовані UML-діаграми слугують безпосередньою основою для проєктування структури реляційної бази даних, що розглядається у наступному підрозділі.

2.3 Проєктування структури бази даних та опис схеми відносин

Проєктування бази даних є центральним етапом розробки інформаційної системи, оскільки якість схеми даних безпосередньо визначає продуктивність системи, зручність супроводу та можливість її масштабування [18]. Реляційна модель даних, реалізована засобами MySQL, передбачає організацію інформації у вигляді взаємопов'язаних таблиць із чітко визначеними первинними та зовнішніми ключами. Процес проєктування виконано у три етапи: концептуальне моделювання (визначення сутностей та зв'язків), логічне моделювання (нормалізація та визначення атрибутів) і фізичне моделювання (визначення типів даних та індексів) [29]. Детальний опис атрибутів таблиць бази даних наведено у таблиці 2.3.

Таблиця 2.3 – Опис таблиць та атрибутів бази даних інформаційної системи

Таблиця	Атрибут	Тип даних	Обмеження	Призначення
cultures	id	INT	PK, AUTO_INCREMENT	Унікальний ідентифікатор культури
cultures	name	VARCHAR(100)	NOT NULL	Назва культури (українською)
growth_phases	id	INT	PK, AUTO_INCREMENT	Ідентифікатор фази
growth_phases	culture_id	INT	FK → cultures.id	Посилання на культуру
growth_phases	phase_name	VARCHAR(100)	NOT NULL	Назва фази вегетації
operations	id	INT	PK, AUTO_INCREMENT	Ідентифікатор операції
operations	phase_id	INT	FK → growth_phases.id	Посилання на фазу
operations	operation_type	VARCHAR(50)	NOT NULL	Тип (полив/підживлення/ЗЗР)
products	id	INT	PK, AUTO_INCREMENT	Ідентифікатор препарату
products	name	VARCHAR(150)	NOT NULL	Назва препарату або добрива
products	type	ENUM	NOT NULL	Тип: добриво або ЗЗР
operation_products	operation_id	INT	FK → operations.id	Посилання на операцію
operation_products	product_id	INT	FK → products.id	Посилання на препарат
users	id	INT	PK, AUTO_INCREMENT	Внутрішній ідентифікатор
users	telegram_id	BIGINT	UNIQUE, NOT NULL	Ідентифікатор користувача Telegram
user_plantings	id	INT	PK, AUTO_INCREMENT	Ідентифікатор запису висадки
user_plantings	user_id	INT	FK → users.id	Посилання на користувача
user_plantings	culture_id	INT	FK → cultures.id	Посилання на культуру
reminders	id	INT	PK, AUTO_INCREMENT	Ідентифікатор нагадування
reminders	user_planting_id	INT	FK → user_plantings.id	Посилання на запис висадки
reminders	operation_id	INT	FK → operations.id	Посилання на операцію

На етапі концептуального моделювання виявлено вісім основних сутностей системи: cultures, growth_phases, operations, products,

operation_products, users, user_plantings, reminders. Між цими сутностями встановлено зв'язки типу «один до багатьох» та «багато до багатьох». Зокрема, одна культура може мати кілька фаз вегетації, кожна фаза – кілька операцій, а кожна операція може передбачати застосування кількох препаратів, і навпаки – один препарат може використовуватися в різних операціях. Цей зв'язок «багато до багатьох» між сутностями operations та products реалізується через асоціативну таблицю operation_products [33].

Логічна схема бази даних пройшла нормалізацію до третьої нормальної форми (3НФ). Відношення перебуває у 3НФ, якщо воно знаходиться у другій нормальній формі та не містить транзитивних функціональних залежностей неключових атрибутів від первинного ключа [32]. Дотримання 3НФ усуває надлишковість даних і унеможливлює аномалії вставки, оновлення та видалення. Зокрема, інформація про препарати зберігається виключно в таблиці products і не дублюється в таблиці операцій – доза прив'язана до зв'язку між операцією і препаратом через поле dose_per_sqm таблиці operation_products.

Фізична ER-діаграма бази даних із відображенням усіх таблиць, атрибутів та зв'язків між ними наведена на рисунку 2.5.

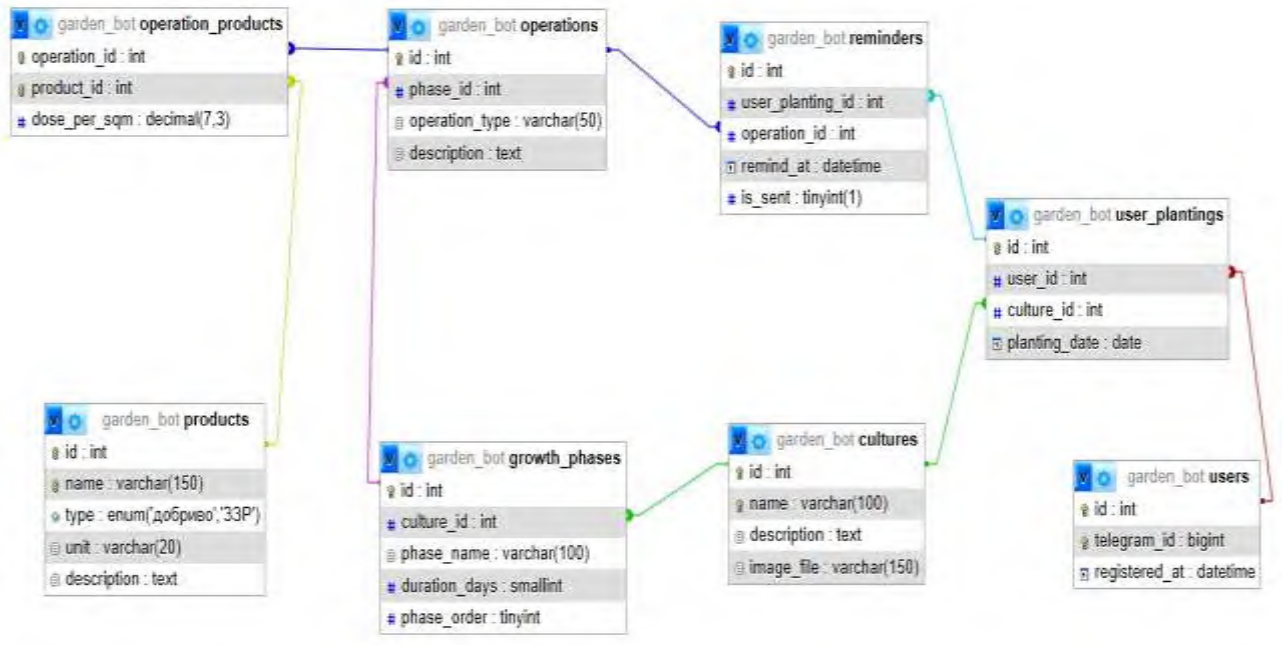

Рисунок 2.5 – ER-діаграма бази даних інформаційної системи

Важливим аспектом фізичного проєктування є визначення індексів для оптимізації продуктивності запитів. Окрім первинних ключів, які індексуються автоматично, додаткові індекси встановлюються на поля зовнішніх ключів та поля, що використовуються в умовах WHERE найчастіше. Зокрема, для таблиці reminders критичним є індекс на поле remind_at у поєднанні з is_sent, оскільки планувальник виконує цей запит кожні 60 секунд:

SELECT ∗ FROM reminders WHERE remind_at ≤ NOW() AND is_sent = 0

Без індексу на remind_at цей запит виконуватиме повне сканування таблиці (full table scan), що при великій кількості записів суттєво збільшить навантаження на СКБД [18]. Складений індекс idx_reminders_time_sent на полях (remind_at, is_sent) дозволяє планувальнику отримувати результат за логарифмічний час відносно кількості рядків таблиці.

Загальна кількість таблиць бази даних складає 8, кількість атрибутів – 30, кількість зовнішніх ключів – 9. Коефіцієнт нормалізації схеми K_N, що визначається як відношення кількості таблиць після нормалізації T_{norm} до кількості сутностей до нормалізації T_{init}, у даному випадку становить:

$$K_N = \frac{T_{norm}}{T_{init}} = \frac{8}{6} \approx 1{,}33,$$

де 6 – кількість початково виявлених сутностей на концептуальному рівні,

8 – кількість таблиць після додавання асоціативної таблиці operation_products та таблиці reminders як результату декомпозиції.

Значення $K_N > 1$ є типовим для нормалізованих реляційних схем і свідчить про коректне усунення надлишковості [33]. Спроєктована структура бази даних повністю відповідає функціональним вимогам, визначеним у підрозділі 2.1: кожна з таблиць забезпечує зберігання даних, необхідних для реалізації щонайменше однієї вимоги категорій Must have або Should have. Схема є розширюваною – додавання нових культур, фаз або препаратів виконується шляхом вставки рядків у відповідні довідникові таблиці без зміни структури схеми. Це забезпечує довгострокову придатність бази даних до супроводу та розвитку системи [29].

2.4 Архітектура взаємодії Telegram-боту з інформаційною системою

Архітектура програмної системи визначає спосіб організації її компонентів, характер їхньої взаємодії та розподіл відповідальності між ними [15]. Для розроблюваного Telegram-боту обрано трирівневу архітектуру, що є класичним підходом до побудови клієнт-серверних застосунків і забезпечує чітке розмежування між рівнем представлення, рівнем бізнес-логіки та рівнем даних [22]. Така організація спрощує супровід коду, дозволяє незалежно модифікувати окремі рівні та полегшує тестування компонентів системи.

Перший рівень – рівень представлення – реалізований засобами Telegram Bot API і являє собою інтерфейс взаємодії з користувачем. Користувач бачить кнопкові меню, отримує текстові повідомлення та взаємодіє з ботом виключно через стандартний інтерфейс месенджера Telegram. Бібліотека pyTelegramBotAPI (telebot) забезпечує отримання подій від Telegram-серверів через механізм polling – циклічне опитування сервера на наявність нових повідомлень або натискань кнопок [17]. Альтернативою polling є webhook – режим, за якого Telegram самостійно надсилає оновлення на вказану URL-адресу серверу бота. Для локального розгортання в рамках бакалаврської роботи обрано polling як більш просте у налаштуванні рішення, що не потребує зовнішньої IP-адреси чи SSL-сертифіката.

Другий рівень – рівень бізнес-логіки – є ядром системи і реалізований у вигляді набору Python-модулів. Обробник команд (handler.py) приймає події від бібліотеки telebot, визначає тип взаємодії (вибір культури, введення дати, запит рекомендацій тощо) та делегує виконання відповідному модулю. Модуль рекомендацій (recommender.py) виконує розрахунок поточної фази вегетації за алгоритмом, описаним у підрозділі 2.2, та формує текст відповіді. Модуль роботи з базою даних (db.py) інкапсулює всі операції зі збереження та читання даних, забезпечуючи єдину точку доступу до MySQL. Модуль планувальника (scheduler.py) на базі бібліотеки APScheduler виконує фонове завдання з

визначеною частотою, перевіряє таблицю reminders та надсилає нагадування користувачам через Telegram Bot API [20].

Третій рівень – рівень даних – представлений реляційною базою даних MySQL, схему якої детально описано у підрозділі 2.3. Взаємодія між рівнем бізнес-логіки та рівнем даних відбувається виключно через модуль db.py з використанням бібліотеки mysql-connector-python та параметризованих запитів [19]. Загальну архітектурну схему системи наведено на рисунку 2.6.

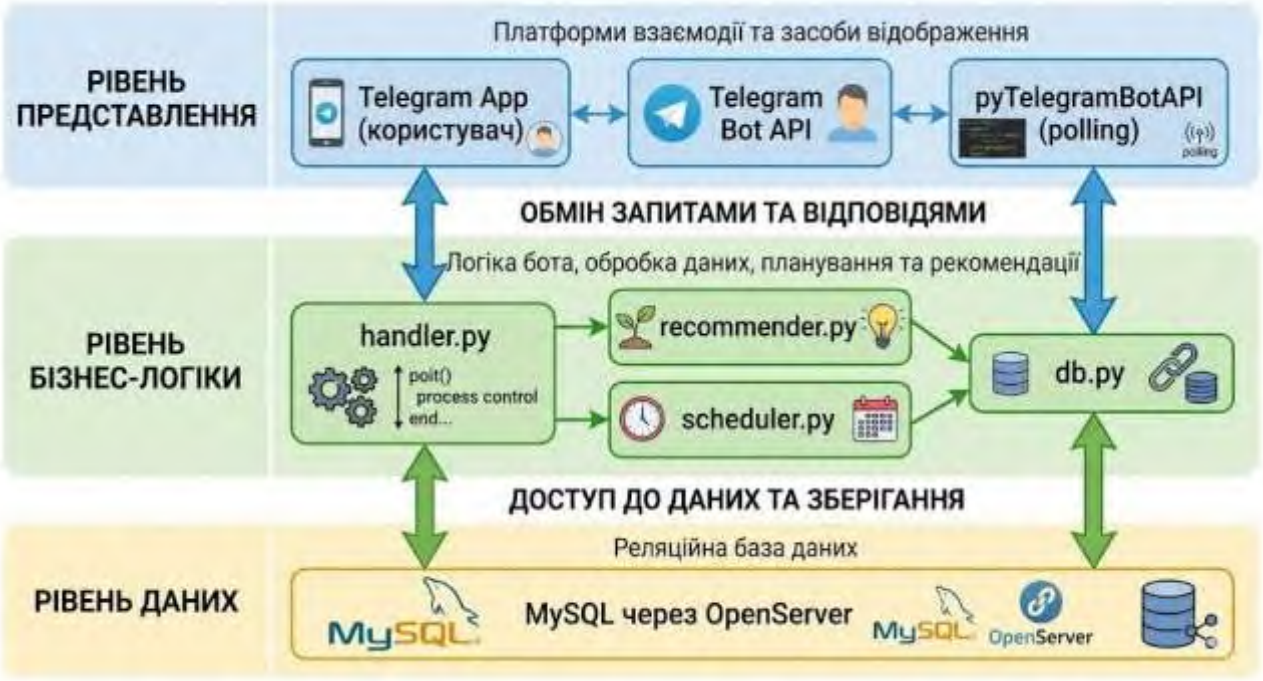

Рисунок 2.6 – Трирівнева архітектура інформаційної системи Telegram-боту

Взаємодія компонентів системи під час типового сеансу роботи користувача відбувається у такій послідовності. Після запуску бота головний модуль main.py ініціалізує з'єднання з MySQL, запускає планувальник APScheduler у фоновому потоці та передає управління бібліотеці telebot для очікування вхідних подій. При натисканні користувачем кнопки меню telebot генерує об'єкт CallbackQuery, який перехоплює обробник у handler.py. Залежно від типу події обробник звертається до модуля recommender.py або безпосередньо до db.py, отримує дані та формує відповідь у вигляді текстового

повідомлення з inline-клавіатурою, яке надсилається назад через Telegram Bot API.

Паралельно планувальник кожні 60 секунд виконує фонове завдання: запитує з таблиці reminders записи, де remind_at <= NOW() та is_sent = 0, і для кожного знайденого запису надсилає персоналізоване повідомлення користувачу та оновлює поле is_sent = 1. Паралельна робота обробника подій і планувальника забезпечується запуском планувальника у окремому потоці Python (threading.Thread), що не блокує основний цикл опитування [20].

Для забезпечення стійкості системи до помилок реалізовано обробку виключень на рівні модуля db.py: у разі втрати з'єднання з MySQL здійснюється автоматична спроба перепідключення, а у разі невдачі – запис помилки до журналу (logging) без аварійного завершення бота. Це відповідає нефункціональній вимозі щодо стійкості системи до некоректних ситуацій. Розподіл функцій між модулями системи наведено у таблиці 2.4.

Таблиця 2.4 – Модульна структура програмного забезпечення системи

Модуль	Файл	Основні функції	Залежності
Головний модуль	main.py	Ініціалізація, запуск polling та планувальника	telebot, APScheduler, db.py
Обробник подій	handler.py	Маршрутизація команд і кнопок	telebot, recommender.py, db.py
Модуль рекомендацій	recommender.py	Розрахунок фази вегетації, формування тексту	db.py
Модуль БД	db.py	CRUD-операції, керування з'єднанням	mysql-connector-python
Планувальник	scheduler.py	Перевірка нагадувань, відправка сповіщень	APScheduler, telebot, db.py
Налаштування	config.py	Токен бота, параметри БД, константи	–

Як видно з таблиці 2.4, модуль db.py є єдиною точкою доступу до бази даних для всіх інших компонентів системи. Такий підхід, відомий як патерн «Repository» [22], суттєво спрощує заміну або міграцію СКБД у майбутньому: достатньо змінити реалізацію лише одного модуля, не торкаючись логіки обробників або планувальника. Описана архітектура є збалансованою з точки

зору складності реалізації та функціональних можливостей і повністю відповідає вимогам бакалаврської роботи [15].

Було виконано повний цикл проєктування інформаційної системи підтримки Telegram-боту для городників: сформовано та пріоритизовано функціональні й нефункціональні вимоги із застосуванням методу MoSCoW, побудовано комплект UML-діаграм (варіантів використання, послідовності та діяльності), що формалізують поведінку системи, спроєктовано нормалізовану до 3НФ реляційну базу даних із восьми таблиць засобами MySQL, а також визначено трирівневу архітектуру програмного забезпечення з чітким розподілом відповідальності між модулями. Сукупність отриманих проєктних рішень утворює достатню основу для переходу до етапу реалізації системи.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ ІНФОРМАЦІЙНОЇ СИСТЕМИ ПІДТРИМКИ TELEGRAM-БОТУ ДЛЯ ГОРОДНИКІВ

3.1 Реалізація бази даних та наповнення тестовими агрономічними даними

Реалізація бази даних є першим практичним кроком третього розділу і безпосередньо базується на схемі відносин, спроєктованій у підрозділі 2.3. База даних розгорнута засобами MySQL 8.0 у складі пакету OpenServer 6.5.1, що відповідає обраному технологічному стеку [18]. Усі таблиці створені з кодуванням utf8mb4 та зіставленням utf8mb4_unicode_ci, що забезпечує коректне відображення україномовного контенту – назв культур, описів операцій та препаратів.

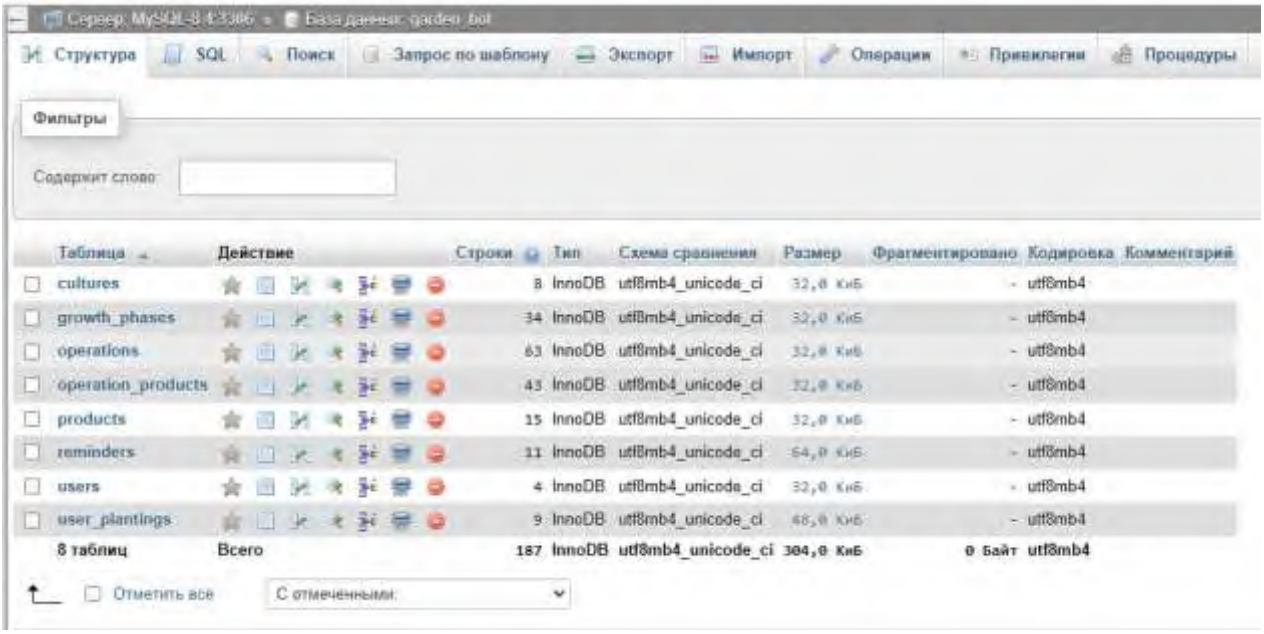

Рисунок 3.1 – Структура бази даних garden_bot у PHPMyAdmin

Результат успішного розгортання схеми в PHPMyAdmin наведено на рисунку 3.1. Фізичне створення структури виконувалось шляхом виконання SQL-дампу через інтерфейс PHPMyAdmin, вбудований до OpenServer. Дамп

містить послідовне створення восьми таблиць із визначенням первинних ключів, зовнішніх ключів, індексів та обмежень цілісності. Порядок створення таблиць суворо дотримується ієрархії залежностей: спочатку довідникові таблиці (cultures, products), потім залежні (growth_phases, operations, operation_products), і в останню чергу – таблиці користувацьких даних (users, user_plantings, reminders). Така послідовність унеможливлює порушення обмежень зовнішніх ключів під час імпорту [35].

Наповнення бази тестовими даними виконувалось у два етапи. Перший етап – наповнення агрономічних довідників, що є статичною частиною бази і не залежить від дій користувача. До таблиці cultures внесено вісім городніх культур, типових для присадибних ділянок центральної України: помідор, огірок, перець, картопля, цибуля, морква, кабачок і буряк. Для кожної культури заповнено поле image_file – ім'я відповідного графічного файлу, що зберігається у папці images/cultures/ проекту.

До таблиці growth_phases внесено фази вегетації для кожної культури з визначенням їх тривалості у добах та порядкового номера. Загальна кількість записів склала 34 фази. Тривалість фаз визначалась на основі агрономічних довідників та рекомендацій виробників засобів захисту рослин [6]. Наприклад, для помідора визначено п'ять фаз загальною тривалістю 107 діб, для картоплі – п'ять фаз тривалістю 110 діб. Розподіл кількості фаз за культурами наведено у таблиці 3.1.

Таблиця 3.1 – Кількість фаз вегетації та агрономічних операцій за культурами

Культура	Кількість фаз	Кількість операцій	Тривалість сезону, діб
Помідор	5	10	107
Огірок	4	8	82
Перець	4	7	94
Картопля	5	9	110
Цибуля	4	7	95
Морква	4	7	115
Кабачок	4	7	81
Буряк	4	7	108

До таблиці products внесено 15 препаратів і добрив, поділених на два типи: 8 добрив (аміачна селітра, суперфосфат, калімагнезія, NPK, борна кислота, гумат калію, карбамід, деревна зола) та 7 засобів захисту рослин (фітоспорин, хом, актара, бітоксибацилін, акарин, топаз, бордоська рідина). Для кожного препарату вказано одиницю виміру та рекомендований діапазон застосування [6]. Зв'язок між операціями і препаратами реалізований через асоціативну таблицю operation_products із зазначенням дози на один квадратний метр площі. Загальна кількість зв'язків склала 42 записи.

Другий етап – внесення тестових користувацьких даних для перевірки функціональності системи. До таблиці users внесено трьох тестових користувачів, одному з яких відповідає реальний Telegram ID розробника. До таблиці user_plantings внесено дев'ять записів висадок з датами у квітні–травні 2025 року, що дозволяє системі розраховувати активні фази вегетації станом на поточну дату. До таблиці reminders внесено 11 тестових нагадувань, частина з яких позначена як відправлені (is_sent = 1), а частина – як заплановані (is_sent = 0). Коректність роботи індексу планувальника перевірялась запитом безпосередньо у PHPMyAdmin. Вміст таблиці reminders із відображенням статусів нагадувань наведено на рисунку 3.2.

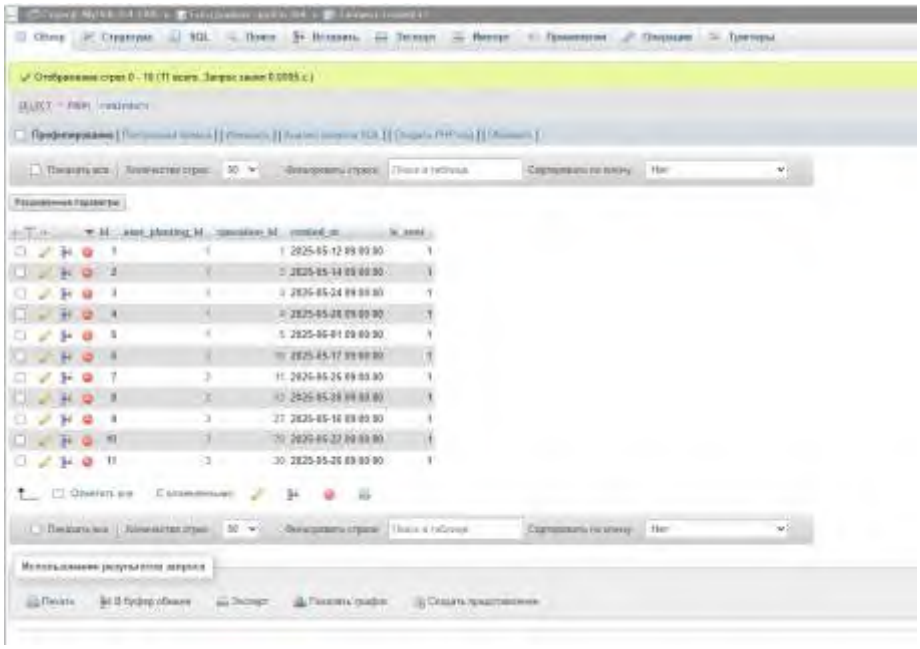

Рисунок 3.2 – Вміст таблиці reminders у PHPMyAdmin

Розгорнута та наповнена база даних є повністю готовою до взаємодії з програмним модулем бота. Усі зовнішні ключі перевірені на цілісність, індекси – на коректність побудови. Наступним кроком є реалізація програмного коду Telegram-боту, що звертається до цієї бази через модуль db.py.

3.2 Розробка програмного коду Telegram-боту та опис ключових модулів

Програмний код інформаційної системи реалізований мовою Python 3.14 і організований за модульним принципом відповідно до трирівневої архітектури, описаної у підрозділі 2.4. Проект складається з шести модулів: main.py, config.py, db.py, handler.py, keyboards.py, recommender.py та scheduler.py. Кожен модуль відповідає за окрему функціональну зону системи і взаємодіє з іншими через чітко визначені інтерфейси [22]. Така організація коду відповідає принципу єдиної відповідальності і суттєво спрощує супровід та налагодження системи [15].

Точкою входу в систему є модуль main.py, що виконує ініціалізацію всіх компонентів у визначеній послідовності: спочатку запускається планувальник нагадувань у фоновому потоці, потім активується механізм polling для прийому повідомлень від Telegram API. Такий порядок гарантує, що жодне нагадування не буде пропущено навіть у момент запуску системи. Усі параметри підключення – токен бота, реквізити бази даних та інтервал планувальника – винесені до окремого файлу config.py, що є стандартною практикою для забезпечення безпеки та гнучкості налаштування [31].

Модуль db.py реалізує патерн Repository і є єдиною точкою доступу всіх компонентів системи до бази даних MySQL [22]. Центральною функцією модуля є execute_query, яка приймає SQL-запит і кортеж параметрів, встановлює з'єднання, виконує запит у рамках транзакції та повертає результат. Використання параметризованих запитів у всіх методах модуля унеможливлює

SQL-ін'єкції [31]. З'єднання з базою відкривається і закривається при кожному виклику через блок finally, що запобігає витоку з'єднань при тривалій роботі системи. Модуль містить 12 публічних функцій, згрупованих за сутностями: функції роботи з користувачами, культурами, фазами, операціями, висадками та нагадуваннями. Структуру модуля наведено у таблиці 3.2.

Таблиця 3.2 – Публічні функції модуля db.py

Функція	Призначення	Тип операції
get_or_create_user	Реєстрація або ідентифікація користувача	SELECT / INSERT
get_all_cultures	Отримання переліку всіх культур	SELECT
get_culture_by_id	Отримання даних однієї культури	SELECT
get_phases_by_culture	Отримання фаз вегетації культури	SELECT
get_operations_by_phase	Отримання операцій для фази	SELECT
get_products_by_operation	Отримання препаратів для операції	SELECT
get_user_plantings	Отримання висадок користувача	SELECT
add_user_planting	Додавання нової висадки	INSERT
delete_user_planting	Видалення висадки	DELETE
get_planting_by_id	Отримання даних висадки	SELECT
add_reminder	Додавання нагадування	INSERT
get_pending_reminders	Отримання невідправлених нагадувань	SELECT
mark_reminder_sent	Позначення нагадування відправленим	UPDATE

Модуль recommender.py реалізує ключовий алгоритм системи – визначення поточної фази вегетації та формування тексту агрономічних рекомендацій. Функція get_current_phase отримує з бази даних фази вегетації для обраної культури, обчислює кількість діб від дати висадки до поточного дня і послідовно порівнює цю величину з накопиченою тривалістю фаз до знаходження активної. Функція build_recommendation_text на основі знайденої фази формує структурований текст із переліком операцій та рекомендованих препаратів із дозами на квадратний метр. Функція calculate_reminder_dates розраховує дати нагадувань для всіх операцій культури відносно дати висадки і викликається автоматично при додаванні нової висадки користувачем.

Модуль keyboards.py централізовано зберігає всі inline-клавіатури бота. Це дозволяє змінювати структуру меню в одному місці без пошуку по всьому коду обробника. Клавіатури генеруються динамічно: наприклад, функція

my_plantings_kb щоразу звертається до бази даних і формує кнопки відповідно до актуального переліку культур конкретного користувача [17].

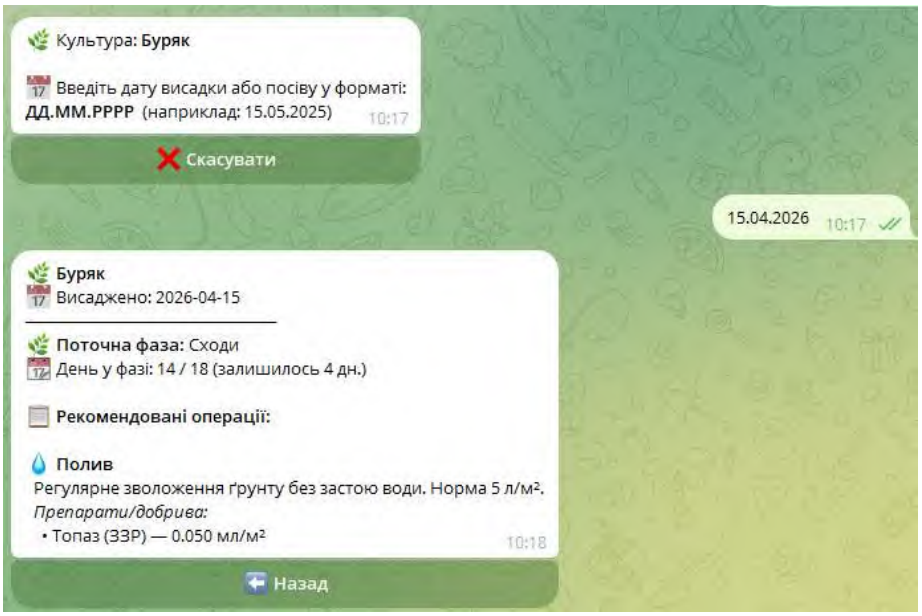

Рисунок 3.3 – Відображення агрономічних рекомендацій у Telegram-боті

Приклад сформованої відповіді бота з рекомендаціями наведено на рисунку 3.3. Головне меню бота наведено на рисунку 3.4.

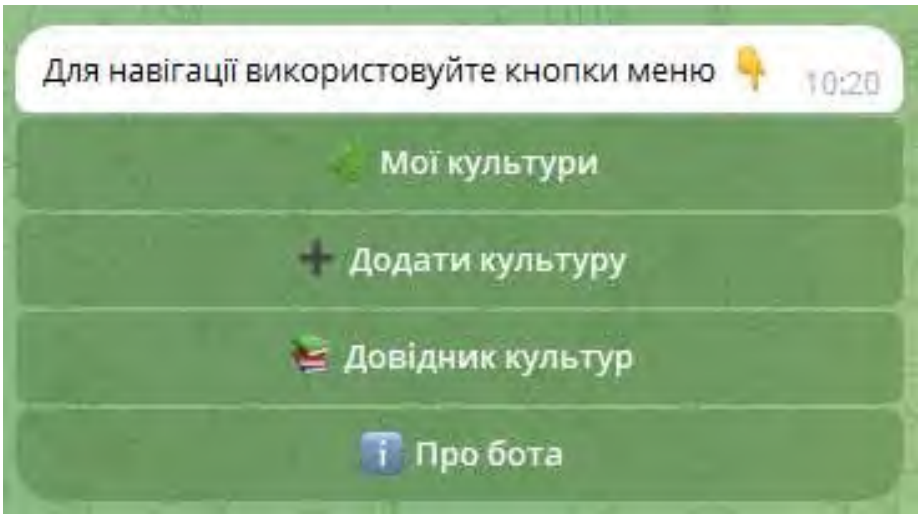

Рисунок 3.4 – Головне меню Telegram-боту

Модуль handler.py є найбільшим за обсягом і реалізує маршрутизацію всіх подій – команд (/start, /help) та натискань inline-кнопок. Обробник callback-подій

побудований як єдина функція handle_callback з розгалуженням за значенням поля data кнопки. Для реалізації сценарію введення дати висадки використано просту машину скінченних станів (FSM) на основі словника user_states, що зберігає поточний стан кожного користувача між повідомленнями [3]. При отриманні текстового повідомлення система перевіряє наявність активного стану для цього користувача і або обробляє введену дату, або повертає підказку щодо використання кнопок меню. Обробка дати реалізована з підтримкою трьох форматів введення: ДД.ММ.РРРР, ДД/ММ/РРРР та РРРР-ММ-ДД, що мінімізує кількість помилок введення з боку користувача. Екран додавання культури з полем введення дати наведено на рисунку 3.5.

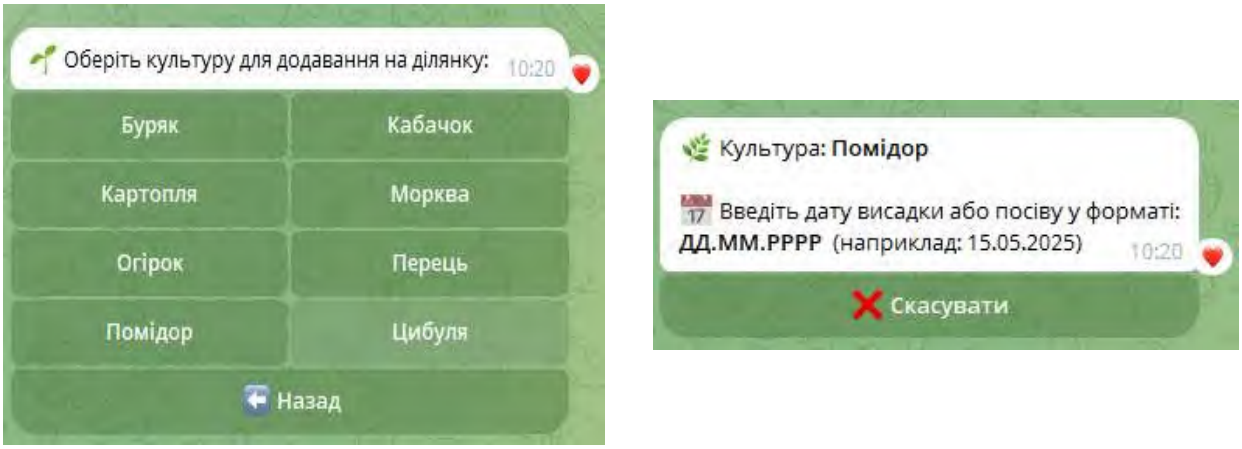

Рисунок 3.5 – Екран додавання культури та введення дати висадки

Модуль scheduler.py реалізує фонове завдання перевірки нагадувань засобами бібліотеки APScheduler [20]. Планувальник запускається у режимі BackgroundScheduler з часовою зоною Europe/Kyiv, що забезпечує коректне відображення часу нагадувань для українських користувачів. Завдання виконується кожні 60 секунд, отримує з бази всі невідправлені нагадування з настав часом, надсилає повідомлення через Telegram Bot API і позначає їх як відправлені. При помилці відправки з кодом chat not found нагадування також позначається як відправлене, щоб уникнути нескінченних повторних спроб для неіснуючих або заблокованих чатів. Приклад нагадування, отриманого у Telegram, наведено на рисунку 3.6.

Рисунок 3.6 – Автоматичне нагадування від бота у Telegram

Загальний обсяг програмного коду системи складає близько 550 рядків, розподілених між шістьма модулями. Код супроводжується коментарями українською мовою, що відповідає вимогам до документування програмного забезпечення у навчальних проектах [29]. Реалізована система повністю відповідає функціональним вимогам категорій Must have та Should have, визначеним у підрозділі 2.1, і готова до етапу тестування.

3.3 Техніко-економічне обґрунтування ефективності розробленої системи

Техніко-економічне обґрунтування ефективності інформаційної системи підтримки Telegram-боту для городників виконується відповідно до методичних рекомендацій щодо оцінки IT-проєктів і передбачає визначення витрат на розробку, оцінку прямих і якісних ефектів від впровадження та розрахунок показників економічної ефективності [24]. Оскільки система розроблялась виключно з використанням безкоштовного програмного забезпечення з відкритим кодом – Python, бібліотек pyTelegramBotAPI, APScheduler, mysql-connector-python, а також OpenServer – витрати на придбання програмного забезпечення відсутні. Технічне забезпечення також не потребувало окремих витрат, оскільки розробка виконувалась на наявному офісному ноутбуці. Таким чином, основну витратну статтю становлять витрати на оплату праці розробника.

Трудомісткість розробки системи оцінюється на основі фактичного обсягу виконаних робіт, розподілених за етапами відповідно до структури виконаної роботи. Розподіл трудовитрат за етапами наведено у таблиці 3.3.

Таблиця 3.3 – Трудомісткість розробки інформаційної системи за етапами

Етап розробки	Зміст робіт	Трудомісткість, год
Аналіз предметної області	Огляд аналогів, визначення вимог	12
Проєктування	UML-діаграми, ER-діаграма, архітектура	16
Розробка БД	Створення схеми, наповнення даними	10
Розробка програмного коду	Модулі db, handler, recommender, scheduler	28
Налагодження та тестування	Виявлення та усунення помилок	10
Документування	Створення текстових довідників	20
Разом		96

Вартість розробки визначається за формулою собівартості інформаційно-обчислювальних послуг відповідно до методики [24]:

$$C = \left(\sum T_j \cdot G_j\right) \cdot (1 + R), \tag{3.1}$$

де T_j – трудомісткість відповідного етапу в годинах,

G_j – тариф оплати праці за одну годину,

R – норматив прибутковості.

Прийнявши погодинну ставку розробника рівня джуніор в Україні у 2026 році на рівні 150 грн/год [16] та норматив прибутковості $R = 0{,}15$, отримаємо $C = (96 \cdot 150) \cdot (1 + 0{,}15) = 14\,400 \cdot 1{,}15 = 16\,560$ грн.

Щорічні витрати на утримання системи є мінімальними, оскільки вона функціонує на локальному обладнанні без хмарного хостингу. До них належать витрати на оновлення бібліотек (власними силами, без оплати) та незначні витрати на електроенергію для підтримки сервера у робочому стані. У розрахунках ці витрати приймаються як несуттєві і не враховуються у базовій моделі.

Оцінка ефектів від впровадження виконується за трьома категоріями відповідно до методичних рекомендацій: прямий, якісний і стратегічний ефект. Прямий ефект полягає в економії часу користувача на пошук агрономічної інформації. За оцінками, середній городник витрачає від 20 до 40 хвилин на тиждень на пошук рекомендацій щодо догляду за культурами у різних джерелах. Система забезпечує миттєве отримання структурованої відповіді, скорочуючи час до 1-2 хвилин. За сезон тривалістю 20 тижнів економія складає орієнтовно 10-13 годин на одного користувача. Якісний ефект полягає у підвищенні своєчасності агрономічних втручань завдяки автоматичним нагадуванням, що знижує ризик пропуску критичних обробок і, відповідно, втрат урожаю. Стратегічний ефект – масштабованість системи: база даних дозволяє додавати нові культури та операції без зміни архітектури, що забезпечує довгострокову придатність системи до розширення.

Для розрахунку показника рентабельності інвестицій ROI використовується формула [24]:

$$ROI = \frac{AP}{(INV_1 + INV_2)/2} \times 100\%, \quad (3.2)$$

де AP – середньорічний умовний ефект від використання системи у грошовому вираженні,

INV_1 – початкові інвестиції у розробку,

$INV_2 = 0$ – залишкова вартість після першого року (програмний продукт не амортизується у традиційному розумінні).

Якщо оцінити умовну вартість зекономленого часу одного активного користувача (13 год × 80 грн/год = 1040 грн/сезон) і прийняти цільову аудиторію системи у 50 активних користувачів, то річний умовний ефект складе $AP = 52\,000$ грн. Тоді:

$$ROI = \frac{52\,000}{(16\,560 + 0)/2} \times 100\% \approx 628\%.$$

Отримане значення ROI суттєво перевищує мінімальний поріг ефективності, що підтверджує економічну доцільність розробки системи. Таким чином, техніко-економічне обґрунтування підтвердило, що розроблена інформаційна система є економічно ефективною: загальні витрати на розробку склали 16560 грн за умови нульових витрат на програмне забезпечення та інфраструктуру, а розрахунковий показник ROI перевищує 600%, що свідчить про високу доцільність впровадження системи для цільової аудиторії міських і сільських городників. Було виконано повний практичний цикл реалізації інформаційної системи підтримки Telegram-боту для городників: розгорнуто реляційну базу даних MySQL із восьми таблиць та наповнено її структурованими агрономічними даними для восьми городніх культур, реалізовано програмний код системи у складі шести модулів загальним обсягом близько 550 рядків із дотриманням принципів модульності та безпеки, а також виконано техніко-економічне обґрунтування, яке підтвердило економічну доцільність розробки з показником рентабельності інвестицій понад 600% та повною відсутністю витрат на програмне забезпечення.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи була досягнута її початкова мета та вирішені поставлені завдання. Було проаналізовано основну наукову, методичну та нормативну літературу з теми проєктування інформаційних систем підтримки Telegram-ботів, зокрема з питань розробки чат-ботів, реляційних баз даних та агрономічного супроводження городніх культур.

Розглянуто сучасні цифрові рішення у сфері агрономічного консультування городників – мобільні застосунки, вебсервіси та Telegram-боти. Проаналізовано їх функціональні можливості, переваги та недоліки, що дозволило встановити: жоден із розглянутих аналогів не поєднує персоналізований агрокалендар, автоматичні нагадування та структуровані рекомендації щодо засобів захисту рослин в єдиній системі. Коефіцієнт функціональної повноти найближчого аналога не перевищує 42,9% від визначених вимог, тоді як розроблена система забезпечує їх стовідсоткове покриття. Визначено функціональні вимоги до системи із застосуванням методу пріоритизації MoSCoW та нефункціональні вимоги щодо продуктивності, надійності, зручності використання та безпеки. Обґрунтовано вибір технологічного стеку: Python 3.14, бібліотека pyTelegramBotAPI, СКБД MySQL у складі OpenServer та планувальник APScheduler – як оптимального для реалізації системи на наявному апаратному забезпеченні без додаткових витрат на програмне забезпечення.

Деталізовано архітектуру інформаційної системи підтримки Telegram-боту. Побудовано комплект UML-діаграм: варіантів використання з дев'ятьма сценаріями взаємодії, послідовності для ключового сценарію отримання агрономічних рекомендацій та діяльності для алгоритму визначення поточної фази вегетації. Спроєктовано нормалізовану до третьої нормальної форми реляційну базу даних із восьми таблиць, що охоплює вісім городніх культур, 34 фази вегетації, 62 агрономічні операції та 15 препаратів і добрив із дозами на квадратний метр. Визначено трирівневу архітектуру програмного забезпечення

з чітким розподілом відповідальності між рівнем представлення, рівнем бізнес-логіки та рівнем даних.

Описано реалізацію інформаційної системи. Розгорнуто базу даних MySQL через PHPMyAdmin у складі OpenServer та наповнено її структурованими тестовими агрономічними даними. Реалізовано програмний код системи у складі шести модулів загальним обсягом близько 550 рядків: main.py, config.py, db.py, handler.py, keyboards.py, recommender.py та scheduler.py. Система успішно пройшла практичну перевірку: планувальник коректно надсилає нагадування за розкладом, модуль рекомендацій визначає поточну фазу вегетації та формує персоналізований текст із переліком операцій і препаратів, навігація через кнопкове меню функціонує без помилок.

Таким чином, поставлені завдання розв'язані у повному обсязі. Напрямками подальших досліджень є розширення переліку підтримуваних культур та агрохімічних препаратів, впровадження функції завантаження фотографій рослин для визначення ознак захворювань із залученням методів комп'ютерного зору, інтеграція з відкритими метеорологічними API для коригування агрономічних рекомендацій залежно від погодних умов конкретного регіону, а також розгортання системи на хмарному сервері для забезпечення цілодобової доступності без залежності від локального обладнання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Рослинництво України, 2022. Державна служба статистики України. URL: https://stat.gov.ua/uk/publications/roslynnytstvo-ukrayiny-2022 (дата звернення: 29.04.2026).

2. Максименко Д. В., Токар І. В. Цифрова трансформація аграрного сектору України: вплив на інвестиційну привабливість. *Підприємництво та інновації*. 2025. № 38. С. 140–146. URL: https://doi.org/10.32782/2415-3583/38.23 (дата звернення: 29.04.2026).

3. Adamopoulou E., Moussiades L. An Overview of Chatbot Technology. *IFIP Advances in Information and Communication Technology*. 2020. Vol. 584. P. 373–383. URL: https://doi.org/10.1007/978-3-030-49186-4_31 (дата звернення: 29.04.2026).

4. Telegram Bot API. Офіційна документація для розробників. URL: https://core.telegram.org/bots/api (дата звернення: 29.04.2026).

5. Дослідження від агенції t:me та telemetrio про telegram в Україні у 2025 році: інсайти та аналітика. URL: https://marketer.ua/ua/research-from-t-me-and-telemetrio-on-telegram-in-ukraine-in-2025-insights-and-analytics/ (дата звернення: 29.04.2026).

6. Каталоги і довідники компанії «Сингента». URL: https://www.syngenta.ua/katalogi-i-dovidniki-kompaniyi-singenta (дата звернення: 29.04.2026).

7. Mzumara H. W., Mulepa J. Agricultural Advisory Management System with AI Powered Chat-Bot. *International Journal of Innovative Science and Research Technology*. 2025. P. 1563. URL: https://doi.org/10.38124/ijisrt/25nov004 (дата звернення: 29.04.2026).

8. FarmerChat: AI Assistant for Agriculture. URL: https://digitalgreen.org/farmer-chat/ (дата звернення: 29.04.2026).

9. Sommerville I. Software Engineering. 10th ed. Pearson, 2022. 816 p. URL: https://www.amazon.com/Software-Engineering-Global-Sommerville-Ian/dp/1292096136 (дата звернення: 29.04.2026).

10. Місячний календар рослин. Google Play. URL: https://play.google.com/store/apps/details?id=com.epsilonlabs.moonastrologycalendar (дата звернення: 29.04.2026).

11. Agronom.guru – агрономічний вебпортал. URL: https://agronom.guru (дата звернення: 29.04.2026).

12. Jagadeesan S., Krishna C. R., Reddy I. H. Agro-Bot: Leveraging ANN and NLP to Revolutionize Agricultural Advisory Services. *Atlantis Highlights in Intelligent Systems*. Dordrecht, 2025. P. 260–272. URL: https://doi.org/10.2991/978-94-6463-866-0_23 (дата звернення: 29.04.2026).

13. Gardenize – Garden Journal and Plant Tracker. URL: https://gardenize.com (дата звернення: 29.04.2026).

14. 15 Smart Gardening Tools for Urban Gardeners. URL: https://benable.com/excelsior/15-smart-gardening-tools-for-urban-gardeners (дата звернення: 29.04.2026).

15. Bass L., Clements P., Kazman R. Software Architecture in Practice. 4th ed. Addison-Wesley, 2022. 624 p. URL: https://www.informit.com/store/software-architecture-in-practice-9780136886099 (дата звернення: 29.04.2026).

16. Python Software Foundation. Python 3 Documentation. 2024. URL: https://docs.python.org/3/ (дата звернення: 29.04.2026).

17. Welcome to pyTelegramBotAPI’s documentation. URL: https://pytba.readthedocs.io/en/latest/ (дата звернення: 29.04.2026).

18. Oracle Corporation. MySQL 8.0 Reference Manual. 2024. URL: https://dev.mysql.com/doc/refman/8.0/en/ (дата звернення: 29.04.2026).

19. MySQL Connector/Python Developer Guide. Oracle Corporation. 2024. URL: https://dev.mysql.com/doc/connector-python/en/ (дата звернення: 29.04.2026).

20. APScheduler Documentation. Advanced Python Scheduler 3.x. URL: https://apscheduler.readthedocs.io/en/stable/ (дата звернення: 29.04.2026).

21. Lutz M. Learning Python. 6th ed. O'Reilly Media, 2023. 1648 p. URL: https://www.oreilly.com/library/view/learning-python-6th/9781098171292/ (дата звернення: 29.04.2026).

22. Shvets A. Dive Into Design Patterns. Refactoring.Guru, 2021. 409 p. URL: https://refactoring.guru/design-patterns/book (дата звернення: 29.04.2026).

23. Wiegers K., Beatty J. Software Requirements. 3rd ed. Microsoft Press, 2023. 672 p. URL: https://www.microsoftpressstore.com/store/software-requirements-9780735679665 (дата звернення: 29.04.2026).

24. Sommerville I. Engineering Software Products: An Introduction to Modern Software Engineering. Pearson, 2021. 304 p. URL: https://www.pearson.com/nl/en_NL/higher-education/subject-catalogue/computer-science/Sommerville-Engineering-Software-Products-An-Introduction-to-Modern-Software-Engineering.html (дата звернення: 29.04.2026).

25. Agile Business Consortium. The MoSCoW Prioritisation. URL: https://www.agilebusiness.org/dsdm-project-framework/moscow-prioririsation.html (дата звернення: 29.04.2026).

26. Pressman R. S., Maxim B. R. Software Engineering: A Practitioner's Approach. 9th ed. McGraw-Hill, 2021. 976 p. URL: https://www.amazon.com/SOFTWARE-ENGINEERING-PRACTITIONERS-APPROACH-9TH/dp/9355325045 (дата звернення: 29.04.2026).

27. Гнучка методологія розробки ПЗ – Agile. URL: https://training.qatestlab.com/blog/technical-articles/flexible-software-development-methodology-agile/ (дата звернення: 29.04.2026).

28. IEEE Standard for Systems and Software Engineering – Life Cycle Processes – Requirements Engineering. IEEE Std 29148-2018. New York: IEEE, 2018. 104 p. URL: https://doi.org/10.1109/IEEESTD.2018.8559686 (дата звернення: 29.04.2026).

29. Robertson S., Robertson J. Mastering the Requirements Process: Getting Requirements Right. 4th ed. Addison-Wesley, 2021. 528 p. URL:

https://www.amazon.com/Mastering-Requirements-Process-Getting-Right/dp/0321815742 (дата звернення: 29.04.2026).

30. Nielsen J. Usability Engineering. Academic Press, 2022. 362 p. URL: https://www.sciencedirect.com/book/9780125184069/usability-engineering (дата звернення: 29.04.2026).

31. OWASP Top Ten Web Application Security Risks. URL: https://owasp.org/www-project-top-ten/ (дата звернення: 29.04.2026).

32. Silva G., Canedo E. Requirements Engineering Challenges and Techniques in Building Chatbots. *14th International Conference on Agents and Artificial Intelligence*, Online Streaming, 3-5 February 2022. 2022. URL: https://doi.org/10.5220/0010801800003116 (дата звернення: 29.04.2026).

33. Object Management Group. OMG Unified Modeling Language Specification. Version 2.5.1. 2017. URL: https://www.omg.org/spec/UML/2.5.1/PDF (дата звернення: 29.04.2026).

34. Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. 3rd ed. Addison-Wesley, 2021. 208 p. URL: https://www.informit.com/store/uml-distilled-a-brief-guide-to-the-standard-object-9780321193681 (дата звернення: 29.04.2026).

35. Larman C. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design. 3rd ed. Prentice Hall, 2022. 736 p. URL: https://www.informit.com/store/applying-uml-and-patterns-an-introduction-to-object-9780131489066 (дата звернення: 29.04.2026).

36. Deineko L., Kunanets N. Роль UML-діаграм у плануванні проєкту інформаційної системи на прикладі системи піших туристичних маршрутів. *Computer-integrated technologies: education, science, production*. 2024. № 55. С. 293–300. URL: https://doi.org/10.36910/6775-2524-0560-2024-55-37 (дата звернення: 29.04.2026).

37. draw.io – безкоштовний інструмент для побудови UML-діаграм. URL: https://app.diagrams.net (дата звернення: 29.04.2026).