Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Інформаційна технологія надання рекомендацій для догляду за кімнатними рослинами»

Виконав: студент 2-го курсу, групи 2КН-23м
спеціальності 122 – «Комп'ютерні науки»

Бортник А. О.
(прізвище та ініціали)

Керівник: к.т.н., доцент каф. КН

Крилик Л. В.
(прізвище та ініціали)

«05» 12 2024 р.

Опонент: к.т.н., доцент каф. АІІТ

Богач І. В.
(прізвище та ініціали)

«05» 12 2024 р.

Допущено до захисту
Завідувач кафедри КН
д.т.н., проф. Яровий А. А.
(прізвище та ініціали)
«06» 12 2024 р.

Вінниця - 2024 року

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 122 Комп'ютерні науки
Освітньо-професійна програма – Системи штучного інтелекту

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
д.т.н., проф. Яровий А. А.
_____
(підпис)
«11» 10 2024 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Бортнику Анатолію Олеговичу
(прізвище, ім'я, по батькові)

1. Тема роботи: «Інформаційна технологія надання рекомендацій для догляду за кімнатними рослинами»
керівник роботи к.т.н., доц., доц. каф. КН, Крилик Л. В.
затверджені наказом вищого навчального закладу від «17» 09 2024 року № 310
2. Строк подання студентом роботи 11. 11. 2024 року
3. Вихідні дані до роботи: мова програмування – об'єктно-орієнтована; мінімальна кількість рекомендованих рослин – 5 шт.; час для отримання відповіді від сервера не має перевищувати 1 хв.; кількість рослин користувача – не менше 1000 шт.; інтерфейс користувача має бути інтуїтивно зрозумілим.
4. Зміст текстової частини: вступ, обґрунтування доцільності розробки інформаційної технології надання рекомендацій для догляду за кімнатними рослинами; моделювання інформаційної технології надання рекомендацій для догляду за кімнатними рослинами; програмна реалізація інформаційної технології надання рекомендацій для догляду за кімнатними рослинами; економічна частина, висновки, список використаних джерел, додатки.
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень): UML-діаграма послідовності взаємодії модулів інформаційної технології надання рекомендацій для догляду за кімнатними рослинами; клієнт-серверна архітектура; загальна структурна модель програмного модуля; схема алгоритму роботи інформаційної технології надання рекомендацій для догляду за кімнатними рослинами; схема алгоритму роботи системи надання рекомендацій по вибору кімнатної рослини; діаграма діяльності серверу під час запиту інформації про рослину; діаграма діяльності серверу під час перевірки чи існує користувач у системі; приклад роботи програми.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	виконан прийн
1–3	Крилик Л. В., к.т.н., доц. каф. КН		
4	Адлер О. О., к.т.н., доц. каф. ЕПВМ		

7. Дата видачі завдання 02.09 2024 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Обгрунтування доцільності розробки	02.09.24 - 09.09.24	Розділ 1
2	Моделювання інформаційної технології надання рекомендацій для догляду за кімнатними рослинами	10.09.24 - 29.09.24	Розділ 2
3	Програмна реалізація інформаційної технології надання рекомендацій для догляду за кімнатними рослинами	29.09.24 - 18.10.24	Розділ 3
4	Підготовка економічної частини	19.10.24 - 01.11.24	Розділ 4
5	Апробація результатів дослідження	02.11.24 - 04.11.24	тези доповіді, авторське право твір
6	Оформлення пояснювальної записки, графічного матеріалу та презентації	05.11.24 - 11.11.24	Пояснювальна записка, графічн матеріал, презентація

Студент _____ Бортник А.
(підпис) (прізвище та іні

Керівник роботи _____ Крилик Л.
(підпис) (прізвище та іні

АНОТАЦІЯ

УДК 004.42

Бортник А. О. Інформаційна технологія надання рекомендацій для догляду за кімнатними рослинами. Магістерська кваліфікаційна робота зі спеціальності 122 «Комп'ютерні науки», освітня програма «Системи штучного інтелекту». Вінниця: ВНТУ, 2024. 127 с.

Укр. мовою. Бібліогр.: 35 назв; рис.: 16; табл.: 9.

У магістерській кваліфікаційній роботі проведено аналіз предметної галузі догляду за кімнатними рослинами. Досліджено основні підходи надання рекомендацій для догляду за різними видами кімнатних рослин. Виконано аналіз аналогів, розглянуто методи та технології, які використовуються для вирішення подібних задач та обґрунтована доцільність розробки. Розроблено структурну модель інформаційної технології та математичну модель для підбору кімнатної рослини відповідно до уподобань користувача, що сприяє розширенню функціональних можливостей інформаційної технології, а також збільшує ефективність отримання рекомендацій.

Ця робота дає можливість користувачу отримати новий інструмент для аналізу стану своїх кімнатних рослин та інструмент для надання рекомендацій по вибору нових рослин зі зручним та інтуїтивно зрозумілим дизайном.

В роботі приведені результати досліджень та аналіз роботи розробленої інформаційної технології надання рекомендацій для догляду за кімнатними рослинами. Доведена економічна доцільність розробки.

Ілюстративна частина складається з 8 плакатів із результатами моделювання.

Ключові слова: кімнатні рослини, догляд за рослинами, рекомендації, інформаційна технологія, сервіси.

ABSTRACT

Bortnyk A. O. Information technology for providing recommendations for houseplant care. Master's thesis in the specialty 122 «Computer science», educational program «Artificial intelligence systems». Vinnytsia: VNTU, 2024. 127 p.

In Ukrainian language. Bibliogr.: 35 titles; fig.: 16; 9 tables.

The master's thesis analyzes the subject area of indoor plant care. The main approaches to providing recommendations for the care of different types of indoor plants are investigated. An analysis of analogues was performed, methods and technologies used to solve similar problems were considered, and the feasibility of development was substantiated. A structural model of information technology and a mathematical model for the selection of indoor plants according to user preferences have been developed, which helps to expand the functionality of information technology and also increases the efficiency of receiving recommendations.

This work provides users with a new tool for analyzing the condition of their houseplants and a tool for providing recommendations for selecting new plants with a convenient and intuitive design.

The study includes the results of research and analysis of the performance of the developed information technology for providing recommendations for houseplant care. The economic feasibility of the development is proven.

The illustrative part consists of 8 posters with modeling results.

Keywords: houseplants, plant care, recommendations, information technology, services.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному світі спостерігається стійке зростання популярності кімнатних рослин. Їхня привабливість, здатність очищати повітря та створювати атмосферу затишку роблять їх важливими елементами декору приміщень. Проте, догляд за рослинами може бути складним завданням, адже кожен вид має свої особливості. Кожен етап догляду за рослинами має важливий вплив на здоров’я рослини, тому важливо їх виконувати вчасно. Для цього потрібно робити нотатки та здійснювати пошук інформації про потрібні процедури.

Ті хто здійснюють пошук інформації про догляд за рослинами натрапляють на велику кількість різної інформації, інколи знайти потрібну інформацію дуже важко. Проте, не всі інформаційні ресурси надають повну інформацію. Нерідко власники рослин стикаються із суперечливими рекомендаціями, що ускладнює догляд та може призвести до їх загибелі.

Для вибору рослин потрібно витратити значну кількість часу, щоб знайти ті які подобаються. Навіть після того як рослину вибрано, потрібно шукати інформацію по догляду за нею. Для того, щоб допомогти початківцям в догляді за рослинами та вибрати ті які прийдуть їм до вподоби пропонується розробка інформаційної технології, яка буде надавати рекомендації по догляду за кімнатними рослинами. Така технологія буде надавати інформацію про необхідну кількість поливів, тип ґрунту, який дозволить рослині краще рости, необхідність підготовки рослини до нових сезонів тощо.

Зв'язок роботи з науковими програмами, планами, темами. Магістерська кваліфікаційна робота виконана відповідно до напрямку наукових досліджень кафедри комп’ютерних наук Вінницького національного технічного університету 22 К1 «Розробка прикладних інтелектуальних інформаційних технологій та систем».

Мета і завдання дослідження. Метою магістерської кваліфікаційної роботи є розширення функціональних можливостей інформаційної технології

надання рекомендацій для догляду за кімнатними рослинами за рахунок застосування математичної моделі надання рекомендацій по вибору нової кімнатної рослини. Це дозволить створити зручний клієнтський додаток, який надаватиме персоналізовані поради для догляду за різними видами кімнатних рослин, враховуючи їхні особливості та умови утримання. Розробка буде інтуїтивно зрозумілою та зможе конкурувати з іншими системами-аналогами на ринку.

Для досягнення поставленої мети потрібно розв'язати такі завдання**:**

- Провести аналіз існуючих систем надання рекомендацій для догляду за рослинами та обґрунтувати доцільність розробки нової інформаційної технології.
- Розробити математичну модель для надання рекомендацій з догляду за кімнатними рослинами, враховуючи різні параметри (освітлення, полив, температура, вологість тощо).
- Розглянути методи та технології, які можуть бути використані для надання персоналізованих рекомендацій для догляду за кімнатними рослинами.
- Спроектувати архітектуру та структуру програмного модуля для надання рекомендацій з догляду за кімнатними рослинами.
- Проаналізувати та обґрунтувати вибір інструментів для розробки.
- Виконати програмну реалізацію інформаційної технології надання рекомендацій для догляду за кімнатними рослинами.
- Провести тестування розробленої програми та проаналізувати отримані результати.
- Економічно обґрунтувати доцільність розробки інформаційної технології надання рекомендацій для догляду за кімнатними рослинами.

Об’єктом дослідження − є процес надання рекомендацій для догляду за кімнатними рослинами.

Предметом дослідження є програмні засоби проектування та реалізації інформаційної технології надання рекомендацій для догляду за кімнатними рослинами.

Методи дослідження. У процесі дослідження використовувались: дослідження сучасних методів створення експертних систем, дослідження методів обробки масивів інформації, дослідження методів аналізу отриманих даних, методи та підходи до розробки WEB-орієнтованих програмних додатків, методи об’єктно-орієнтованого програмування.

Наукова новизна одержаних результатів полягає в наступному: удосконалено інформаційну технологію надання рекомендацій для догляду за кімнатними рослинами, яка відрізняється від існуючих використанням математичної моделі надання рекомендацій по вибору нової кімнатної рослини та можливості контролю виконаних завдань по догляду. Використання цієї математичної моделі дозволило розширити функціональні можливості інформаційної технології, а також забезпечило підвищення ефективності та персоналізації рекомендацій для користувачів. Система надання рекомендацій по вибору нової кімнатної рослини враховує індивідуальні умови вирощування та преференції користувача, що дозволяє більш ефективно підібрати кімнатні рослини для конкретних умов та потреб.

Практичне значення одержаних результатів полягає у наступному:

1. Розроблено алгоритм функціонування інформаційної технології надання рекомендацій для догляду за кімнатними рослинами.

2. Розроблено діаграму послідовності взаємодії модулів інформаційної технології надання рекомендацій для догляду за кімнатними рослинами.

3. Здійснено програмну реалізацію інформаційної технології надання рекомендацій для догляду за кімнатними рослинами.

Достовірність теоретичних положень магістерської кваліфікаційної роботи підтверджується строгістю постановки задач, коректним застосуванням методів під час доведення наукових положень, порівнянням результатів із відомими та збіжністю результатів моделювання з результатами, що отримані під час впровадження розроблених програмних засобів.

Особистий внесок магістранта. Результати магістерської кваліфікаційної роботи отримані самостійно. В публікації у співавторстві

здобувачу належить дослідження перспектив розробки інформаційної технології надання рекомендацій для догляду за кімнатними рослинами.

Апробація результатів роботи. Результати досліджень було апробовано на LIII науково-технічній конференції підрозділів Вінницького національного технічного університету (НТКП ВНТУ–2024) м. Вінниці у 2024 р [1].

Публікації. За результатами магістерської кваліфікаційної роботи опубліковано тези доповідей на науково-технічній конференції [1] та отримано свідоцтво про реєстрацію авторського права на твір у формі комп'ютерної програми – «Інформаційна технологія надання рекомендацій для догляду за кімнатними рослинами» [2].

1 ОБҐРУНТУВАННЯ ДОЦІЛЬНОСТІ РОЗРОБКИ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ НАДАННЯ РЕКОМЕНДАЦІЙ ДЛЯ ДОГЛЯДУ ЗА КІМНАТНИМИ РОСЛИНАМИ

1.1 Аналіз предметної галузі кімнатних рослин

Кімнатні рослини – це рослини, які вирощуються в приміщеннях для декоративних цілей. Основна ідея полягає в тому, що вони дозволяють створити природне середовище всередині будинку чи офісу без необхідності виходити на вулицю. Водночас кімнатні рослини дають змогу покращити якість повітря та створити затишну атмосферу для людей, які живуть або працюють у приміщенні.

Кімнатні рослини адаптовані до життя в умовах обмеженого простору, освітлення та вологості. За своєю суттю, догляд за кімнатними рослинами базується на ідеї створення оптимальних умов для їх росту та розвитку, що включає правильний полив, освітлення, температурний режим та підживлення.

Вони можуть рости в горщиках різних розмірів і форм, що дозволяє розміщувати їх на підвіконнях, полицях, столах або підлозі. Ця універсальність робить кімнатні рослини ідеальними для декорування будь-якого простору, від маленьких квартир до великих офісних приміщень. Крім того, використання різних типів горщиків та підставок дозволяє створювати унікальні композиції, що відповідають індивідуальному стилю інтер'єру [3].

Перевагами кімнатних рослин є:

- Покращення якості повітря. Багато кімнатних рослин здатні поглинати шкідливі речовини з повітря, такі як формальдегід, бензол та трихлоретилен, тим самим очищаючи повітря в приміщенні;
- Естетичний вигляд. Кімнатні рослини додають краси та природності інтер'єру, створюючи приємну атмосферу та покращуючи загальний вигляд приміщення;
- Психологічний ефект. Догляд за рослинами може мати терапевтичний ефект, знижуючи стрес та покращуючи настрій;

- Різноманітність. Існує величезна кількість видів кімнатних рослин, що дозволяє підібрати рослини для будь-яких умов та смаків;

- Відносна невибагливість. Багато видів кімнатних рослин не потребують складного догляду і можуть довго жити при мінімальному догляді.

Недоліками є:

- Необхідність догляду. Хоча багато кімнатних рослин невибагливі, вони все ж потребують регулярного догляду, включаючи полив, обрізку та пересадку;

- Алергії. Деякі люди можуть бути алергічними на певні види рослин або пилок, який вони виробляють;

- Шкідники. Кімнатні рослини можуть приваблювати шкідників, таких як павутинні кліщі або щитівки, що вимагає додаткових заходів для їх контролю;

- Токсичність. Деякі популярні кімнатні рослини можуть бути токсичними для дітей або домашніх тварин при проковтуванні;

- Обмеження росту. В умовах обмеженого простору та освітлення деякі рослини можуть не досягати свого повного потенціалу росту або цвітіння [4].

За своєю суттю, догляд за кімнатними рослинами базується на ідеї створення оптимальних умов для їх росту та розвитку, що включає правильний полив, освітлення, температурний режим та підживлення. Це вимагає розуміння специфічних потреб кожного виду рослин. Наприклад, сукуленти потребують менше води і більше світла, тоді як папороті віддають перевагу вологому середовищу та помірному освітленню.

Правильний полив є ключовим аспектом догляду за кімнатними рослинами. Надмірний полив може призвести до гниття коренів, тоді як недостатній − до висихання рослини. Освітлення також відіграє важливу роль: деякі рослини потребують яскравого непрямого світла, тоді як інші можуть витримувати низький рівень освітлення.

Температурний режим важливий для забезпечення комфортного середовища росту. Більшість кімнатних рослин добре себе почувають при кімнатній температурі, але деякі види можуть потребувати конкретних температурних умов. Підживлення забезпечує рослини необхідними поживними

речовинами для здорового росту та цвітіння, особливо важливе в обмеженому середовищі горщика.

Крім того, догляд за кімнатними рослинами може включати такі процедури, як обрізка для підтримки форми та стимулювання росту, пересадка при перевищенні розміру горщика, а також боротьба зі шкідниками та хворобами. Ці практики допомагають підтримувати здоров'я та привабливий вигляд рослин протягом тривалого часу [5].

У сучасному світі кімнатні рослини набувають все більшого значення як елемент дизайну інтер'єру та засіб покращення якості життя в міському середовищі. Вони не тільки прикрашають простір, але й сприяють покращенню ментального здоров’я, підвищують продуктивність та створюють більш здорове середовище.

Найпопулярніша кімнатна рослина "Фікус Бенджаміна" в 2023 році досягла середньої висоти 2 метри при вирощуванні в домашніх умовах. Цікавою особливістю росту фікуса є закладена в його генетику здатність адаптуватися до умов освітлення, тобто з кожним новим листком рослина краще пристосовується до наявного рівня світла. Такі властивості дозволяють фікусу рости в різних умовах. Через популярність фікуса Бенджаміна, починають набирати популярність й інші види фікусів, які мають різні форми листя, розміри та вимоги до догляду.

Зараз на ринку існує велика кількість різновидів кімнатних рослин. Однак, більшість з них не користуються таким попитом, який є у фікуса Бенджаміна, оскільки вони можуть бути більш вимогливими до догляду або мати менш привабливий вигляд. Популярність багатьох нових видів кімнатних рослин залежить від трендів в інтер'єрному дизайні та соціальних мережах, а не від їх практичних властивостей, що дає можливість прогнозувати їх популярність за допомогою аналізу різних зовнішніх чинників, таких як згадки в Instagram чи Pinterest.

1.2 Огляд відомих програмних систем-аналогів

Першим аналогом для прогнозування надання рекомендацій за доглядом кімнатної рослини є «Blossom». Програма пропонує персоналізовані графіки поливу та догляду за рослинами на основі їх виду та умов вирощування. Додаток Blossom надає можливість створення «цифрового саду» з фотографіями та записами про кожну рослину, використовуючи при цьому алгоритми машинного навчання для адаптації рекомендацій з часом [6].

Blossom використовує складні алгоритми, які враховують не лише вид рослини, але й фактори навколишнього середовища, такі як вологість повітря, температура та освітлення. Це дозволяє створювати дуже точні графіки догляду. Крім того, додаток має функцію нагадувань, яка сповіщає користувачів про необхідність поливу, підживлення чи пересадки рослин. Також корисною є функція аналізу росту рослин: користувачі можуть регулярно завантажувати фотографії своїх рослин, а система аналізує їх стан та надає рекомендації щодо покращення догляду. Загальний вигляд інтерфейсного вікна сайту продемонстровано на рисунку 1.1.

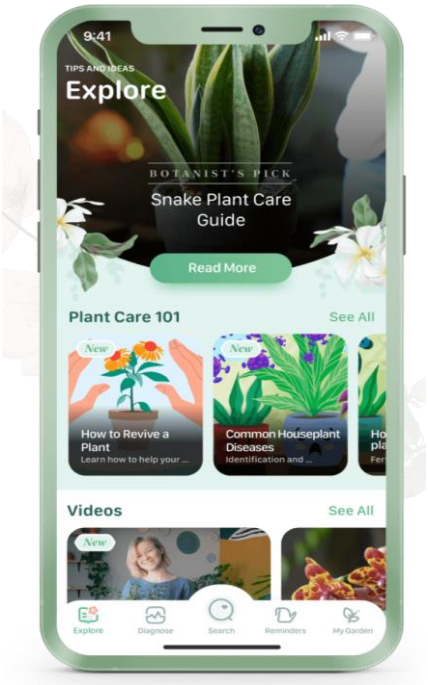

Рисунок 1.1 – Загальний вигляд інтерфейсного вікна програми Blossom

Другим додатком для ідентифікації та догляду за рослинами є PlantNet [7]. На сайті стверджується, що для ідентифікації використовується технологія комп'ютерного зору та штучного інтелекту на основі великої бази даних рослин. Вони використовують такі фактори як: форма листя, структура квітів, текстура кори та інші візуальні характеристики рослин. PlantNet також пропонує функцію «громадської науки», де користувачі можуть допомагати покращувати точність ідентифікації.

PlantNet має велику базу даних, яка містить інформацію про понад 20 000 видів рослин. Додаток дозволяє користувачам робити фотографії різних частин рослини для більш точної ідентифікації. Після визначення виду рослини, PlantNet надає детальну інформацію про її характеристики, умови вирощування та потенційні проблеми зі здоров'ям. Також у ньому є можливість порівняння схожих видів, що допомагає користувачам краще розуміти тонкощі класифікації рослин. Загальний вигляд інтерфейсного вікна сайту продемонстровано на рисунку 1.2.

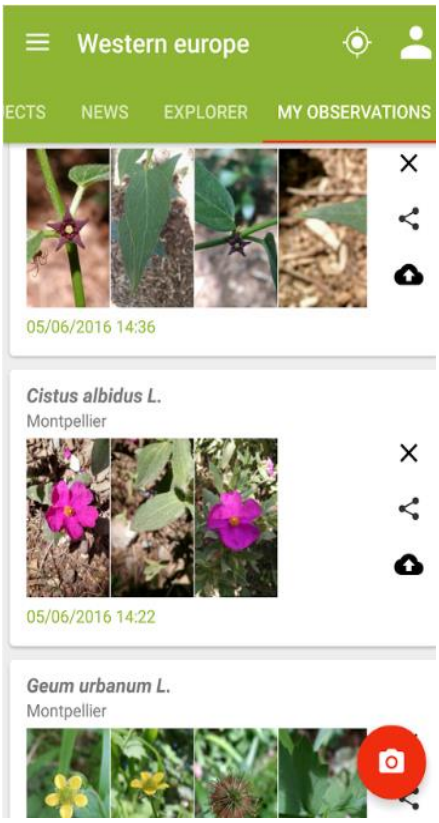

Рисунок 1.2 – Загальний вигляд інтерфейсного вікна програми PlantNet

Третім додатком для планування та догляду за садом є Gardenia [8]. Додаток пропонує комплексний підхід до садівництва, включаючи інструменти для планування саду, ведення щоденника рослин та отримання експертних порад. Gardenia використовує такі функції як: створення віртуальних макетів саду, персоналізований календар садових робіт та поради щодо поєднання рослин для створення гармонійних композицій.

Gardenia має інструмент планування саду, який дозволяє користувачам створювати 3D-моделі своїх садів. Це допомагає візуалізувати, як різні рослини будуть виглядати разом, враховуючи їх розміри при повному зростанні. Додаток також має обширну базу даних рослин з детальною інформацією про їх вимоги до ґрунту, світла та води. Особливо корисною є функція сезонних порад, яка надає рекомендації для догляду за садом відповідно до пори року та кліматичної зони користувача. Загальний вигляд інтерфейсного вікна сайту продемонстровано на рисунку 1.3.

Рисунок 1.3 – Загальний вигляд інтерфейсного вікна програми Gardenia

Недоліком є те, що користувачу не може обрати рослину на основі його уподобань. Цей недолік значно обмежує корисність додатків для нових користувачів або тих, хто шукає ідеї для розширення своєї колекції рослин. Вирішення цієї проблеми могло б значно підвищити привабливість та практичну цінність додатку для догляду за рослинами.

1.3 Постановка задачі дослідження

Поєднання традиційного садівництва з сучасними технологіями все частіше набуває широкого застосування та удосконалює догляд за кімнатними рослинами. Гарною колаборацією ботаніки та ІТ в наш час є «розумні» системи поливу, які з кожним роком стають більш доступними та функціональними [9].

Проблема полягає у відсутності простого, безкоштовного і водночас багатофункціонального та ефективного інструменту для надання персоналізованих рекомендацій по догляду за кімнатними рослинами.

Загальний опис функціоналу інформаційної технології для надання рекомендацій по догляду за кімнатними рослинами:

1. Реалізація входу до інформаційної системи за допомогою графічного інтерфейсу.
2. Реалізація вибору видів рослин та умов вирощування.
3. Реалізація виведення персоналізованих рекомендацій за допомогою графічного інтерфейсу.
4. Реалізація виведення історії догляду за рослинами користувача.

У цій роботі потрібно реалізувати інформаційну технологію для надання рекомендацій по догляду за кімнатними рослинами. Інформаційна технологія буде використовуватись користувачами для отримання оптимізованих графіків поливу, підживлення та інших процедур догляду за кімнатними рослинами. Таким чином, використання такої розробки дасть можливість користувачу отримати новий інструмент для аналізу стану своїх рослин та інструмент для перевірки ефективності різних стратегій догляду.

Інформаційна технологія, що розробляється, перш за все орієнтована на точність та зручність рекомендацій. Точність полягає у можливості отримати більш персоналізовані рекомендації ніж в існуючих програм-аналогів. Зручність полягає у можливості вибрати необхідні для користувача види рослин, умови вирощування та можливістю перегляду власної історії догляду для аналізу отриманих результатів.

Тому, для вирішення проблеми надання рекомендацій по догляду за кімнатними рослинами потрібно вирішити такі основні завдання:

1. Здійснити проектування структури інформаційної технології надання рекомендацій по догляду за кімнатними рослинами.
2. Розробити математичну модель оптимізації графіків догляду за кімнатними рослинами.
3. Реалізувати програмний модуль інформаційної технології.
4. Провести тестування інформаційної технології.

Отже, потрібно розробити інформаційну технологію, в якій буде реалізовано алгоритм надання персоналізованих рекомендацій по догляду за кімнатними рослинами з достатньою точністю.

1.4 Висновок до розділу 1

В першому розділі було розглянуто основні поняття догляду за кімнатними рослинами, проведено аналіз технічних рішень програм для надання рекомендацій по догляду за кімнатними рослинами та доведена доцільність розробки інформаційної технології надання рекомендацій для догляду за кімнатними рослинами. Розглянуто системи-аналоги: «Blossom», «Gardenia» та «PlantNet». Встановлено, що існуючі рішення такої задачі представлені у вигляді комерційного сервісу і не мають системи надання рекомендацій для вибору нових рослин, обмежуючи користувача певними рамками рекомендацій.

Вся описана та проаналізована вище інформація і стала передумовою для розробки інформаційної технології надання рекомендацій для догляду за

кімнатними рослинами, яка дозволить дати користувачам більше інформації для ефективного догляду за своїми рослинами. Задача надання рекомендацій для догляду за кімнатними рослинами є дуже схожою до задачі надання рекомендацій для садівництва, проте має і деякі власні відмінності. Вплив умов вирощування на стан кімнатних рослин є достатньо суттєвим, тому створення інформаційної технології надання рекомендацій для догляду за кімнатними рослинами є доцільним.

Визначено задачі дослідження, вимоги клієнта, які допоможуть в подальшому відобразити структуру системи, підібрати доцільні технології та мови програмування для реалізації інформаційної технології надання рекомендацій для догляду за кімнатними рослинами.

2 МОДЕЛЮВАННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ НАДАННЯ РЕКОМЕНДАЦІЙ ДЛЯ ДОГЛЯДУ ЗА КІМНАТНИМИ РОСЛИНАМИ

2.1 Обґрунтування вибору методів реалізації інформаційної технології надання рекомендацій для догляду за кімнатними рослинами

У сучасному світі, де технології все більше проникають у наше повсякденне життя, догляд за кімнатними рослинами також стає все більш технологічно підкованим. Розвиток методів штучного інтелекту, великих даних, та автоматизованих систем дозволяє навіть любителям-садівникам отримувати персоналізовані та точні рекомендації для догляду за своїми зеленими улюбленцями. Застосування сучасних технологій може значно полегшити догляд за рослинами, допомагаючи уникати помилок та підвищувати якість вирощування. Існує кілька основних методів, які використовуються при створенні систем рекомендацій по догляду за рослинами. Кожен з цих підходів має свої унікальні характеристики, переваги і обмеження, які потрібно враховувати при виборі відповідної стратегії.

1. Фундаментальний аналіз. Цей метод базується на глибоких ботанічних знаннях і є найнадійнішим джерелом науково обґрунтованих рекомендацій. Цей підхід враховує специфічні потреби кожного виду рослин, наприклад, їхні природні умови зростання, вимоги до ґрунту, води, світла, температури і вологості повітря. Багаторічні дослідження в галузі ботаніки дають можливість систематизувати знання про різні види рослин, що робить фундаментальний аналіз дуже точним інструментом для складання рекомендацій.

Однак, цей метод має свої недоліки. Оскільки фундаментальний аналіз не завжди враховує індивідуальні умови вирощування в конкретному приміщенні, такі як мікроклімат, рівень вологості або освітлення, користувачеві може бути складно застосувати загальні поради до своїх власних умов. Крім того, цей підхід потребує великих знань ботаніки і може бути складним для початківців [10].

2. Метод аналізу користувацького досвіду. Цей підхід враховує реальний досвід вирощування та має значну перевагу в тому, що він адаптований до реальних ситуацій, з якими стикаються любителі кімнатних рослин. Користувацький досвід часто містить практичні поради, які можуть бути не включені в академічну ботанічну літературу, і дозволяє швидко ідентифікувати найпоширеніші проблеми і способи їх вирішення. Він також легко масштабується за допомогою соціальних мереж, форумів або спеціалізованих додатків, де користувачі можуть обмінюватися своїм досвідом. Використання технологій аналізу великих даних дозволяє обробляти значні обсяги інформації, що дозволяє отримувати рекомендації на основі тенденцій та індивідуальних результатів користувачів.

Проте, метод аналізу користувацького досвіду має певні обмеження. Оскільки інформація може бути суб'єктивною або містити неточності, це може призвести до появи помилкових рекомендацій. Крім того, активність користувачів у соціальних мережах не завжди стабільна, що може вплинути на кількість та якість отриманих даних.

3. Системний підхід. Системний підхід до догляду за рослинами полягає в комплексному врахуванні всіх факторів, які впливають на ріст та розвиток рослини. Цей метод розглядає не тільки окремі параметри, такі як світло чи полив, але й взаємозв'язки між ними. Наприклад, системний підхід дозволяє враховувати, як зміна температури вплине на необхідність у вологості або як різні рівні освітлення впливають на частоту поливу.

Такий підхід дозволяє отримати цілісне розуміння процесу догляду за рослинами та створювати комплексні рекомендації, які враховують широкий спектр умов. Однак, він може бути складним для реалізації, оскільки потребує великої кількості вхідних даних та їхньої детальної обробки. Крім того, для простих ситуацій цей метод може виявитися надмірно складним, особливо для рослин, що не потребують особливого догляду.

4. Нейронні мережі. Цей метод використовує передові технології машинного навчання для аналізу великих обсягів даних і виявлення неочевидних

закономірностей. Нейронні мережі здатні адаптуватися до нових даних та умов, що робить їх особливо цінними для створення динамічних систем рекомендацій. Вони можуть враховувати безліч факторів одночасно і виявляти складні взаємозв'язки. Однак, цей метод вимагає значних обчислювальних ресурсів і великої кількості якісних даних для навчання. Крім того, результати роботи нейронних мереж можуть бути складними для інтерпретації.

5. Математичні моделі часових рядів застосовуються для прогнозування різних аспектів догляду за рослинами, таких як частота поливу, підживлення або зміни в рості протягом певного періоду. Цей підхід дозволяє передбачати довгострокові тенденції в догляді за рослинами на основі минулих даних.

Моделі часових рядів можуть бути автоматизовані і інтегровані у системи догляду, що значно полегшує процес планування регулярного догляду за рослинами. Наприклад, модель може передбачити, коли рослина потребуватиме наступного поливу або пересадки, що робить догляд більш організованим і менш залежним від людського фактору.

Проте, математичні моделі не завжди враховують непередбачувані фактори, такі як раптові зміни температури чи вологості. Вони також можуть бути менш ефективними для рослин із нерегулярними циклами росту, оскільки не завжди можуть точно передбачити їх поведінку.

6. Експертні системи − це програмні комплекси, які імітують процес прийняття рішень людиною-експертом у конкретній галузі знань. У контексті догляду за кімнатними рослинами такі системи можуть надавати детальні і точні рекомендації, ґрунтуючись на великому обсязі структурованих знань і правил. Вони можуть враховувати різні фактори, такі як тип рослини, освітлення, температура, вологість, тип ґрунту, сезон та інші умови.

Експертні системи дозволяють користувачам отримувати високоякісні поради з догляду за рослинами, які враховують їх індивідуальні потреби. Наприклад, система може рекомендувати, коли і як часто поливати рослину, який вид добрив використовувати або коли її пересаджувати [11].

Кожен з розглянутих методів має свої сильні сторони та обмеження. Для створення ефективної системи рекомендацій по догляду за кімнатними рослинами оптимальним рішенням може бути комбінація різних підходів. Наприклад, фундаментальний аналіз може забезпечити базові знання про потреби рослин, які потім можуть бути доповнені даними з аналізу користувацького досвіду. Системний підхід може допомогти інтегрувати ці дані в цілісну картину.

Важливо також враховувати, що вибір методу залежить від конкретних цілей, наявних ресурсів та цільової аудиторії системи рекомендацій. Для простих додатків, орієнтованих на початківців, може бути достатньо базового фундаментального аналізу та аналізу користувацького досвіду. Для більш складних систем, орієнтованих на професійних садівників або дослідницькі цілі, може бути доцільним використання більш складних методів, таких як нейронні мережі або системний підхід.

Переваги та недоліки розглянутих методів можна представити за допомогою таблиці. 2.1.

Кожен з розглянутих методів має свої переваги та обмеження. Для створення ефективної системи рекомендацій по догляду за кімнатними рослинами оптимальним рішенням буде комбінація різних підходів. Фундаментальний аналіз може забезпечити надійну наукову основу, на яку можна спиратися при створенні базових рекомендацій, а метод аналізу користувацького досвіду дозволить адаптувати ці рекомендації до реальних умов вирощування. Системний підхід допоможе врахувати всі взаємозв'язки між різними факторами догляду, тоді як нейронні мережі та математичні моделі можуть забезпечити динамічні та довгострокові прогнози.

Важливо також зазначити, що вибір методу може залежати від конкретної ситуації: для домашнього користувача може бути достатньо простих рекомендацій, заснованих на аналізі користувацького досвіду, тоді як для професійного садівництва може знадобитися більш складний підхід з використанням всіх доступних методів. Тобто проблема надання рекомендацій

по догляду за кімнатними рослинами залишається відкритою та актуальною для вивчення.

Включення експертних систем у комплексний підхід до створення рекомендацій по догляду за кімнатними рослинами може значно підвищити якість та надійність порад, особливо для складних або рідкісних видів рослин, де досвід реальних користувачів може бути обмеженим.

Таблиця 2.1 – Переваги та недоліки методологій надання рекомендацій по вибору кімнатної рослини

Метод	Переваги	Недоліки
Фундаментальний аналіз	Базується на глибоких ботанічних знаннях. Враховує специфічні потреби кожного виду рослин. Надає надійні, науково обґрунтовані рекомендації	Не завжди враховує індивідуальні умови вирощування.
Метод аналізу користувацького досвіду	Враховує реальний досвід вирощування рослин. Може виявляти практичні поради, не описані в літературі. Легко масштабується з використанням аналізу соціальних мереж.	Може містити суб'єктивні або неточні дані Залежить від активності користувачів у соціальних мережах Може не враховувати рідкісні види рослин.
Системний підхід	Розглядає всі аспекти догляду в комплексі. Дозволяє виявити взаємозв'язки між різними факторами. Забезпечує цілісне розуміння процесу догляду за рослинами.	Може бути складним для імплементації. Вимагає великої кількості даних для аналізу. Може бути надмірно складним для простих випадків.
Експертні системи	Висока точність рекомендацій. Здатність пояснювати рішення. Можливість роботи з неповними даними	Складність створення та оновлення. Можливість конфліктів правил. Складність роботи з нечіткими даними

Продовження таблиці 2.1

Метод	Переваги	Недоліки
Нейронні мережі	Здатність аналізувати великі обсяги даних. Може виявляти неочевидні закономірності. Адаптується до нових даних та умов.	Вимагає значних обчислювальних ресурсів. Потребує великої кількості якісних даних для навчання. Результати можуть бути складними для інтерпретації.
Математичні моделі часових рядів	Забезпечують точні, кількісні прогнози. Дозволяють виявляти довгострокові тенденції. Можуть бути легко автоматизовані.	Можуть не враховувати непередбачувані фактори. Вимагають регулярного збору даних. Можуть бути менш ефективними для рослин з нерегулярними циклами росту.

2.2 Розробка математичної моделі надання рекомендацій по вибору рослини

Розробка оптимальної математичної моделі для надання рекомендацій щодо вибору кімнатних рослин є ключовим етапом у розробці ефективної інформаційної системи. Цей процес вимагає ретельного аналізу та врахування багатьох факторів, що впливають на успішне вирощування рослин у домашніх умовах [12].

При розробці такої моделі необхідно враховувати складність та багатогранність завдання. Кожна кімнатна рослина має свої унікальні вимоги до умов утримання, які можуть значно відрізнятися залежно від виду, сорту та навіть індивідуальних особливостей конкретного екземпляра. Водночас, умови в кожному приміщенні також є унікальними, враховуючи такі фактори як освітленість, вологість, температура, наявність протягів тощо. Розглянемо детальніше три ключові аспекти, які лежать в основі створення такої моделі: багатокритеріальний підхід, використання вагових коефіцієнтів та нормалізацію

даних. Кожен з цих елементів відіграє важливу роль у забезпеченні ефективності та точності системи рекомендацій.

Багатокритеріальна модель дозволяє враховувати різноманітні фактори, які впливають на вибір рослини. Це можуть бути не лише фізичні параметри середовища, такі як освітленість, вологість і температура, але й характеристики самої рослини − її розмір, швидкість росту, естетичні якості. Також важливо врахувати фактори, пов'язані з доглядом − частоту поливу, необхідність обрізки, стійкість до шкідників тощо [13]. Використання функції S дозволяє гнучко підходити до оцінки придатності рослини до уподобань користувача:

$$S = f(L, H, T, R, C, \dots), \tag{2.1}$$

де L, H, T, R, C – це величини критеріїв, які вносяться у багатокритеріальну модель.

Ця функція може бути лінійною, якщо вплив факторів є адитивним, або нелінійною, якщо існують складні взаємозв'язки між параметрами [14]. Наприклад, деякі рослини можуть компенсувати недостатнє освітлення за рахунок більш інтенсивного поливу, і це можна врахувати у нелінійній функції.

Використання вагових коефіцієнтів є покращенням багатокритеріальної моделі, яка дозволяє надати різну важливість різним критеріям. Це особливо корисно, коли в системі рекомендацій потрібно враховувати індивідуальні переваги користувачів [15]. Наприклад, для одного користувача може бути критично важливим, щоб рослина не вимагала частого поливу, в той час як для іншого головним критерієм може бути естетична привабливість.

Визначення оптимальних вагових коефіцієнтів може бути здійснено різними методами. Один з підходів − це проведення опитування користувачів і статистичний аналіз їхніх переваг. Інший метод − це використання експертних оцінок, коли досвідчені садівники або ботаніки визначають важливість різних факторів. Також можливе застосування методів машинного навчання для

автоматичного підбору вагових коефіцієнтів на основі історії успішних виборів рослин [16].

Нормалізація даних є критично важливим етапом у процесі обробки інформації для системи рекомендацій для вибору кімнатних рослин. Цей метод забезпечує коректність розрахунків шляхом приведення різнорідних параметрів до єдиної шкали.

Суть методу полягає в трансформації вхідних даних таким чином, щоб всі параметри мали однаковий діапазон значень, незалежно від їхніх початкових одиниць вимірювання чи масштабу. Це дозволяє уникнути ситуації, коли параметри з більшим діапазоном значень непропорційно впливають на кінцевий результат [17].

Min-max нормалізація є простим і ефективним методом, але вона чутлива до викидів. Альтернативою може бути z-score нормалізація, яка враховує середнє значення і стандартне відхилення [18]:

$$X_{norm} = \frac{X-\mu}{\sigma}, \quad (2.2)$$

де μ − середнє значення,

σ − стандартне відхилення.

Важливо також враховувати, що деякі параметри можуть мати оптимальні діапазони, а не просто «чим більше, тим краще». Наприклад, для багатьох рослин існує оптимальний діапазон температур, і відхилення в будь-яку сторону є небажаним. У таких випадках може бути доцільним використання функцій приналежності з теорії нечітких множин. Реалізація такої моделі вимагає створення бази даних рослин з їхніми характеристиками та вимогами до умов утримання.

Метод аналізу ієрархій (MAI) є потужним інструментом для прийняття рішень у багатокритеріальних задачах. У контексті вибору кімнатних рослин, цей метод може бути використаний для визначення відносної важливості різних

критеріїв. MAI передбачає створення ієрархічної структури критеріїв та проведення попарних порівнянь між ними [19]. Математично це можна представити у вигляді матриці попарних порівнянь:

$$A = a_{ij}\ , \tag{2.3}$$

де a_{ij} − відносна важливість критерію *i* порівняно з критерієм *j*.

Цей метод дозволяє враховувати як об'єктивні характеристики рослин, так і суб'єктивні переваги користувачів.

Іншим підходом є використання методів нечіткої логіки. Нечітка логіка особливо корисна при роботі з лінгвістичними змінними та невизначеностями, які часто присутні в описах умов догляду за рослинами. Наприклад, поняття «яскраве освітлення» може бути представлене як нечітка множина з функцією приналежності:

$$\mu_{\text{яскраве}} = \begin{cases} 0, \text{якщо x} \leq 1000 \text{ лк}, \\ \frac{\text{x}}{1000}, \text{якщо } 1000 \text{ лк} < \text{x} < 2000 \text{ лк}, \\ 1, \text{якщо } x \geq 2000 \text{ лк}. \end{cases} \tag{2.4}$$

Використання нечітких множин дозволяє більш гнучко описувати вимоги рослин та умови в приміщенні, що підвищує адаптивність системи рекомендацій.

Для системи рекомендацій щодо вибору кімнатних рослин було обрано багатокритеріальну лінійну модель це обумовлено рядом факторів, які роблять її оптимальним рішенням для поставленого завдання. Розглянемо ці фактори:

1. Перший фактор. Лінійна модель відрізняється своєю простотою та інтерпретованістю. Вона дозволяє легко зрозуміти вплив кожного окремого критерія на кінцевий результат, що є важливим як для розробки так і для кінцевих користувачів.

2. Другий фактор. Лінійна модель є достатньо гнучкою для врахування широкого спектру критеріїв, які впливають на вибір кімнатних рослин. Ми можемо включити в модель такі параметри як освітленість, вологість,

температура, розмір рослини, складність догляду та багато інших. При цьому, завдяки лінійності моделі, додавання нових факторів не призводить до значного ускладнення розрахунків [20].

3. Третім фактором є можливість легкого оновлення та адаптації моделі. У міру накопичення нових даних про рослини та умови їх вирощування, можна легко коригувати коефіцієнти моделі, не змінюючи її загальну структуру. Це забезпечує гнучкість системи та її здатність враховувати нові знання та тенденції в галузі догляду за кімнатними рослинами.

4. Четвертим аргументом на користь вибору лінійної моделі є її обчислювальна ефективність. Лінійні розрахунки виконуються швидко навіть на великих обсягах даних, що дозволяє системі працювати в режимі реального часу, надаючи миттєві рекомендації користувачам.

Таким чином, багатокритеріальна лінійна модель надає оптимальний баланс між простотою, гнучкістю, інтерпретованістю та ефективністю, що робить її ідеальним вибором для системи рекомендацій для вибору кімнатних рослин. Вона дозволяє врахувати всі важливі аспекти вибору рослин, забезпечуючи при цьому зрозумілість та прозорість рекомендацій для користувачів.

Оскільки обираємо рослину, вона має найкраще відповідати заданим умовам середовища та потребам користувача. Кожна рослина має свої вимоги та характеристики, які можуть бути представлені через ряд критеріїв:

1. **Розмір рослини (x_1)**: Описує бажаний максимальний розмір рослини в зрілому віці.

2. **Насиченість кольору (x_2)**: Відображає рівень бажаної насиченості кольору листя або квіток

3. **Комфортна температура (x_3)**: Діапазон температур, у яких рослина буде рости оптимально.

4. **Тип рослини (x_4)**: Категорія рослини, яку шукає користувач.

5. **Частота поливу (x_5)**: Як часто користувач готовий поливати рослину.

6. **Складність догляду (x_6)**: Рівень догляду, який користувач готовий забезпечити.

7. **Кількість світла (x_7)**: Рівень освітлення, необхідний для рослини.

Тепер розглянемо саму модель представлену у наступному рівнянні:

$$S_i = \sum_{j=1}^{n} f_i\left(x_{i,j}, x_i^{user}\right), \quad (2.5)$$

де n − кількість критеріїв,

x_i^{user} – значення j-го критерію, вибране користувачем,

$x_{i,j}$ − значення j-го критерію для рослини i,

$f_i(x_{i,j}, x_i^{user})$ – функція, яка обчислює відповідність конкретного критерію,

$f_i\left(x_{i,j}, x_i^{user}\right) = 1$, якщо значення критерію рослини $x_{i,j}$ відповідає вибору користувача x_i^{user},

$f_i\left(x_{i,j}, x_i^{user}\right) = 0$, якщо значення не відповідає.

Загальний показник відповідності S_i для кожної рослини − це кількість критеріїв, які відповідають вибору користувача. Рослина, для якої S_i є максимальним, буде рекомендована як найкращий варіант.

2.3 Розробка UML-діаграми послідовності інформаційної технології надання рекомендацій для догляду за кімнатними рослинами

Для ефективнішої розробки та розуміння роботи програмного модуля інформаційної технології надання рекомендацій для догляду за кімнатними рослинами розробимо та розглянемо UML-діаграму. Розглянемо для початку що являє собою UML-діаграма.

UML – це уніфікована мова моделювання, яка використовується для візуалізації роботи системи. Інженери програмного забезпечення створюють UML- діаграми для розуміння конструкції, архітектури коду та щоб ефективніше

реалізовувати складні програмні системи. UML-діаграми також використовуються для моделювання робочих та бізнес процесів.

При створені програмного забезпечення може створюватись сотні функцій та тисячі рядків коду, що ускладнює його розуміння. UML-діаграма спрощує цю інформацію перетворюючи її у візуальне посилання, яке легше сприймати. В UML-діаграмах використовується стандартизований метод для написання моделі системи та захоплення концептуальних ідей [21].

Крім коду, UML-діаграми також гарно підходять для візуалізації зв'язків та ієрархій у програмних компонентах, подібно до дерева рішень або блок-схеми − але специфічно для програмного забезпечення.

Існує дві категорії UML-діаграм: структурні діаграми та діаграми поведінки:

1. Структурні діаграми візуалізують компоненти, які складають систему, і взаємозв'язок між ними, показуючи статичні аспекти системи.

2. Поведінкові діаграми представляють те, що відбувається всередині системи, включаючи те, як компоненти взаємодіють один з одним та з іншими системами або користувачами.

В свою чергу ці дві категорії розділяються на типи. Розглянемо кілька конкретних типів структурних UML- діаграм:

- UML-діаграма класів важлива для об'єктно-орієнтованого проектування, візуально представляючи статичну структуру системи. Вона показує класи, їхні атрибути, операції та зв'язки за допомогою прямокутників. Наприклад, розробники програмного забезпечення та зацікавлені сторони бізнесу використовують ці діаграми, щоб візуалізувати структуру та взаємодію в програмному додатку.

- Діаграма залежностей організовує та показує залежності між проектами, які можуть містити різні елементи, такі як класи, варіанти використання та інші проекти. Можна уявити що це як організація різних частин системи в папки, це полегшує керування та розуміння загальної архітектури.

- Діаграми об’єктів візуалізують конкретні екземпляри об’єктів та їхні зв’язки в певний момент часу. Вони зазвичай використовують приклади з реального світу. Фактично, вони також відомі як діаграми екземплярів, оскільки вони відображають, як виглядає система в певний момент часу.

- Діаграми компонентів візуалізують внутрішню структуру класу, включаючи його частини, інтерфейси та спільну роботу. Вони дають уявлення про те, як компоненти взаємодіють у більшій архітектурі системи, допомагаючи зрозуміти організацію та залежності елементів програмного забезпечення.

- Діаграми розгортання показують фізичне розгортання програмних компонентів на вузлах у розподіленій системі. Вони показують, як програмне та апаратне забезпечення взаємодіють, і є важливими для планування розгортання системи в різних середовищах.

Тепер розглянемо приклади поведінкових UML-діаграм:

- Діаграми діяльності − це, блок-схеми, які зображують потік діяльності в системі. Вони зображують послідовність дій, точки прийняття рішень і одночасну діяльність, що робить їх корисними для моделювання бізнес-процесів і поведінки системи.

- UML-діаграми послідовності показують, як об’єкти взаємодіють з часом, візуалізуючи порядок повідомлень, якими обмінюються об’єкти. Вони особливо ефективні для ілюстрації динамічної взаємодії між різними компонентами системи. Вони часто використовуються щоб зрозуміти, як структурувати нову систему чи вдосконалити існуючий процес

- Подібно до діаграм послідовності, діаграми зв’язку зосереджені на повідомленнях, якими обмінюються об’єкти для досягнення певних функцій. Вони надають ширший погляд на системну взаємодію та співпрацю між об’єктами.

- Оглядові діаграми взаємодії поєднують елементи діаграм діяльності та послідовності, щоб забезпечити високорівневе уявлення про взаємодію системи. Вони використовують фрейми для представлення дій і взаємодій, пропонуючи повний погляд на динаміку системи та робочі процеси.

- Часові діаграми показують поведінку об’єктів у певних проміжках часу, показуючи час та тривалість подій і взаємодій. Вони корисні для розуміння часових обмежень та аспектів продуктивності системних операцій.

- Діаграми варіантів використання візуалізують взаємодію між акторами (користувачами або системами) і варіанти використання (функціональні вимоги) у системі. Вони надають графічний огляд функціональності системи та допомагають зацікавленим сторонам зрозуміти зв’язки та залежності між різними варіантами використання.

- Діаграми кінцевого стану моделюють поведінку об’єктів, коли вони переміщуються через стани у відповідь на події. Вони зображають переходи станів, дії та умови, надаючи уявлення про те, як компоненти поводяться в різних сценаріях.

Для розгляду роботи інформаційної технології було обрано діаграму послідовності, оскільки вона дозволяє детально проаналізувати та візуалізувати повну взаємодію користувача з системою та показує всі етапи, які проходить певна дія. Діаграма послідовності міститиме основні компоненти, які описують повний цикл роботи програмного модуля, а саме: WEB-клієнт, що забезпечує інтерфейс користувача та обробку користувацьких дій; сервер, який відповідає за обробку запитів та реалізацію бізнес-логіки; постачальник даних, що керує отриманням та перетворенням даних; база даних для зберігання та управління інформацією. Також застосуємо об'єкт користувач для відображення взаємодії користувача з іншими модулями програми, це дозволить краще зрозуміти послідовність дій та обмін даними в системі.

Розроблена UML-діаграма послідовностей зображена на рисунку 2.1.

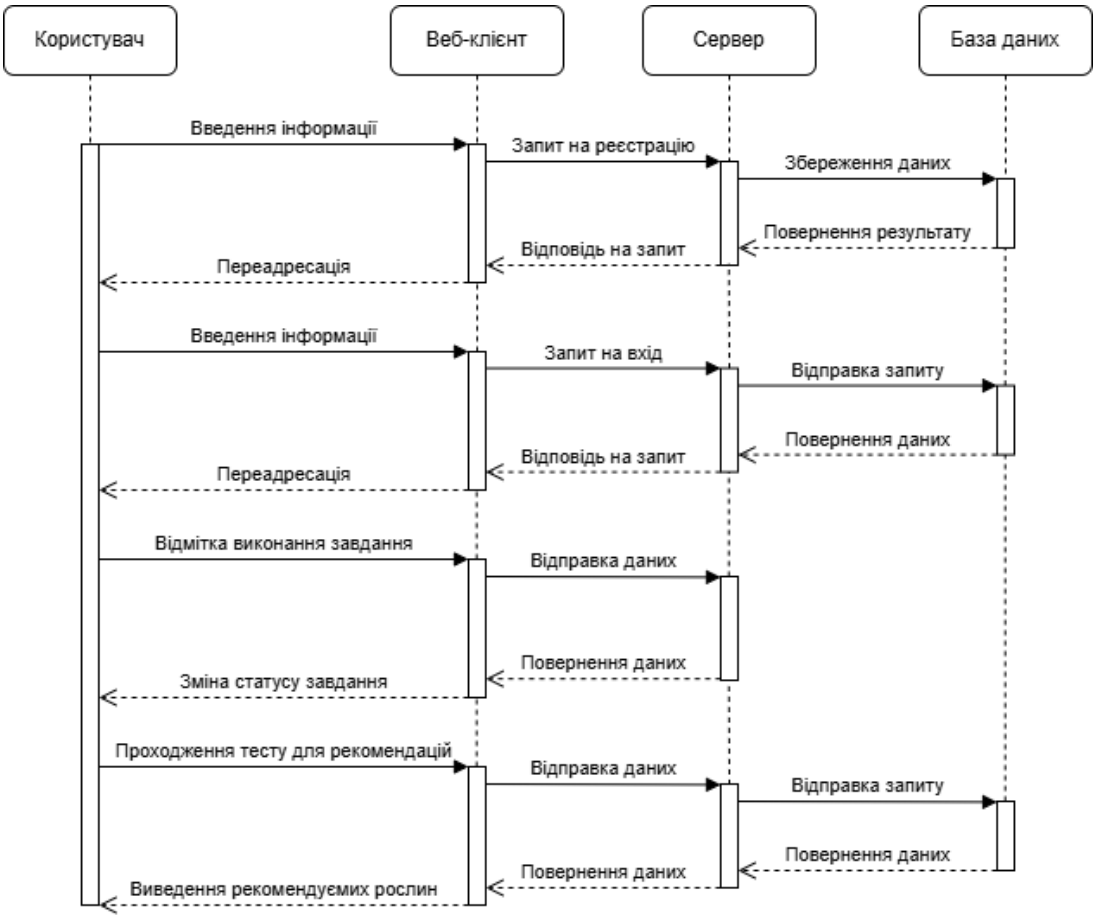

Рисунок 2.1 – UML-діаграма послідовності взаємодії модулів інформаційної технології надання рекомендацій для догляду за кімнатними рослинами

2.4 Висновок до розділу 2

У другому розділі було розглянуто методи реалізації інформаційної технологій надання рекомендацій для догляду за кімнатними рослинами.

Удосконалено математичну модель програмного модуля надання рекомендацій для вибору кімнатної рослини на основі лінійної моделі. Модель відповідає за генерування рекомендованих кімнатних рослин на основі переданої користувачем інформації. Розроблено UML-діаграму послідовності для опису роботи програмного модуля інформаційної технології надання рекомендацій для догляду за кімнатними рослинами.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ НАДАННЯ РЕКОМЕНДАЦІЙ ДЛЯ ДОГЛЯДУ ЗА КІМНАТНИМИ РОСЛИНАМИ

3.1 Обґрунтування вибору архітектури інформаційної технології надання рекомендацій для догляду за кімнатними рослинами

Вибір архітектури інформаційної технології надання рекомендацій для догляду за кімнатними рослинами є важливим кроком, що впливає на загальну структуру системи, її продуктивність, масштабованість та можливості розширення. Розглянемо архітектуру клієнт-сервер, яка широко застосовується при роботі з базами даних в мережі, відома вже давно і найчастіше застосовувалась у великих організаціях. Сьогодні ця технологія все частіше використовується, оскільки в світі нагромаджено величезну кількість інформації по різноманітних питаннях і найчастіше ця інформація зберігається в базах даних [22].

Архітектура клієнт-сервер – це концепція інформаційної мережі, в якій основна частина її ресурсів зосереджена в серверах, обслуговуючих своїх клієнтів. Розглянута архітектура визначає два типи компонентів: сервери та клієнти. Розглянемо їх детальніше:

Сервер – це об’єкт, що дає відповідь іншим об’єктам мережі за їх запитами. Він працює за завданнями клієнтів та управляє виконанням їх завдань. Після виконання кожного завдання сервер надсилає отримані результати клієнту, який надіслав це завдання [23].

Процес, який викликає сервісну функцію за допомогою певних операцій, називається клієнтом. Ним може бути програма або користувач. На рисунку 3.1 зображено приклад архітектури клієнт-сервер.

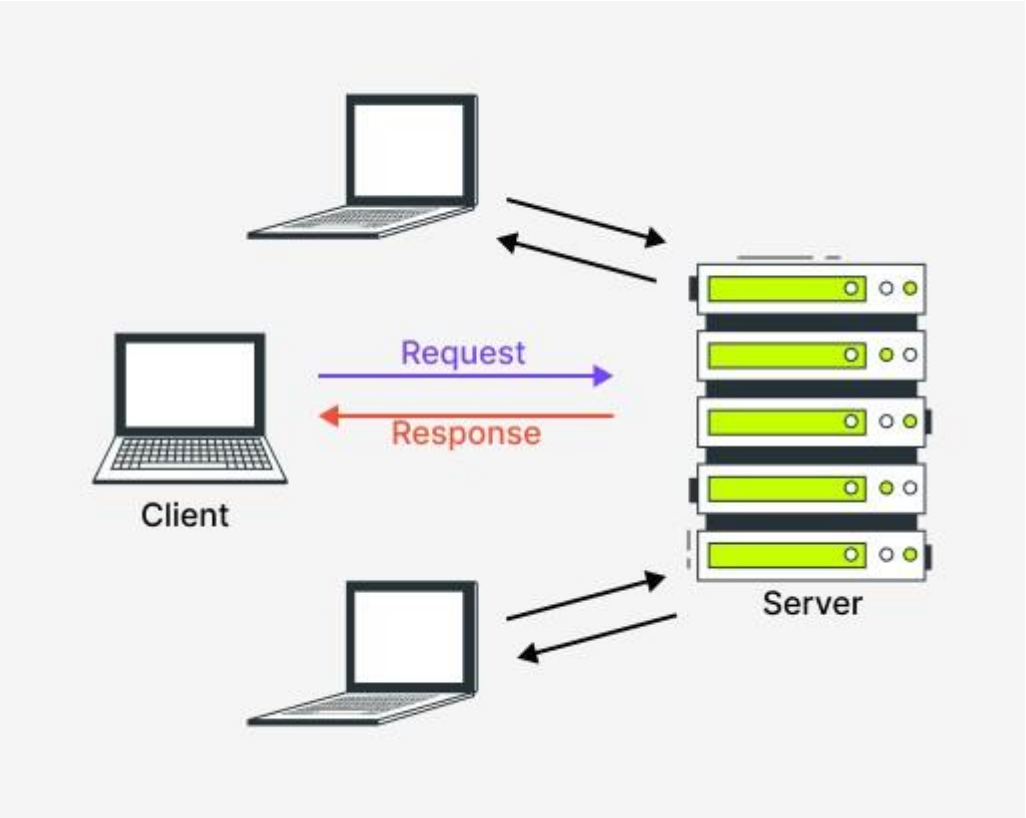

Рисунок 3.1 − Клієнт-серверна архітектура

Клієнти – це робочі станції, які використовують ресурси сервера і надають зручні інтерфейси користувача. Інтерфейси користувача це процедури взаємодії користувача з системою або мережею [24].

Тому для інформаційної технології було обрано клієнт-серверну архітектуру. Такий архітектурний підхід має ряд переваг для розробки інформаційної технології надання рекомендацій для догляду за кімнатними рослинами:

Ця архітектура забезпечує масштабованість системи. Серверна частина може бути масштабована горизонтально, додаючи нові сервери для обробки зростаючого навантаження, що дозволяє обслуговувати велику кількість одночасних користувачів. Клієнтську частину можна розгорнути на різних пристроях, таких як WEB-браузери та мобільні додатки, забезпечуючи гнучкість та доступність системи.

Крім того, клієнт-серверна архітектура дозволяє ефективно розподіляти навантаження. Складні обчислення, пов'язані з аналізом даних про рослини та

генерацією рекомендацій, будуть перенесені на сервер, що розвантажує клієнтські пристрої і дозволяє використовувати інформаційну технології на більшій кількості пристроїв. Клієнти в свою чергу можуть виконувати лише операції, пов'язані з відображенням інтерфейсу та взаємодією з користувачем.

Важливою перевагою є також централізоване управління та оновлення системи. Розміщення логіки на сервері дозволяє здійснювати розгортання або оновлення лише сервера, який в свою чергу забезпечить клієнтам доступ до актуальної версії. Це спрощує процес підтримки та гарантує відповідність сервісів вимогам.

Крім того, клієнт-серверна архітектура підвищує безпеку та конфіденційність системи. Зберігання та обробка конфіденційних даних може бути реалізована на серверній частині, а доступ до критичних функцій та даних може бути захищений більш надійними методами, оскільки доступ до інформації буде в обмеженої кількості людей.

Клієнтська частина може бути розроблена з використанням сучасних фреймворків для WEB-інтерфейсів або мобільних додатків, а серверна − із застосуванням технологій, оптимальних для обробки даних та реалізації алгоритмів рекомендацій.

Клієнт-серверна архітектура для інформаційної технології надання рекомендацій для догляду за кімнатними рослинами дозволить створити гнучку, масштабовану та безпечну систему, здатну ефективно обслуговувати велику кількість користувачів.

3.2 Проектування структури та розробка алгоритму функціонування інформаційної технології надання рекомендацій для догляду за кімнатними рослинами

Оскільки інформаційна технологія має клієнт-серверну архітектуру, потрібно розробити декілька частин проекту таких як сервер, клієнт та базу

даних. Для швидкої роботи технології та можливості подальшої її підтримки доцільно використати модульний підхід.

Модульне програмування − це спосіб створення програмного забезпечення шляхом розбиття його на менші незалежні частини, які називаються модулями. Кожен модуль схожий на міні-програму, яка обробляє певну частину більшої програми. Основні ідеї модульного програмування включають:

1. Висока згуртованість – усе всередині модуля тісно пов’язане та працює для досягнення однієї мети.

2. Слабкий зв’язок – модулі з’єднуються один з одним лише через зовнішні шари, тому їм не потрібно багато знати один про одного, щоб працювати разом.

3. Абстракція – модулі приховують свою внутрішню роботу, показуючи лише те, що необхідно для їх використання.

4. Автономність – модулі мають усе необхідне для самостійного функціонування, зменшуючи залежність від зовнішніх частин.

Цей підхід дозволяє впровадити логіку більш прозоро, простіше писати та перевіряти, зменшити витрати на технічне обслуговування та модифікацію, підвищити стійкість до несправностей.

В програмі будуть представлені: модуль інтерфейсу, модуль сервера, модуль надання рекомендацій по підбору рослин, модуль роботи з постачальником даних. Структура програмного модуля надання рекомендацій для догляду за кімнатними рослинами наведена на рисунку 3.2.

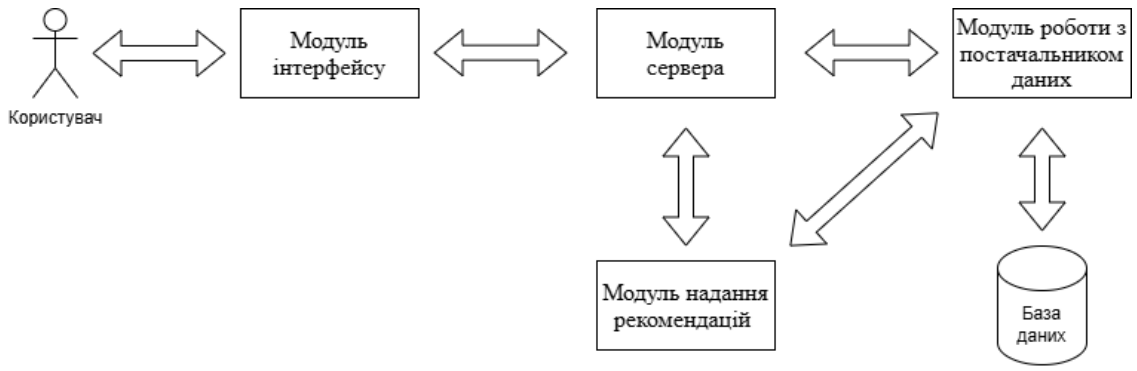

Рисунок 3.2 – Загальна структурна модель програмного модуля

Завдяки графічному інтерфейсу, який пов'язаний з модулем сервера користувач отримує всю необхідну йому інформацію. Він відповідає за коректне відображення на екрані результатів надання рекомендацій, а також за введення та виведення відповідної інформації.

Модуль сервера пов’язаний з модулем роботи з постачальником даних та модулем надання рекомендацій. Цей модуль є головним модулем програми та містить основні зв’язки для функціонування інформаційної технології. Він отримує та передає дані з модуля роботи постачальника даних, а також взаємодіє з модулем надання рекомендацій для отримування рекомендацій на основі уподобань користувача.

Модуль надання рекомендацій по підбору рослин пов’язаний з модулем роботи з постачальником даних. Цей модуль відповідає за логіку підбору рослин для користувача, а також взаємодіє з БД, щоб отримати більш точний результат рекомендацій.

Модуль роботи з постачальником даних пов’язаний з модулем сервера, надання рекомендацій по підбору рослин та самою базою даних. Цей модуль забезпечує швидке і безперебійне отримання актуальних даних та збереження даних.

Для реалізації інформаційної технології надання рекомендацій для догляду за кімнатними рослинами було розроблено схему алгоритму її роботи. Ця схема відображає основні етапи взаємодії користувача з системою та логіку обробки даних. Схема алгоритму зображена на рисунку 3.3.

І випадок

Алгоритм у випадку натиснення на кнопку «Вхід» складається з таких кроків:

Крок 1. Користувач входить на стартову сторінку;

Крок 2. Користувач натискає кнопку «Вхід»;

Крок 3. Введення вхідних даних: Ім’я користувача, пароль;

Крок 4. Перевірка чи є такий користувач у БД;

Крок 9. Вхід у систему, якщо запис існує.

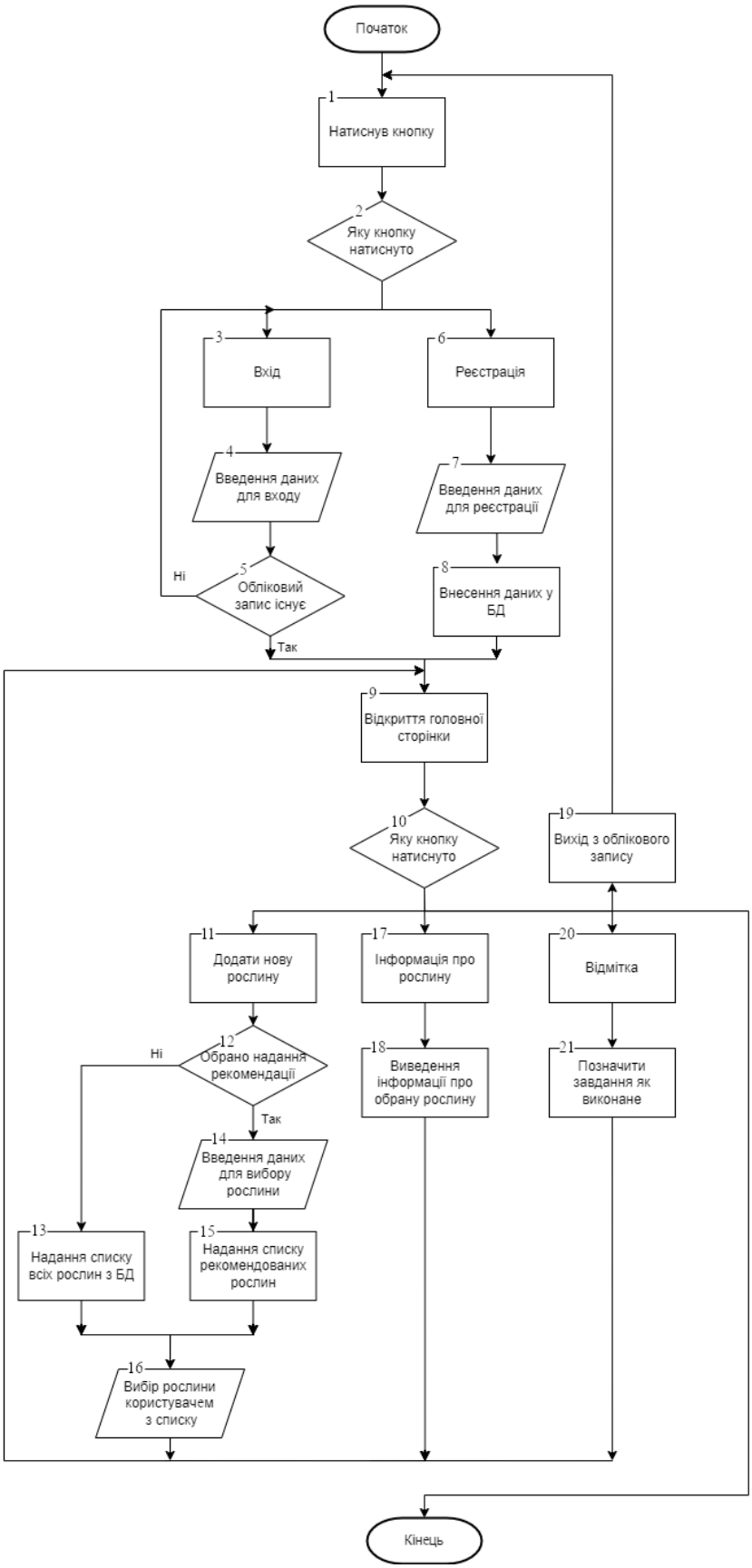

Рисунок 3.3 – Схема алгоритму роботи інформаційної технології надання рекомендацій для догляду за кімнатними рослинами

II випадок

Алгоритм у випадку натиснення на кнопку «Реєстрації» складається з таких кроків:

Крок 1. Користувач входить на стартову сторінку;

Крок 6. Користувач натискає кнопку «Реєстрація»;

Крок 7. Введення вхідних даних: Ім’я, email, пароль;

Крок 8. Створення нового запису у базу даних;

Крок 9. Вхід у систему.

III випадок

Алгоритм у випадку натиснення на кнопку «Відмітка» складається з таких кроків:

Крок 1. Користувач входить на стартову сторінку;

Крок 2. Користувач натискає кнопку «Вхід»;

Крок 3. Введення вхідних даних: Ім’я користувача, пароль;

Крок 4. Перевірка чи є такий користувач у БД;

Крок 9. Вхід у систему, якщо запис існує;

Крок 20: Користувач натискає кнопку «Відмітка»;

Крок 21: Користувач позначає завдання (наприклад, полив рослини) як виконане.

IV випадок

Алгоритм у випадку натиснення на кнопку «Додати нову рослину» складається з таких кроків:

Крок 1. Користувач входить на стартову сторінку;

Крок 2. Користувач натискає кнопку «Вхід»;

Крок 3. Введення вхідних даних: Ім’я користувача, пароль;

Крок 4. Перевірка чи є такий користувач у БД;

Крок 9. Вхід у систему, якщо запис існує;

Крок 11: Користувач натискає кнопку «Додати нову рослину;

Крок 12: Користувач обирає опцію надання рекомендацій для нової рослини;

Крок 14: Введення даних для вибору рослини;

Крок 15: Система надає список рекомендованих рослин;

Крок 16: Користувач обирає рослину зі списку або може переглянути весь список рослин із бази даних.

V випадок

Алгоритм у випадку натиснення на кнопку «Додати нову рослину» складається з таких кроків:

Крок 1. Користувач входить на стартову сторінку;

Крок 2. Користувач натискає кнопку «Вхід»;

Крок 3. Введення вхідних даних: Ім’я користувача, пароль;

Крок 4. Перевірка чи є такий користувач у БД;

Крок 9. Вхід у систему, якщо запис існує;

Крок 11: Користувач натискає кнопку «Додати нову рослину»;

Крок 12: Користувач обирає опцію надання рекомендацій для нової рослини;

Крок 13: Введення даних для вибору рослини;

Крок 16: Користувач обирає рослину зі списку або може переглянути весь список рослин із бази даних.

VI випадок

Алгоритм у випадку натиснення на кнопку «Інформація про рослину» складається з таких кроків:

Крок 1. Користувач входить на стартову сторінку;

Крок 2. Користувач натискає кнопку «Вхід»;

Крок 3. Введення вхідних даних: Ім’я користувача, пароль;

Крок 4. Перевірка чи є такий користувач у БД;

Крок 9. Вхід у систему, якщо запис існує;

Крок 17: Користувач натискає кнопку «Інформація про рослину»;

Крок 18: Система виводить інформацію про обрану рослину.

VII випадок

Алгоритм у випадку натиснення на кнопку «Вихід з облікового запису» складається з таких кроків:

Крок 1. Користувач входить на стартову сторінку;

Крок 2. Користувач натискає кнопку «Вхід»;

Крок 3. Введення вхідних даних: Ім’я користувача, пароль;

Крок 4. Перевірка чи є такий користувач у БД;

Крок 9. Вхід у систему, якщо запис існує;

Крок 19: Користувач натискає кнопку «Вихід з облікового запису»;

Крок 1: Система завершує сесію користувача і повертає на стартову сторінку.

3.3 Розробка схеми алгоритму системи надання рекомендацій по вибору кімнатної рослини

Для реалізації системи надання рекомендацій по вибору кімнатної рослини було розроблено схему алгоритму її роботи. Ця схема відображає основні етапи взаємодії користувача з модулем та логіку обробки даних для надання персоналізованих рекомендацій. Схема алгоритму зображена на рисунку 3.4.

Алгоритм складається з таких кроків:

Крок 1: Завантаження даних з бази даних (БД);

Крок 2: Виведення запитання користувачу;

Крок 3: Користувач відповідає на запитання;

Крок 4: Чи залишилися ще запитання?

Якщо питання залишились, переходимо до кроку 2;

Крок 5: Обробка даних;

Крок 6: Виведення рекомендованих рослин користувачу.

Рисунок 3.4 – Схема алгоритму роботи системи надання рекомендацій по вибору кімнатної рослини

3.4 Обґрунтування вибору мови програмування для реалізації інформаційної технології надання рекомендацій для догляду за кімнатними рослинами

Для створення інформаційної технології надання рекомендацій для догляду за кімнатними рослинами було розглянуто такі мови програмування C# [25], Python [26] та Java [27], доцільно провести детальний порівняльний аналіз цих мов програмування. Такий підхід дозволить оцінити переваги та потенційні недоліки кожної технології в контексті поставленого завдання.

Розглядаючи ключові аспекти кожної з цих мов, важливо враховувати не лише їхні технічні характеристики, але й особливості екосистеми, інструментарію та спільноти розробників. У процесі вибору було звернено увагу на такі аспекти, як продуктивність мови, зручність розробки WEB-ресурсів, наявність відповідних фреймворків та бібліотек, підтримка роботи з базами даних, а також масштабованість та підтримка з боку спільноти розробників. Важливим фактором також є наявність готових рішень та компонентів, які можуть прискорити процес розробки та підвищити якість кінцевого продукту та можливість обробки великих обсягів даних про рослини, реалізація складних алгоритмів рекомендацій, створення інтуїтивно зрозумілого користувацького інтерфейсу та забезпечення високої продуктивності системи при одночасній роботі багатьох користувачів. Розглянемо ці мови програмування детальніше.

C# − це об'єктно-орієнтована мова програмування, розроблена компанією Microsoft як частина платформи .NET. Вона широко використовується для розробки різноманітних додатків, включаючи WEB-ресурси, мобільні додатки, ігри та програмне забезпечення для настільних комп'ютерів. C# має сильну типізацію, що забезпечує надійність та безпеку коду. Ця мова постійно розвивається, і кожна нова версія додає нові функції та можливості, що робить її все більш привабливою для розробників. Одним з ключових переваг C# є його тісна інтеграція з платформою .NET, яка надає багатий набір бібліотек та

інструментів для розробки. Це дозволяє швидко створювати складні додатки, використовуючи готові компоненти та функції.

Для WEB-розробки на C# використовується фреймворк ASP.NET Core. Цей фреймворк дозволяє створювати WEB-додатки та API з використанням архітектури MVC (Model-View-Controller) або більш легшої архітектури Razor Pages. ASP.NET Core також підтримує розробку одно-сторінкових додатків (SPA) з використанням популярних JavaScript-фреймворків, таких як Angular, React або Vue.js.

Для роботи з базами даних в C# часто використовується Entity Framework Core – потужний ORM (Object-Relational Mapping) фреймворк, який спрощує взаємодію з базами даних та дозволяє працювати з ними, використовуючи об'єктно-орієнтований підхід. Також для роботи з базами даних широко використовується ADO.NET. Він пропонує низькорівневий підхід до взаємодії з базами даних, що дозволяє мати більший контроль над SQL-запитами та операціями з даними. Це значно прискорює розробку та робить код більш читабельним та підтримуваним.

Python − це високорівнева мова програмування з акцентом на читабельність коду та простоту синтаксису. Ця мова відома своєю універсальністю та широким спектром застосування, від WEB-розробки до наукових обчислень та машинного навчання. Python має динамічну типізацію, що дозволяє швидко прототипувати ідеї та розробляти програми.

Однією з ключових особливостей Python є величезна кількість бібліотек та фреймворків, доступних для різних задач. Для WEB-розробки широко використовуються фреймворки Django та Flask. Django − це повнофункціональний фреймворк, який дотримується принципу «batteries included» (все включено), надаючи багато готових компонентів для швидкої розробки WEB-додатків. Flask, навпаки, є мікрофреймворком, який надає базові інструменти для WEB-розробки, дозволяючи більше контролювати архітектуру свого додатка.

Python також є лідером у сфері наукових обчислень та аналізу даних. Бібліотеки NumPy та Pandas надають потужні інструменти для роботи з великими масивами даних та їх аналізу. SciPy розширює можливості NumPy, додаючи більше наукових алгоритмів. Для візуалізації даних часто використовуються бібліотеки Matplotlib та Seaborn.

Java − є однією з найбільш широко використовуваних мов програмування у світі. Створена компанією Sun Microsystems, Java відома своєю кросплатформністю завдяки концепції «write once, run anywhere». Це досягається за допомогою Java Virtual Machine, яка дозволяє Java-програмам працювати на будь-якій платформі, де встановлена JVM.

Java широко використовується для розробки корпоративних додатків, особливо у фінансовому секторі та великих організаціях. Її надійність, безпека та масштабованість роблять Java ідеальним вибором для створення складних бізнес-систем. Для WEB-розробки на Java часто використовується фреймворк Spring, який надає комплексний набір інструментів для створення сучасних WEB-додатків та мікросервісів.

Spring Framework є одним з найпопулярніших фреймворків для Java. Він надає широкий спектр можливостей, включаючи інверсію контролю, аспектно-орієнтоване програмування, інтеграцію з базами даних та багато іншого. Spring Boot, частина екосистеми Spring, спрощує процес створення автономних, продакшн-готових Spring-додатків з мінімальною конфігурацією.

Для розробки WEB-сервісів на Java часто використовуються технології, такі як JAX-RS та Spring Web Services. Ці інструменти дозволяють легко створювати та розгортати RESTful API та SOAP WEB-сервіси.

У сфері обробки великих даних Java відіграє значну роль завдяки таким фреймворкам, як Apache Hadoop та Apache Spark. Ці інструменти, написані на Java, широко використовуються для розподіленої обробки та аналізу великих обсягів даних.

Ця порівняльна оцінка дозволить зробити обґрунтований вибір технології, яка найкраще відповідає вимогам проекту з розробки інформаційної технології

надання рекомендацій для догляду за кімнатними рослинами, забезпечуючи оптимальний баланс між продуктивністю, зручністю розробки та потенціалом для майбутнього розвитку. Розглянемо порівняння цих мов програмування, які подано в таблиці 3.1:

Таблиця 3.1 Порівняння мов програмування

Фактор	C#	Java	Python
Продуктив-ність	Висока продуктивність, особливо для WEB-ресурсів	Також забезпечує високу продуктивність	Зазвичай менш продуктивний порівняно з C# та Java
Екосистема WEB-розробки	Має потужний веб фреймворк ASP.NET з широкими можливостями	Має кілька популярних фреймворків (Spring, JavaServer Faces)	Популярні фреймворки включають Django та Flask, але можуть поступатися в потужності ASP.NET
Швидкість розробки	Середня швидкість розробки	Схожа на C# за швидкістю розробки	Відомий своєю високою швидкістю
Типізація	Статична типізація	Статична типізація	Динамічна типізація, яка може призвести до помилок у runtime
Робота з базами даних	Відмінна інтеграція з різними СУБД через Entity Framework та ADO.NET	Потужні інструменти як Hibernate для ORM	Має ORM-рішення, але менш потужні порівняно з C# та Java
Масштабо-ваність	Відмінна масштабованість	Також забезпечує хорошу масштабованість	Може мати проблеми з масштабуванням для великих додатків

Враховуючи ці фактори, вибір C# та ASP.NET для нашого проекту можна обґрунтувати так:

- Висока продуктивність C# важлива для обробки складних алгоритмів рекомендацій.

- Потужність ASP.NET для WEB-розробки забезпечує створення надійного та масштабованого WEB-ресурсу.

- Статична типізація C# зменшує ймовірність помилок у роботі системи рекомендацій.

- Відмінна інтеграція з базами даних через Entity Framework спрощує роботу з даними про рослини.

- Масштабованість C# та ASP.NET важлива для потенційного розширення системи в майбутньому.

Таким чином, хоча Python та Java також мають свої значні переваги, наприклад, простота вивчення, потужні бібліотеки для роботи з даними та штучним інтелектом, а також висока кросплатформеність, C# є більш збалансованим вибором для розробки інформаційної технології надання рекомендацій для догляду за кімнатними рослинами.

ASP.NET MVC, як фреймворк для розробки WEB-ресурсів, пропонує ряд переваг, які роблять його оптимальним вибором для проекту. Перш за все, це тісна інтеграція з серверною частиною на C#, що дозволяє використовувати єдину мову програмування та спільні концепції на всіх рівнях додатку. Це не тільки спрощує процес розробки, але й зменшує ймовірність виникнення помилок при передачі даних між клієнтом і сервером [28].

Порівнюючи ASP.NET MVC з іншими популярними фреймворками для клієнтської розробки, такими як React [29], Angular [30] чи Vue.js [31], варто відзначити, що кожен з них має свої переваги та недоліки. React, наприклад, відомий своєю високою продуктивністю та гнучкістю, але вимагає додаткової конфігурації для інтеграції з серверною частиною на C#. Angular пропонує потужний набір інструментів для розробки складних WEB-ресурсів, але має більш вищу складність навчання . Vue.js, зі свого боку, легкий у вивченні та використанні, але має менш розвинену екосистему порівняно з іншими варіантами. Розглянемо детальніше кожний варіант

ASP.NET MVC:

- Тісна інтеграція з серверною частиною на C#, що забезпечує єдиний стек технологій.

- Підтримка Razor Pages для створення динамічних WEB-сторінок.

- Вбудована підтримка безпеки та аутентифікації.

- Можливість використання TagHelpers для спрощення роботи з HTML.

React:

- Популярний JavaScript фреймворк для створення інтерактивних інтерфейсів.

- Висока продуктивність завдяки віртуальному DOM.

- Великий вибір готових компонентів та бібліотек.

- Необхідність додаткової конфігурації для інтеграції з серверною частиною на C#.

Angular:

- Повноцінний фреймворк з TypeScript, який надає широкі можливості для розробки.

- Вбудована підтримка двостороннього зв'язування даних.

- Потужні інструменти для розробки та тестування.

- Більш складна крива навчання порівняно з ASP.NET MVC.

Vue.js:

- Легкий та гнучкий фреймворк, який легко інтегрується в існуючі проекти.

- Простий у вивченні та використанні.

- Хороша продуктивність та реактивність.

- Менша екосистема порівняно з React або Angular.

Порівнюючи ці технології, можна виділити такі переваги вибору ASP.NET MVC для розробки:

- Єдина екосистема: використання ASP.NET MVC дозволяє залишатися в межах єдиної екосистеми .NET, що спрощує розробку, тестування та розгортання.

- Інтеграція з серверною частиною: тісна інтеграція з C# на сервері дозволяє легко передавати дані між клієнтом та сервером, використовуючи моделі та контролери.

- Безпека: ASP.NET MVC має вбудовані механізми безпеки, які легко налаштовуються та інтегруються з серверною аутентифікацією та авторизацією.

- Продуктивність: завдяки серверному рендерингу, ASP.NET MVC може забезпечити швидку початкову завантаження сторінки, що важливо для SEO та швидкодії.

- Масштабованість: ASP.NET MVC добре масштабується для великих проектів, що важливо для потенційного розширення нашої системи рекомендацій.

- Спільнота та підтримка: велика спільнота розробників .NET та активна підтримка від Microsoft забезпечують довгострокову перспективу розвитку технології.

Вибір C# та ASP.NET дозволяє досягти оптимального поєднання високої продуктивності, критичної для обробки великих масивів даних та алгоритмів рекомендацій, і надійності, що важливо для безперебійної роботи WEB-ресурсу. Завдяки статичній типізації, зменшується ризик помилок, пов'язаних з обробкою даних, що гарантує стабільність системи в довгостроковій перспективі. Крім того, завдяки потужним інструментам для роботи з базами даних, наприклад, Entity Framework, C# спрощує керування даними про рослини, забезпечуючи зручну і ефективну роботу з ними. Ця екосистема також надає масштабованість, яка дозволить легко розширювати функціонал системи в майбутньому, додаючи нові можливості для рекомендацій, інтеграцій або підтримки більшої кількості користувачів. Загалом, C# та ASP.NET забезпечують баланс між швидкістю розробки, високою продуктивністю та гнучкістю, роблячи їх ідеальним вибором для розвитку розробки.

3.5 Обґрунтування вибору середовища для реалізації програмного продукту

При розробці інформаційної технології надання рекомендацій для догляду за кімнатними рослинами важливо обрати оптимальне середовище реалізації програми. Враховуючи вибір мови програмування C#, розглянемо три основні підходи: WEB-ресурс, мобільний додаток та десктопний додаток. Кожен з цих методів має свої переваги та особливості, які важливо проаналізувати в контексті завдання.

WEB-ресурс: Реалізація у вигляді WEB-додатку є найбільш універсальним рішенням. Використовуючи фреймворк ASP.NET Core, можна створити потужний та гнучкий WEB-ресурс, доступний з будь-якого пристрою з WEB-браузером.

Переваги:

1. Широка доступність: користувачі можуть отримати доступ до системи з будь-якого пристрою з інтернет-з'єднанням.

2. Легкість оновлення: оновлення відбуваються централізовано на сервері.

3. Кросплатформеність: працює на різних операційних системах.

4. Інтеграція з хмарними сервісами: легко розгортати та масштабувати в хмарних платформах, як Azure.

Особливості реалізації:

- Використання ASP.NET Core MVC або Blazor для створення інтерактивного інтерфейсу.

- Реалізація RESTful API для обміну даними між клієнтом та сервером.

- Використання Entity Framework Core для роботи з базою даних.

Мобільний додаток: Створення мобільного додатку може бути реалізовано за допомогою Xamarin.Forms, що дозволяє розробляти кросплатформені мобільні додатки на C#.

Переваги:

1. Зручність використання на мобільних пристроях.

2. Можливість використання нативних функцій пристрою (наприклад, камери для сканування рослин).

3. Робота в офлайн-режимі з подальшою синхронізацією.

Особливості реалізації:

- Використання Xamarin.Forms для створення кросплатформеного інтерфейсу.

- Реалізація локального сховища даних для роботи офлайн.

- Інтеграція з WEB-сервісами для синхронізації даних.

Десктопний додаток: Розробка десктопного додатку може бути здійснена за допомогою Windows Forms або WPF (Windows Presentation Foundation).

Переваги:

1. Висока продуктивність та швидкість роботи.
2. Можливість глибокої інтеграції з операційною системою.
3. Зручність використання для користувачів, які надають перевагу традиційним додаткам.

Особливості реалізації:

- Використання WPF для створення сучасного та гнучкого інтерфейсу.

- Реалізація локальної бази даних з можливістю синхронізації з хмарним сервером.

- Використання багатопотоковості для обробки складних обчислень.

Враховуючи специфіку завдання − надання рекомендацій для догляду за кімнатними рослинами − оптимальним вибором є реалізація у вигляді WEB-ресурсу. Цей підхід забезпечує найбільшу гнучкість та доступність для користувачів, дозволяючи їм отримувати рекомендації з будь-якого пристрою. Крім того, WEB-ресурс легше оновлювати та масштабувати, що важливо для системи, яка може регулярно отримувати нові дані про рослини та методи догляду.

Реалізація WEB-додатку на базі ASP.NET Core дозволить використовувати всі переваги екосистеми .NET, включаючи потужні інструменти для роботи з даними, реалізації складних алгоритмів рекомендацій та створення зручного

користувацького інтерфейсу. Крім того, це рішення забезпечить гарну основу для потенційного майбутнього розширення функціоналу, наприклад додавання API для інтеграції з мобільними додатками або іншими системами.

Отже, WEB-ресурс на базі ASP.NET Core є найбільш оптимальним методом реалізації інформаційної технології, що забезпечить широку доступність, гнучкість у розробці та гарні перспективи для подальшого розвитку інформаційної технології.

3.6 Реалізація програмного модуля інформаційної технології надання рекомендацій для догляду за кімнатними рослинами

Для розробки програмного модуля використовується .NET та ASP.NET Core. Сам програмний модуль складається з 3 модулів кожен з яких відповідає за свої функції інформаційної технології.

Розглянемо перший модуль який відповідає за взаємодію з базою даних та містить основні моделі. Він містить три класи, які представляють собою моделі даних.

Першим класом є модель користувача, яка зберігає у собі пошту, ім’я користувача, хеш пароль та код, який використовується для хешування пароля.

Друга модель відповідає за збереження характеристик рослини, в ній зберігається назва рослини, шлях до зображення, опис та період поливу, удобрення, пересадку.

Третя модель відповідає за рослину користувача, в ній знаходиться інформація про конкретну рослину, яка є у користувача. Вона має такі поля: остання та наступна дата поливу, удобрення, пересадки. Також в ній є три методи, які надсилають інформацію про зміну наступної і останньої дати поливу, удобрення, пересадки.

Ще цей модуль містить класи для взаємодії з базою даних. В ньому використовується ADO.NET, який дозволяє надсилати команди та отримувати інформацію до SQL сервера [32]. Функції для отримання інформації з БД

використовують SQL команди та зберігають інформацію у моделі, після чого повертають їх, функція для отримання інформації про рослини виглядає так:

Діаграма діяльності серверу під час запиту інформації про рослину зображена на рисунку 3.5.

Рисунок 3.5 – Діаграма діяльності серверу під час запиту інформації про рослину

Зображений вище алгоритм (рис. 3.5) можна записати у вигляді коду:

```
public Plant GetPlant(int id)
{
   string sqlCom = @"select top 1 * from Plants where PlantId = @Id";
   Plant plant = ExecuteQuery(sqlCom, id).FirstOrDefault();
   return plant;
}
```

За допомогою цієї функції формується команда та передається да наступної функції, яка обробляє команду та виконує її.

```
private IEnumerable<Plant> ExecuteQuery(string sqlCom, int? id = null)
{
   List<Plant> plants = new List<Plant>();
   using (SqlConnection sqlConnection = new SqlConnection(_connectionString))
   {
      SqlCommand cn = new SqlCommand(sqlCom, sqlConnection);
      if (id != null)
         cn.Parameters.AddWithValue("@Id", id);
      sqlConnection.Open();
      using (SqlDataReader reader = cn.ExecuteReader())
      {
         while (reader.Read())
         {
            int PlantId = reader.GetInt32(0);
            string Name = reader.GetString(1);
            string Descripption = reader.GetString(2);
            int WateringPeriod = reader.GetInt32(3);
            int TransplantPeriod = reader.GetInt32(4);
            int FertalizationPeriod = reader.GetInt32(5);
            string UniqueImageName = reader.GetString(6);
            Plant plant = new Plant()
            {
               Id = PlantId,
               Name = Name,
               Description = Descripption,
               WateringPeriod = WateringPeriod,
               TransplantPeriod = TransplantPeriod,
               FertalizationPeriod = FertalizationPeriod,
               UniqueImageName = UniqueImageName
```

```
            };
            plants.Add(plant);
        }
    }
  }
  return plants;
}
```

Ця функція створює з’єднання з SQL сервером, інтегрує параметри у команду та ініціалізує її виконання, після чого за допомогою SqlDataReader зчитує інформацію та перетворює її на модель рослини програмного модуля та додає у список, який повертає весь знайдений результат. Такі функції використовують для отримання всіх рослин та рослин користувача. Також в цьому класі відбувається вибір рослини користувачем, видалення та зміна статусу поливу, удобрення та пересадки. Розглянемо функцію, яка сприяє вибору рослини користувачем, передає ІД користувача та рослини на SQL сервер, який виконує збережену процедура. Зміст функції такий:

```
ALTER PROCEDURE [dbo].[AddPlantToUserPlants]
  @UserId INT,
  @PlantId INT
AS
BEGIN
  INSERT INTO PlantUserRelations (UserId, PlantId)
  VALUES (@UserId, @PlantId);
  DECLARE @Id INT;
  SELECT TOP 1 @Id = PlantUserRelationsId
  FROM PlantUserRelations
  ORDER BY PlantUserRelationsId DESC;
    INSERT INTO UserPlantsState
```

```
        VALUES (@Id, CAST(GETDATE() AS DATE), CAST(GETDATE() AS
DATE), CAST(GETDATE() AS DATE),
            CAST(GETDATE() AS DATE), CAST(GETDATE() AS DATE),
CAST(GETDATE() AS DATE));
    END;
```

Останній клас у цьому модулі відповідає за отримання та запис інформації про користувача. Розглянемо метод, який перевіряє чи існує вже ім’я користувача, чи електрона адреса у базі даних та повертає пустий рядок, якщо цього ім’я немає або заповнює її текстом "Username already exists", або "Email already exists". Діаграма діяльності серверу під час перевірки чи існує користувач у системі зображена на рисунку 3.6.

Рисунок 3.6 – Діаграма діяльності серверу під час перевірки чи існує користувач у системі

Зображений вище алгоритм (рис. 3.6) можна записати у вигляді коду:

```
public string IsEmailOrUserNameExist(string email, string username)
 {
    string result = string.Empty;
    string sqlEmailQuery = "Select Count(Email) from Users where Email =
@Email";
    string sqlUsernameQuery = "Select Count(Username) from Users where
Username = @Username";
    using (SqlConnection sqlConnection = new SqlConnection(_connectionString))
    {
       SqlCommand cmd = sqlConnection.CreateCommand();
       cmd.CommandText = sqlEmailQuery;
       cmd.Parameters.AddWithValue("@Email", email);
       sqlConnection.Open();
       if (cmd.ExecuteNonQuery() != -1)
          result = "Email already exists";
       cmd.CommandText = sqlUsernameQuery;
       cmd.Parameters.Clear();
       cmd.Parameters.AddWithValue("@Username", username);
       if (cmd.ExecuteNonQuery() != -1)
          result = "Username already exists";
    }
    return result;
}
```

Другий модуль відповідає за систему надання рекомендацій та містить три моделі, один клас для взаємодії з базою даних та клас, який відповідає за логіку системи. Розглянемо модулі:

1. Першою є модель варіанту відповіді, яка містить ІД, текст відповіді та її значення.

2. Наступною є модуль питання. Він включає в себе ІД, текст, номер питання та варіанти відповідей.

3. Останньою є модель, яка зберігає результат відповідей на питання.

Клас, який відповідає за зв’язок з БД має методи для отримання питань. Він використовує дані з двох таблиць, створює моделі питань та повертає список питань:

```
public async Task<IEnumerable<QuestionModel>> SeedQuestionsAsync()
{
   string sqlCom = @"select * from Questions";
   List<QuestionModel> questions = new List<QuestionModel>();
   using (SqlConnection sqlConnection = new SqlConnection(_connectionString))
   {
      SqlCommand cn = new SqlCommand(sqlCom, sqlConnection);
      sqlConnection.Open();
      using (SqlDataReader reader = await cn.ExecuteReaderAsync())
      {
         int QuestionNumber = 0;
         while (await reader.ReadAsync())
         {
            int QuestionId = reader.GetInt32(0);
            string QuestionText = reader.GetString(1);
            QuestionModel questionModel = new QuestionModel()
            {
               Id = QuestionId,
               QuestionText = QuestionText,
               QuestionNumber = QuestionNumber
            };
            QuestionNumber++;
            questions.Add(questionModel);
         }
      }
```

```
            sqlCom = @"select * from Answers";
            cn.CommandText = sqlCom;

            using (SqlDataReader reader = await cn.ExecuteReaderAsync())
            {
               while (await reader.ReadAsync())
               {
                  int AnswerId = reader.GetInt32(0);
                  int QuestionId = reader.GetInt32(1);
                  string AnswerText = reader.GetString(2);
                  int AnswerValue = reader.GetInt32(3);
                  SurveyOption answerModel = new SurveyOption()
                  {
                     Id = AnswerId,
                     OptionText = AnswerText,
                     OptionValue = AnswerValue
                  };
                  questions.Where(x => x.Id == QuestionId).
                     First().Options.Add(answerModel);
               }
            }
         }
         return questions;
      }
```

Також він має функцію, яка порівнює відповіді користувача з характеристиками рослин у базі даних і повертає ті, які найкраще відповідають його вимогам, на основі пройденого тесту:

```
      public async Task<List<Plant>> GetSimiliarPlantsAsync(PlantScoreModel
plantScoreModel)
      {
```

```
            List<Plant> plants = new List<Plant>();
            string sqlCom = "CompareUserPlant";
            using (SqlConnection sqlConn = new SqlConnection(_connectionString))
            {
                SqlCommand cmd = new SqlCommand(sqlCom, sqlConn);
                cmd.CommandType = System.Data.CommandType.StoredProcedure;
                cmd.Parameters.AddWithValue("@WateringFrequency",
plantScoreModel.WateringFrequency);
                cmd.Parameters.AddWithValue("@ComplexityOfCare",
plantScoreModel.ComplexityOfCare);
                cmd.Parameters.AddWithValue("@Size", plantScoreModel.Size);
                cmd.Parameters.AddWithValue("@Temperature",
plantScoreModel.Temperature);
                cmd.Parameters.AddWithValue("@Lighting", plantScoreModel.Lighting);
                cmd.Parameters.AddWithValue("@ColorSaturation",
plantScoreModel.ColorSaturation);
                sqlConn.Open();
                using (SqlDataReader reader = await cmd.ExecuteReaderAsync())
                {
                    while (reader.Read())
                    {
                        plants.Add(ReadPlant(reader));
                    }
                }
            }
            return plants;
        }
```

У випадку якщо таких рослин немає, то викликається інша функція, яка використовує 10 найбільш прийнятних рослин:

```
        public async Task<List<Plant>> GetTop10PlantsAsync()
```

```
{
    List<Plant> plants = new List<Plant>();
    string sqlCom = "GetTop10Plants";
    using (SqlConnection sqlConn = new SqlConnection(_connectionString))
    {
        SqlCommand cmd = new SqlCommand(sqlCom, sqlConn);
        cmd.CommandType = System.Data.CommandType.StoredProcedure;
        sqlConn.Open();
        using (SqlDataReader reader = await cmd.ExecuteReaderAsync())
        {
            while (reader.Read())
            {
                plants.Add(ReadPlant(reader));
            }
        }
    }
    return plants;
}
```

Ці дві функції використовують збережені процедури, розглянемо їх детальніше:

```
ALTER PROCEDURE [dbo].[CompareUserPlant]
    @WateringFrequency INT,
    @ComplexityOfCare INT,
    @Size INT,
    @Temperature INT,
    @Lighting INT,
    @ColorSaturation INT
AS
BEGIN
    SELECT top 10 pl.*
```

```
            FROM PlantScoreModels p
              join Plants pl on pl.PlantId = p.PlantId
            WHERE (@WateringFrequency - 1 = p.WateringFrequency
                OR @WateringFrequency + 1 = p.WateringFrequency
                OR @WateringFrequency = p.WateringFrequency)
             AND (@ComplexityOfCare - 1 = p.ComplexityOfCare
                OR @ComplexityOfCare + 1 = p.ComplexityOfCare
                OR @ComplexityOfCare = p.ComplexityOfCare)
             AND (@Size - 1 = p.Size
                OR @Size + 1 = p.Size
                OR @Size = p.Size)
             AND (@Temperature - 1 = p.Temperature
                OR @Temperature + 1 = p.Temperature
                OR @Temperature = p.Temperature)
             AND (@Lighting - 1 = p.Lighting
                OR @Lighting + 1 = p.Lighting
                OR @Lighting = p.Lighting)
             AND (@ColorSaturation - 1 = p.ColorSaturation
                OR @ColorSaturation + 1 = p.ColorSaturation
                OR @ColorSaturation = p.ColorSaturation)
              ORDER BY
            CASE WHEN @WateringFrequency = p.WateringFrequency THEN 1 ELSE 0
END +
            CASE WHEN @ComplexityOfCare = p.ComplexityOfCare THEN 1 ELSE 0
END +
            CASE WHEN @Size = p.Size THEN 1 ELSE 0 END +
            CASE WHEN @Temperature = p.Temperature THEN 1 ELSE 0 END +
            CASE WHEN @Lighting = p.Lighting THEN 1 ELSE 0 END +
            CASE WHEN @ColorSaturation = p.ColorSaturation THEN 1 ELSE 0 END
DESC;
        END;
```

Ця процедура порівнює характеристики та сортує результат за найбільш важливими критеріями. Наступна процедура використовує 10 найбільш популярних рослин та повертає їх відсортованими за популярністю.

Третій клас має список запитань, модель попереднього класу та модель класу взаємодії з БД для рослин. Основна функція отримує модель відповідей користувача та в залежності від неї викликає інші методи:

```
public async Task<List<Plant>> GetSimiliarPlantsAsync(PlantScoreModel
userPlant)
        {
            List<Plant> similiarPlants = await
_surveySeeder.GetSimiliarPlantsAsync(userPlant);

            if (!similiarPlants.Any())
            {
                similiarPlants = await _surveySeeder.GetTop10PlantsAsync();
            }

            return similiarPlants;
        }
```

Останній модуль відповідає за WEB та клієнтську частину. В ньому використовується ASP.NET CORE MVC. Він складається з трьох основних складових, які розподілені в 3 папках:

1. Controllers – містить контролери, які відповідають за логіку взаємодії з WEB-сторінками.

2. Views – містить папки з WEB-сторінками. Кожна папка підключена до контролера та має мати таку саму назву як і контролер.

3. Models – містить моделі, які будуть використовуватись у програмному модулі.

Розглянемо моделі. В цьому модулі є дві моделі RegisterViewModel та OptionPlantModel. RegisterViewModel містить інформацію для сторінки реєстрації та має поля для ім’я користувача, електронної адреси та пароля. OptionPlantModel відповідає за збереження моделей питань та моделі відповіді.

Розглянемо контролери. Перший контролер, який викликається при вході на сайт − це HomeController. В ньому присутній лише один метод, який викликає WEB-сторінку Index, яка дає користувачу можливість авторизуватись, пройти реєстрацію або якщо користувач вже входив раніше, то його перенаправити на його ж сторінку. HomeController зображений нижче:

```
[AllowAnonymous]
 public class HomeController : Controller
 {
    [AllowAnonymous]
    public ActionResult Index()
    {
       if (User.Identity.IsAuthenticated)
       {
          return RedirectToAction("Index", "Main");
       }
       return View();
    }
  }
```

Розглянемо контролер AccountController. Він відповідає за авторизацію, реєстрацію та вихід з акаунту. Також для входу та реєстрації використовується метод хешування, який відбувається в окремому класі PasswordHelper. Він має три функції GenerateSalt, HashPassword та VerifyPassword.

Наступним є контролер для відображення головної сторінки, де зображуються рослини користувача та можна переходити на інші функції, як рекомендація нових рослин. Він має три основні функції такі як

GetUpdatedPlants, RemovePlant та GetPlantInfo, які оновлюють інформацію про рослини користувача. Також є функція, яка отримує всі дані про користувача та запускається при відкриті головної сторінки:

```
private List<UsersPlant> GetDashboardItems()
{
   List<UsersPlant> plantsModel;
   int userId = int.Parse(User.Identity.Name);
   plantsModel = _sqlPlantReader.GetListOfUserPlantsById(userId);
   if (plantsModel == default) plantsModel = new List<UsersPlant>();
   return plantsModel;
}
```

3.7 Тестування інформаційної технології надання рекомендацій для догляду за кімнатними рослинами

Тестування − це дуже важливий крок, коли створюється новий комп'ютерний продукт. Потрібно впевнитись, що програма робить все, що від неї очікують, і працює без збоїв та помилок. Головне завдання такої перевірки − знайти всі можливі проблеми та недоліки в роботі програми до того, як нею почнуть користуватися люди. Це допомагає зробити програму кращою та уникнути неприємностей у майбутньому.

Проведено понад 100 тестових запусків додатку, випробувано його функціональні можливості та оцінено його роботу. Для того, щоб почати роботу з інформаційною технологією, потрібно мати доступ до інтернет браузера. Головною умовою є стабільне підключення до мобільної мережі інтернету або Wifi. На рисунку 3.7 зображено вікно авторизації користувача.

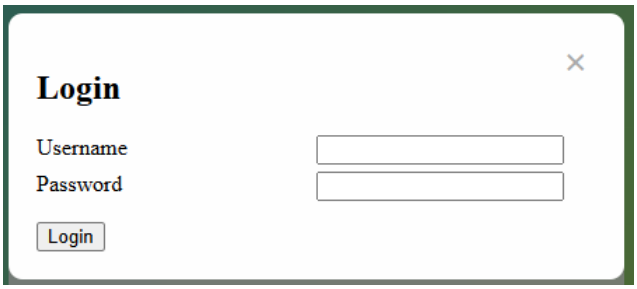

Рисунок 3.7 – Загальний вигляд інтерфейсного вікна авторизації користувача

Після успішної авторизації користувач переадресовується на вікно головного меню програми. Це вікно містить всю необхідну для користувача інформацію. У головному вікні можна побачити рослини та прогрес по догляду рослини. На рисунку 3.8 зображена загальний вигляд інтерфейсного вікна роботи програми.

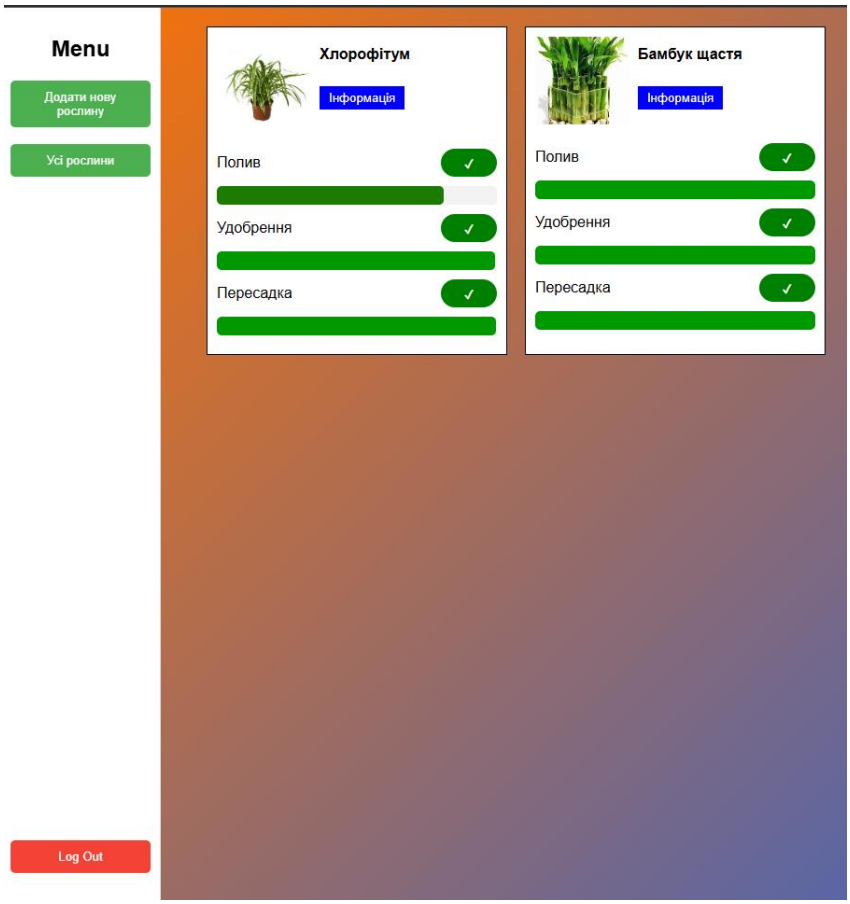

Рисунок 3.8 – Загальний вигляд інтерфейсного вікна роботи програми

Кожну рослину представлено у вигляді блока на якому і зображена основна інформація. На рисунку 3.9 зображено вікно блоку «Рослини».

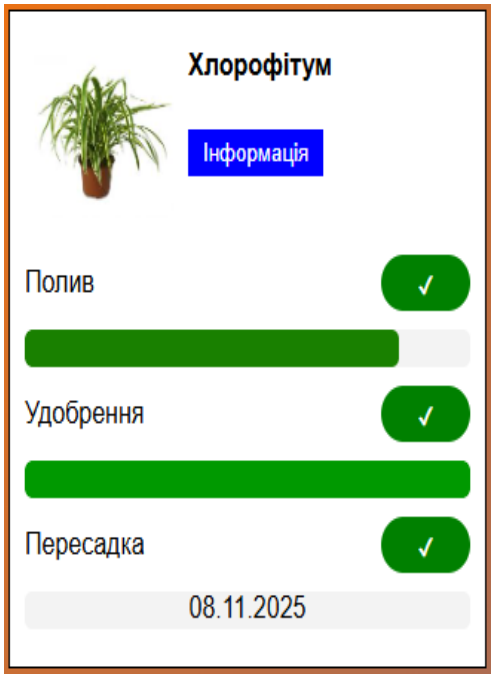

Рисунок 3.9 – Загальний вигляд блоку «Рослини»

Також у лівій частині зображено можливості користувача такі як додати нову рослину, вийти з акаунта або переглянути всі рослини (рис. 3.10).

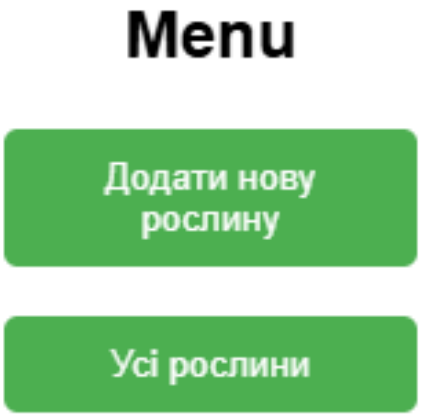

Рисунок 3.10 – Загальний вигляд блоку «Меню»

При натисненні на кнопку «Додати нову рослину» відкривається вікно з питаннями на які користувач має дати відповіді, після чого він отримає

результати у вигляді списку рослин, які система надання рекомендацій вважає найбільш прийнятними для користувача. Вікно питання зображено на рисунку 3.11, вікно результату зображено на рисунку 3.12.

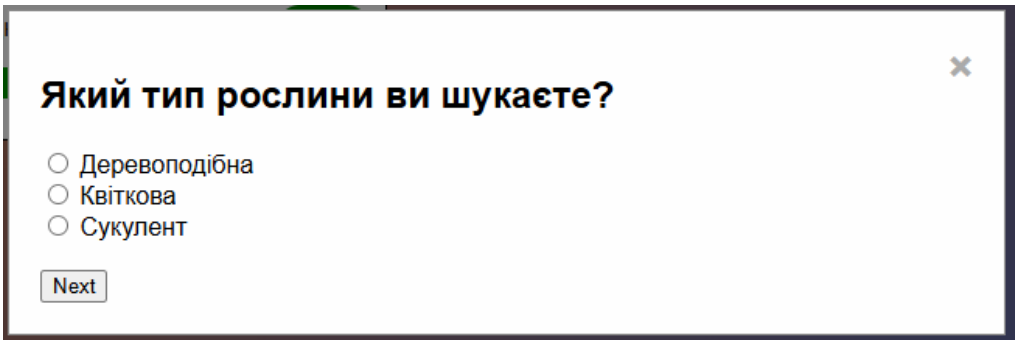

Рисунок 3.11 – Загальний вигляд вікна «Питання»

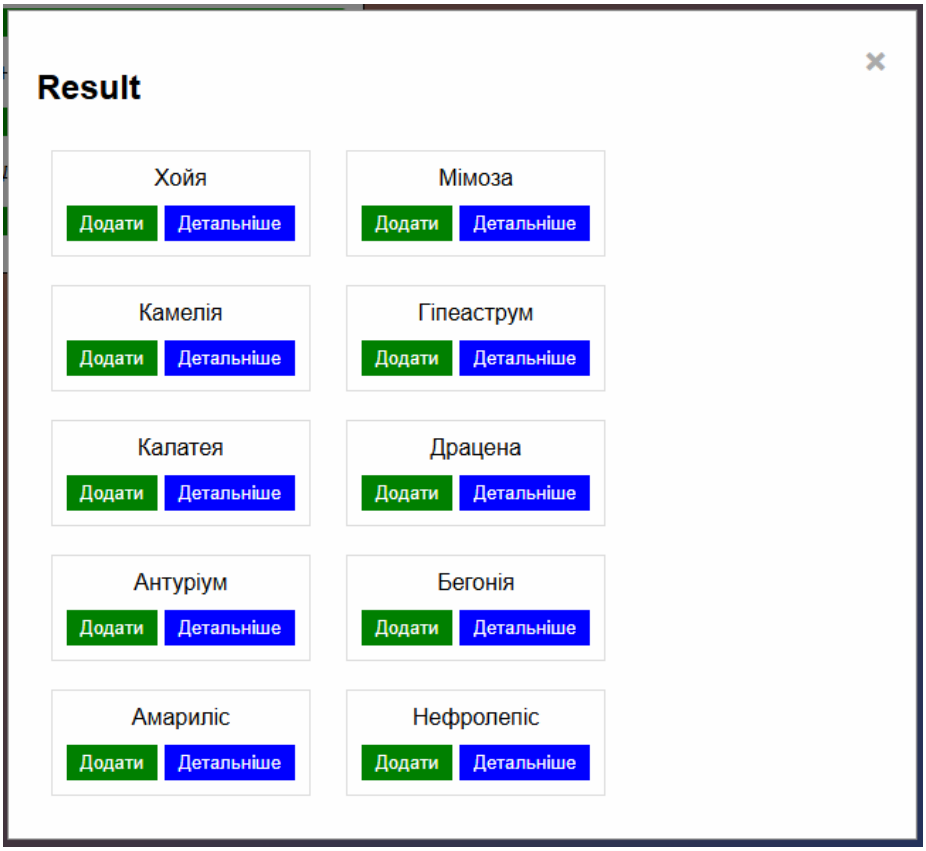

Рисунок 3.12 – Загальний вигляд вікна «Результат»

Розроблений програмний модуль надання рекомендацій для догляду за кімнатними рослинами успішно пройшов тестування та відповідає всім заданим вимогам.

В таблиці 3.2 подано результати тестування розробленого програмного продукту та аналогів.

Таблиця 3.2 – Результати тестування розробленого програмного продукту та аналогів

	Розроблений програмний продукт	Blossom	PlantNet	Gardenia
Українська локалізація	+	-	+	-
Відсутність реклами	+	+	-	+
Можливість контролю виконання дії з рослиною	+	-	-	+
Рослин користувача понад 1000 шт.	+	-	+	-
Рекомендація рослини до уподобань користувача	+	-	-	-

Отже, проаналізувавши та протестувавши розроблений програмний продукт, можна дійти висновку, що поставлених цілей було досягнуто.

З таблиці 3.2 можна побачити, що розроблений програмний модуль надання рекомендацій для догляду за кімнатними рослинами має розширений функціонал у порівнянні з системами аналогами щонайменше на 3 можливості, а саме: додано українську локалізацію, можливість контролю виконання дії з рослиною та рекомендація рослини до уподобань користувача. Розширений функціонал забезпечить вище задоволення користувачів від використання цього програмного продукту.

3.8 Висновок до розділу 3

У третьому розділі обґрунтовано вибір архітектури інформаційної технології надання рекомендацій для догляду за кімнатними рослинами. Для реалізації інформаційної технології було обрано клієнт-серверний тип архітектури, оскільки він підтримує багатокористувацьку роботу, гарантує цілісність і безпеку даних, має гарну оптимізацію та можливість оновлення

системи без втручання у клієнтський застосунок. Розроблено схему алгоритму роботи програмного модуля та описано детально його кроки.

Проаналізовано мови програмування для створення як клієнтських частин: Angular та ASP.NET MVC, так і серверних частин: Java, C# та Python. Визначено їх особливості, переваги та недоліки, а також доцільність у використанні для поставленого завдання. В результаті аналізу для реалізації клієнтської частини програмного модуля для інформаційної технології було обрано мову ASP.NET MVC, а для серверної частини − C#. C# та ASP.NET дозволяє досягти оптимального поєднання високої продуктивності, критичної для обробки великих масивів даних та алгоритмів рекомендацій і надійності, що важливо для безперебійної роботи WEB-ресурсу.

Описано процес реалізації основних частин програмного модуля та системи надання рекомендацій. Також продемонстровано взаємодію серверної частини з базою даних.

Було проведено тестування програмного продукту надання рекомендацій для догляду за кімнатними рослинами та здійснено його порівняння з аналогами. Як результат, зроблено висновок, що основні функції, які потрібно було реалізувати – реалізовано. Розроблений програмний продукт має розширений функціонал щонайменше на 3 можливості, а саме: додано українську локалізацію, можливість контролю виконання дії з рослиною та рекомендація рослини до уподобань користувача.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Проведення комерційного та технологічного аудиту інформаційної технології надання рекомендацій для догляду за кімнатними рослинами

Метою проведення комерційного і технологічного аудиту є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності, тобто під час виконання магістерської кваліфікаційної роботи [33].

Для проведення комерційного та технологічного аудиту залучаємо 3-х незалежних експертів, якими є провідні викладачі випускової або спорідненої кафедри.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу здійснюємо із застосуванням п’ятибальної системи оцінювання за 12-ма критеріями, а результати зводимо до таблиці 4.1.

Таблиця 4.1 – Результати оцінювання науково-технічного рівня і комерційного потенціалу інформаційної технології надання рекомендацій для догляду за кімнатними рослинами

Критерії	Експерти		
	Експерт 1	Експерт 2	Експерт 3
	Бали, виставлені експертами		
Технічна здійсненність концепції	2	3	3
Ринкові переваги (наявність аналогів)	2	3	2
Ринкові переваги (ціна продукту)	2	2	3
Ринкові переваги (технічні властивості)	3	3	2
Ринкові переваги (експлуатаційні витрати)	3	3	3
Ринкові перспективи (розмір ринку)	3	2	2
Ринкові перспективи (конкуренція)	3	2	2
Практична здійсненність (наявність фахівців)	3	2	3

Продовження таблиці 4.1

Критерії	Експерти		
	Експерт 1	Експерт 2	Експерт 3
	Бали, виставлені експертами		
Практична здійсненність (наявність фінансів)	3	3	2
Практична здійсненність (необхідність нових матеріалів)	3	3	2
Практична здійсненність (термін реалізації)	3	2	3
Практична здійсненність (розробка документів)	3	2	2
Сума балів	33	30	29
Середньоарифметична сума балів, СБ	30,7		

За результатами розрахунків, наведених в таблиці 4.1 робимо висновок про те, що науково-технічний рівень та комерційний потенціал інформаційної технології надання рекомендацій для догляду за кімнатними рослинами – вище середнього.

4.2 Розрахунок витрат на здійснення науково-дослідної роботи розробки інформаційної технології надання рекомендацій для догляду за кімнатними рослинами

Витрати на оплату праці. Належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці, також будь-які види грошових і матеріальних доплат, які належать до елемента «Витрати на оплату праці».

Основна заробітна плата дослідників. Витрати на основну заробітну плату дослідників ($З_о$) розраховують відповідно до посадових окладів працівників, за формулою:

$$З_о = \sum_{i=1}^{к} \frac{М_{ni} \cdot t_i}{Т_р}, \quad (4.1)$$

де к – кількість посад дослідників, залучених до процесу дослідження;

$М_{ni}$ – місячний посадовий оклад конкретного розробника (інженера, дослідника, науковця тощо), грн.;

$Т_р$ – число робочих днів в місяці; приблизно $Т_р$ = (21…23) дні;

t_i – число робочих днів роботи розробника (дослідника).

Зроблені розрахунки зводимо до таблиці 4.2.

Таблиця 4.2 – Витрати на заробітну плату дослідників

Посада	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник	41000	1864	37	68968
Розробник	30000	1364	35	47740
Консультанти	18000	819	21	17199
Всього:	133907			

Основна заробітна плата робітників. Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт розраховують за формулою:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i, \quad (4.2)$$

де $С_i$ – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника на виконання певної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду С і можна визначити за формулою:

$$С_і = \frac{М_м \cdot К_і \cdot К_с}{Т_р \cdot t_{зм}}, \quad (4.3)$$

де $М_м$ – розмір прожиткового мінімуму працездатної особи або мінімальної місячної заробітної плати (залежно від діючого законодавства), у 2024 році $М_м$=8000 грн;

$К_і$ – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду;

$К_с$ – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об'єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати;

$Т_р$ – середня кількість робочих днів в місяці, приблизно $Т_р$ = 21…23 дні;

$t_{зм}$ – тривалість зміни, год.

Зроблені розрахунки зводимо до таблиці 4.3.

Таблиця 4.3 – Витрати на заробітну плату робітників

Найменування робіт	Трудомісткість, н-год.	Розряд роботи	Погодинна тарифна ставка	Тариф. коеф.	Величина, грн.
Аналіз предметної області	18	5	61,8	1,36	1112,4
Моделювання інформаційної технології	125	5	61,8	1,36	7725
Створення основної структури проєкту	40	6	65,9	1,45	2636
Створення та наповнення бази даних	100	6	65,9	1,45	6590
Тестування інформаційної технології	75	5	61,8	1,36	4635
Всього					22698,4

Додаткова заробітна плата. Додаткова заробітна плата $З_д$ всіх розробників та робітників, які брали участь у виконанні даного етапу роботи, розраховується як (10…12)% від суми основної заробітної плати всіх розробників та робітників, тобто:

$$З_д = 0{,}1 \cdot (З_о + З_р) = 0{,}12 \cdot (133907 + 22698{,}4) = 18793 \text{ (грн)}. \quad (4.4)$$

Відрахування на соціальні заходи. Нарахування на заробітну плату $Н_{зп}$ розробників та робітників, які брали участь у виконанні даного етапу роботи, розраховуються за формулою:

$$Н_{зп} = \beta \cdot (З_о + З_р + З_д) = \quad (4.5)$$
$$= 0{,}22 \cdot (133907 + 22698{,}4 + 18793) = 35080 \text{ (грн)}.$$

де $З_о$ – основна заробітна плата розробників, грн.;

$З_р$ – основна заробітна плата робітників, грн.;

$З_д$ – додаткова заробітна плата всіх розробників та робітників, грн.;

β – ставка єдиного внеску на загальнообов'язкове державне соціальне страхування, % (приймаємо для 1-го класу професійності ризику 22%).

Програмне забезпечення. До балансової вартості програмного забезпечення входять витрати на його інсталяцію, тому ці витрати беруться додатково в розмірі 10…12% від вартості програмного забезпечення. Балансову вартість програмного забезпечення розраховують за формулою:

$$В_{прг} = \sum_{1}^{к} Ц_{іпрг} \cdot С_{прг.і} \cdot К_і, \quad (4.6)$$

де $Ц_{іпрг}$ – ціна придбання програмного забезпечення і-го виду, грн.;

$С_{прг.і}$ – кількість одиниць програмного забезпечення відповідного виду, шт.;

$К_і$ – коефіцієнт, що враховує інсталяцію, налагодження програмного забезпечення, $К_і$ = (1,1…1,12);

к – кількість видів програмного забезпечення.

Зроблені розрахунки зводимо до таблиці 4.4.

Таблиця 4.4 – Витрати на придбання програмного забезпечення

Найменування програмного забезпечення	Ціна за одиницю, грн.	Витрачено	Вартість програмного забезпечення, грн.
SQL Server Standard	8569	1	8569
VS Code Professional	1845	1	1845
Всього, з врахуванням коефіцієнта інсталяції та налагодження			10414

Амортизація обладнання. Амортизація обладнання, комп'ютерів та приміщень, які використовувались під час (чи для) виконання даного етапу роботи.

У спрощеному вигляді амортизаційні відрахування А в цілому бути розраховані за формулою:

$$А = \frac{Ц_б}{Т_в} \cdot \frac{t}{12}, \quad (4.7)$$

де $Ц_б$ – загальна балансова вартість всього обладнання, комп'ютерів, приміщень тощо, що використовувались для виконання даного етапу роботи, грн.;

t – термін використання основного фонду, місяці;

$Т_в$ – термін корисного використання основного фонду, роки.

Зроблені розрахунки зводимо до таблиці 4.5.

Таблиця 4.5 – Амортизаційні відрахування за видами основних фондів

Найменування	Балансова вартість, грн.	Строк корисного використання, років	Термін використання, місяців	Сума амортизації, грн.
Настільний ПК	35000	2	1,5	2188
Монітор	9000	2	1,5	563
Стіл	6000	5	1,5	150
Стілець	8000	5	1,5	200
Мишка	1000	2	1,5	63
Клавіатура	1500	2	1,5	94
Навушники	2000	2	1,5	125
Всього	3383			

Витрати на електроенергію для науково-виробничих цілей. Витрати на силову електроенергію $В_{е.}$ Витрати на електроенергію зображені у таблиці 4.6.

Таблиця 4.6 – Витрати на електроенергію

Найменування обладнання	Потужність, кВт	Тривалість годин роботи
Настільний ПК	0,35	300
Монітор	0,1	300
Освітлення	0,034	130

Якщо ця стаття має суттєве значення для виконання даного етапу роботи, розраховуються за формулою:

$$Ве = \sum \frac{W_i \cdot t_i \cdot Ц_е \cdot К_{впі}}{ККД} = \tag{4.8}$$

$$= \frac{0{,}35 \cdot 300 \cdot 7{,}5 \cdot 0{,}95}{0{,}98} + \frac{0{,}1 \cdot 300 \cdot 7{,}5 \cdot 0{,}95}{0{,}98} +$$

$$+ \frac{0{,}034 \cdot 130 \cdot 7{,}5 \cdot 0{,}95}{0{,}98} = 1014 \text{ (грн.)},$$

де W_i – встановлена потужність обладнання, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год.;

Це – вартість 1 кВт електроенергії, грн.;

$К_{впі}$ – коефіцієнт використання потужності;

ККД – коефіцієнт корисної дії обладнання.

Інші витрати. До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$Ів = \left(З_о + З_р\right) \cdot \frac{Н_{ів}}{100\%} = (133907 + 22698{,}4) \cdot \frac{65}{100} = 101794 \text{ (грн.)}, \quad (4.9)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати».

Накладні (загальновиробничі) витрати. До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$\text{Внзв} = \left(З_о + З_р\right) \cdot \frac{Н_{нзв}}{100\%} = (133907 + 22698{,}4 + 101794) \cdot \frac{145}{100} = \quad (4.10)$$
$$= 374678 \text{ (грн.)},$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати».

Витрати на проведення науково-дослідної роботи. Витрати на проведення науково-дослідної роботи розраховуються як сума всіх попередніх статей витрат за формулою:

$$В_{заг} = З_о + З_р + З_{дод} + З_н + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + \quad (4.11)$$
$$+ I_в + В_{нзв} =$$
$$= 133907 + 22698{,}4 + 18793 + 35080 + 10414 + 3383 +$$
$$+ 1014 + 101794 + 374678 = 701177 \text{ (грн.)}.$$

Загальні витрати. Загальні витрати ЗВ на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються за формулою:

$$ЗВ = \frac{В_{заг}}{\eta} = \frac{701177}{0{,}9} = 779735 \text{ (грн.)}, \quad (4.12)$$

де η – коефіцієнт, що характеризує етап виконання науково-дослідної роботи.

Оскільки, якщо науково-технічна розробка знаходиться на стадії впрвадження, то η=0,9.

4.3 Розрахунок економічної ефективності науково-технічної розробки інформаційної технології надання рекомендацій для догляду за кімнатними рослинами за її можливої комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

В даному випадку відбувається розробка засобу, тому основу майбутнього економічного ефекту буде формувати: ΔN – збільшення кількості споживачів, яким надається відповідна інформаційна послуга в аналізовані періоди часу; N – кількість споживачів, яким надавалась відповідна інформаційна послуга у році до впровадження результатів нової науково-технічної розробки; $Ц_б$ – вартість послуги у році до впровадження інформаційної системи; $\pm\Delta Ц_о$ – зміна вартості послуги (зростання чи зниження) від впровадження результатів науково-технічної розробки в аналізовані періоди часу [34, 35].

Можливе збільшення чистого прибутку у потенційного інвестора ΔПі для кожного із років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховується за формулою:

$$\Delta П = (\pm\Delta Ц_о \cdot N + Ц_о \cdot \Delta N_i)_i \cdot \lambda \cdot \rho \cdot \left(1 - \frac{\vartheta}{100}\right), \quad (4.13)$$

де ±ΔЦ – зміна основного якісного показника від впровадження результатів науково-технічної розробки в аналізованому році, зазвичай, таким показником може бути зміна ціни реалізації одиниці нової розробки в аналізованому році (відносно року до впровадження цієї розробки). $\pm\Delta Ц_о$ може мати як додатне, так і від'ємне значення (від'ємне – при зниженні ціни відносно року до впровадження цієї розробки, додатне – при зростанні ціни),

N – основний кількісний показник, який визначає величину попиту на аналогічні чи подібні розробки у році до впровадження результатів нової науково-технічної розробки,

$Ц_о$ – основний якісний показник, який визначає ціну реалізації нової науково-технічної розробки в аналізованому році,

$Ц_б$ – основний якісний показник, який визначає ціну реалізації існуючої (базової) науково-технічної розробки у році до впровадження результатів,

ΔN – зміна основного кількісного показника від впровадження результатів науково-технічної розробки в аналізованому році. Зазвичай таким показником може бути зростання попиту на науково-технічну розробку в аналізованому році (відносно року до впровадження цієї розробки),

λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2024 році ставка податку на додану вартість становить 20%, а коефіцієнт $\lambda = 0{,}8333$,

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту (послуги). Рекомендується брати $\rho = 0{,}2...0{,}5$; ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2024 році $\vartheta = 18\%$.

Очікуваний термін життєвого циклу розробки 1 рік, тому:

$$\Delta П = \big((139 - 99) \cdot 290000 - (250000 - 290000) \cdot 139\big) \cdot 0{,}8333 \times \quad (4.14)$$
$$\times\, 0.3 \cdot \left(1 - \frac{18}{100}\right) = 3760099 \text{ (грн.)}.$$

Далі розраховують приведену вартість збільшення всіх чистих прибутків ПП, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t} = \frac{3760099}{(1+0{,}1)^1} = 3418272 \text{ (грн.)}, \quad (4.15)$$

де ΔП – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн.,

Т – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки (приймаємо Т=1 рік) ,

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =0,05…0,15,

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

Далі розраховують величину початкових інвестицій PV , які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки. Для цього можна використати формулу:

$$PV = k_{інв} \cdot ЗВ = 2 \cdot 779735 = 1559470 \text{ (грн.)}. \quad (4.16)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо. Зазвичай $k_{інв}$=2...5, але може бути і більшим,

ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

Тоді абсолютний економічний ефект $Е_{абс}$ або чистий приведений дохід для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$Е_{абс} = ПП - PV = 3418272 - 1559470 = 1858802 \text{ (грн.)}, \quad (4.17)$$

де ПП – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, грн.,

PV – теперішня вартість початкових інвестицій, грн.

Оскільки $Е_{абс}>0$, то можемо припустити про потенційну зацікавленість інвесторів у розробці.

Для остаточного прийняття рішення з цього питання необхідно розрахувати внутрішню економічну дохідність Ев або показник внутрішньої норми дохідності вкладених інвестицій та порівняти її з так званою бар'єрною ставкою дисконтування, яка визначає ту мінімальну внутрішню економічну дохідність, нижче якої інвестиції в будь-яку науково-технічну розробку вкладати буде економічно недоцільно.

Внутрішня економічна дохідність інвестицій Ев, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки, розраховується за формулою:

$$Е_в = \sqrt[Т_ж]{1 + \frac{Еабс}{PV}} - 1 = \sqrt[1]{1 + \frac{1858802}{1559470}} - 1 = 0{,}48, \quad (4.18)$$

де $Т_ж$ – життєвий цикл розробки, роки.

Визначимо бар'єрну ставку дисконтування $\tau_{мін}$, тобто мінімальну внутрішню економічну дохідність інвестицій, нижче якої кошти у впровадження науково-технічної розробки та її комерціалізацію вкладатися не будуть[35].

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$ визначається за формулою:

$$\tau_{мін} = d + f = 0{,}12 + 0{,}05 = 0{,}17, \quad (4.19)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках. В 2024 році в Україні d =0,9...0,12;

f – показник, що характеризує ризикованість вкладення інвестицій; зазвичай величина f =0,05...0,5, але може бути і значно вищою.

Оскільки $E_в$=0,48>$\tau_{мін}$=0,17, то потенційний інвестор може бути зацікавлений у фінансуванні впровадження науково-технічної розробки та виведенні її на ринок, тобто в її комерціалізації.

Далі розраховуємо період окупності інвестицій $T_о$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_о = \frac{1}{E_в} = \frac{1}{0{,}48} = 2{,}08 \text{ (року)}. \quad (4.20)$$

Оскільки $T_о$<1…3-х років, то це свідчить про комерційну привабливість науково-технічної розробки інформаційної технології надання рекомендацій для догляду за кімнатними рослинами і може спонукати потенційного інвестора профінансувати впровадження цієї розробки та виведення її на ринок.

4.4 Висновок до розділу 4

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Інформаційна технологія надання рекомендацій для догляду за кімнатними рослинами» становить 30,7 балів, що свідчить про вище середнього рівня комерційного потенціалу розробки.

Загальні витрати на проведення науково-дослідної роботи становлять 779735 грн. При оцінці економічної ефективності було встановлено, що абсолютний економічний ефект становить 1858802 грн, що свідчить про потенційну зацікавленість інвесторів у розробці.

Внутрішня економічна дохідність інвестицій становить 0,48, що значно перевищує бар'єрну ставку дисконтування (0,17). Це вказує на те, що потенційний інвестор може бути зацікавлений у фінансуванні впровадження науково-технічної розробки та виведенні її на ринок.

Термін окупності становить 2,08 роки, що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження цієї розробки та виведення її на ринок. Отже, можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Інформаційна технологія надання рекомендацій для догляду за кімнатними рослинами».

ВИСНОВКИ

Всі задачі, поставлені для реалізації інформаційної технології надання рекомендацій для догляду за кімнатними рослинами були виконані в повному обсязі, а саме: проведено аналіз існуючих систем надання рекомендацій для догляду за рослинами та обґрунтовано доцільність розробки інформаційної технології; розроблено математичну модель для надання рекомендацій з догляду за кімнатними рослинами; розглянуто методи та технології, які можуть бути використані для надання персоналізованих рекомендацій для догляду за кімнатними рослинами; спроектовано архітектуру та структуру програмного модуля для надання рекомендацій з догляду за кімнатними рослинами; проаналізовано та обґрунтовано вибір інструментів для розробки; програмно реалізовано інформаційну технологію надання рекомендацій для догляду за кімнатними рослинами; проведено тестування розробленої програми та проаналізовано отримані результати;и економічно обґрунтовано доцільність розробки.

При виконанні магістерської кваліфікаційної роботи реалізовано інформаційну технологію надання рекомендацій для догляду за кімнатними рослинами.

Проведено аналіз технічних рішень існуючих технологій надання рекомендацій для догляду за рослинами та доведено доцільність розробки інформаційної технології надання рекомендацій для догляду за кімнатними рослинами. Розглянуті системи-аналоги є корисними та доволі актуальними, проте усі вони не дають можливості відслідковування за рослинами та можливості підбору їх за вподобаннями користувача.

На основі аналізу існуючих моделей та методів, було обґрунтовано використання в інформаційній технології математичної багатокритеріальної лінійної моделі. Модель відповідає за підбір рослини на основі вимог та уподобань користувача.

Для реалізації розробки була розроблена структура інформаційної технології надання рекомендацій для догляду за кімнатними рослинами, створено UML-діаграму послідовності та розроблено схему алгоритму роботи.

Крім того, в роботі розроблено структуру інформаційної технології надання рекомендацій для догляду за кімнатними рослинами та спроектовано схему алгоритму роботи системи. Обґрунтовано вибір архітектури інформаційної технології надання рекомендацій для догляду за кімнатними рослинами. Здійснено проектування та розробку інформаційної технології надання рекомендацій для догляду за кімнатними рослинами.

Для розробки була використана мова програмування C#, для управління базою даних – SQL, а платформами для реалізації обрано WEB-браузери.

Розроблена інформаційна технологія успішно пройшла тестування та в порівнянні з системами-аналогами надає користувачам щонайменше 3 додаткові можливості, а саме: додано українську локалізацію, можливість контролю виконання дії з рослиною та рекомендація рослини за уподобанням користувача.

Економічно обґрунтовано доцільність розробки інформаційної технології надання рекомендацій для догляду за кімнатними рослинами. Було спрогнозовано орієнтовану величину витрат по кожній зі статей витрат, результатом стала сума 779735 гривень. Шляхом аналізу розрахунків було доведено, що розроблена інформаційна технологія матиме абсолютний економічний ефект у розмірі 1858802 грн, а внутрішня економічна дохідність інвестицій (0,48) значно перевищує бар'єрну ставку дисконтування (0,17). Період окупності інформаційної технології надання рекомендацій для догляду за кімнатними рослинами складе близько 2,08 років, що свідчить про її комерційну привабливість для потенційних інвесторів.

Поставлена мета, а саме розширення функціональних можливостей інформаційної технології надання рекомендацій для догляду за кімнатними рослинами була досягнута за рахунок впровадження української локалізації, можливість контролю виконання дії з рослиною та рекомендація рослини за

уподобанням користувача. В розробці враховано те, що користувач може зберігати понад 1000 шт. кімнатних рослин.

Це дозволило розробити зручний клієнтський додаток для інформаційної технології надання рекомендацій для догляду за кімнатними рослинами, який відповідає всім запитам користувача, інтуїтивно зрозумілий та має суттєві переваги порівняно з іншими системами-аналогами на ринку.

Отже, розроблена інформаційної технології надання рекомендацій для догляду за кімнатними рослинами відповідає заданим вимогам.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бортник А. О., Крилик Л. В. Перспективи розробки інформаційної технології надання рекомендацій для догляду за кімнатними рослинами. Матеріали LIII науково-технічної конференції підрозділів Вінницького національного технічного університету, Вінниця, 20−22 березня 2024 р. [Електронний ресурс]. URL: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2024/paper/view/20385/16890 (дата звернення 10.09.2024).

2. Свідоцтво про реєстрацію авторського права на твір № 130259 від 01.10.2024 р. Комп'ютерна програма «Інформаційна технологія надання рекомендацій для догляду за кімнатними рослинами» / Крилик Л. В., Бортник А. О. ДО «Український національний офіс інтелектуальної власності та інновацій» (УКРНОІВІ). – 2 с.

3. Користь домашніх рослин [Електронний ресурс]. URL: https://floren.com.ua/ua/publications/polza_domashnih_rasteniy/?srsltid=afmbooqfe4je-lcufvw-_tu3eqm6oa0vokceziv7xs6zkcguvayyk9w1 (дата звернення 10.10.2024).

4. Переваги кімнатних рослин [Електронний ресурс]. URL: https://kvitofor.ua/ua/publications/perevagi-kimnatnih-roslin/ (дата звернення 10.10.2024).

5. FlowersUA. Догляд за кімнатними рослинами [Електронний ресурс]. URL: https://flowers.ua/ua/articles/dohliad-za-kimnatnymy-roslynamy (дата звернення 10.09.2024).

6. Blossom [Електронний ресурс].
URL: https://blossomplant.com/features/plant-identification (дата звернення: 12.09.2024).

7. PlantNet [Електронний ресурс]. URL: https://identify.plantnet.org/uk (дата звернення: 12.09.2024).

8. Gardenia [Електронний ресурс]. URL: http://getgardenia.co/ (дата звернення: 12.09.2024).

9. Автоматизація поливу [Електронний ресурс]. URL:

https://www.smarthouse.ua/ua/umnyj-dom-poliv.html (дата звернення: 12.10.2024).

10. Procedia Computer Science [Електронний ресурс]. URL: https://www.sciencedirect.com/science/article/pii/S1877050918312079 (дата звернення: 12.10.2024).

11. Expert system [Електронний ресурс]. URL: https://www.techtarget.com/searchenterpriseai/definition/expert-system (дата звернення: 12.10.2024).

12. Mathematical Modeling [Електронний ресурс]. URL: https://www.sciencedirect.com/topics/mathematics/mathematical-modeling (дата звернення: 12.09.2024).

13. Методичні вказівки до виконання магістерських кваліфікаційних робіт для студентів спеціальності 122 «Комп'ютерні науки» [Електронний ресурс] / уклад.: А. А. Яровий, О. К. Колесницький. Вінниця : ВНТУ, 2023. (PDF, 58 с.)

14. Multi-Criteria Decision Analysis (MCDA/MCDM) [Електронний ресурс]. URL: https://www.1000minds.com/decision-making/what-is-mcdm-mcda (дата звернення: 12.09.2024).

15. Weighting methods for multi-criteria decision making technique [Електронний ресурс]. URL: https://www.researchgate.net/publication/335806989_Weighting_methods_for_multi-criteria_decision_making_technique (дата звернення: 12.09.2024).

16. Determining Weights in Multi-Criteria Decision [Електронний ресурс]. URL: https://www.mdpi.com/2227-7390/8/2/191 (дата звернення: 12.09.2024).

17. Normalization [Електронний ресурс]. URL: https://www.codecademy.com/article/normalization (дата звернення: 12.09.2024).

18. Data Normalization [Електронний ресурс]. URL: https://www.sciencedirect.com/topics/computer-science/data-normalization (дата звернення: 12.09.2024).

19. Analytical Hierarchy Process [Електронний ресурс]. URL: https://www.sciencedirect.com/topics/social-sciences/analytical-hierarchy-process

(дата звернення: 12.09.2024).

20. Multi-criteria optimization in regression [Електронний ресурс]. URL: https://link.springer.com/article/10.1007/s10479-021-03990-9 (дата звернення: 12.09.2024).

21. What is a UML diagram? [Електронний ресурс]. URL: https://miro.com/diagramming/what-is-a-uml-diagram/ (дата звернення: 12.10.2024).

22. Архітектура клієнт-сервер [Електронний ресурс]. URL: http://inter.ptngu.com/kompyuterni-merezhi/arhitektura-kliyent-server (дата звернення: 15.09.2024).

23. Яровий А. А., Крилик Л. В., Козловський А. В. Сучасні інформаційні технології у сфері штучного інтелекту : електронний навчальний посібник комбінованого (локального та мережного) використання [Електронний ресурс]. Вінниця : ВНТУ, 2023. 145 с.

24. Клієнт-серверна архітектура [Електронний ресурс]. URL: https://javarush.com/ua/quests/lectures/ua.questservlets.level14.lecture00 (дата звернення: 15.09.2024).

25. C# [Електронний ресурс]. URL: https://dotnet.microsoft.com/en-us/languages/csharp (дата звернення: 15.09.2024).

26. Java [Електронний ресурс]. URL: https://aws.amazon.com/what-is/java/ (дата звернення: 15.09.2024).

27. What Is Python Used For? [Електронний ресурс]. URL: https://www.coursera.org/articles/what-is-python-used-for-a-beginners-guide-to-using-python (дата звернення: 15.09.2024).

28. ASP.NET MVC Overview [Електронний ресурс]. URL: https://learn.microsoft.com/en-us/aspnet/mvc/overview/older-versions-1/overview/asp-net-mvc-overview (дата звернення: 16.09.2024).

29. Overview of React.js [Електронний ресурс]. URL: https://www.patterns.dev/react/ (дата звернення: 16.09.2024).

30. An Overview of Angular [Електронний ресурс]. URL:

https://leobit.com/blog/why-to-use-angular-in-an-overview-of-angular-pros-and-cons/ (дата звернення: 16.09.2024).

31. Getting started with Vue [Електронний ресурс]. URL: https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Vue_getting_started (дата звернення: 16.09.2024).

32. ADO.NET [Електронний ресурс]. URL: https://learn.microsoft.com/en-us/dotnet/framework/data/adonet/ (дата звернення: 16.10.2024).

33. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад.: В. О. Козловський, О. Й. Лесько, В. В. Кавецький. Вінниця : ВНТУ, 2021. 42 с.

34. Як розраховується прибуток [Електронний ресурс]. URL: https://support.joinposter.com/uk/articles/7067856 (дата звернення: 16.10.2024).

35. Собівартість послуг: особливості формування [Електронний ресурс]. URL: https://interbuh.com.ua/ua/documents/ib/10166/138487 (дата звернення: 16.10.2024).

Додаток А (обов'язковий)
Протокол перевірки кваліфікаційної роботи на наявність текстових запозичень

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Інформаційна технологія надання рекомендацій для догляду за кімнатними рослинами

Тип роботи: магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ кафедра комп'ютерних наук, ФІТА
(кафедра, факультет)

Показники звіту подібності Turnitin

Оригінальність 87,00% Схожість 13,00%

Аналіз звіту подібності (відмітити потрібне):

✓ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Ознайомлені з повним звітом подібності, який був згенерований системою Turnitin щодо роботи.

Автор роботи _____ Бортник А.О.

Керівник роботи _____ Крилик Л.В.

Опис прийнятого рішення

Магістерську кваліфікаційну роботу допущено до захисту

Особа, відповідальна за перевірку _____ Озеранський В.С.

Додаток Б (обов'язковий)

Лістинг програми

```
namespace DataModule
{
   public class Plant
   {
      public Plant()
      {
      }
      public Plant(int id, string name, string description, int wateringPeriod)
      {
         Id = id;
         Name = name;
         Description = description;
         WateringPeriod = wateringPeriod;
      }
      public Plant(int id, string name, string uniqueImageName, string description)
      {
         Id = id;
         Name = name;
         UniqueImageName = uniqueImageName;
         Description = description;
      }
      public int Id { get; set; }
      public string Name { get; set; }
      public string UniqueImageName { get; set; }
      public string Description { get; set; }
      public int WateringPeriod { get; set; }
      public int FertalizationPeriod { get; set; }
      public int TransplantPeriod { get; set; }
      public string GetImagePath() => $"~/Content/images/{UniqueImageName}";
   }
}
namespace DataModule
{
   public class User
   {
      public int Id { get; set; }
      public string Name { get; set; }
      public string Username { get; set; }
```

```
        public string Email { get; set; }
        public string PasswordHash { get; set; }
        public string Salt { get; set; }
    }
}
using DataModule.Seeders.Strategies;
using System;
using System.Threading.Tasks;
namespace DataModule
{
    public class UsersPlant : Plant
    {
        public int PlantUserRelationsId { get; set; }
        public DateTime NextWateringDate { get; set; }
        public DateTime NextFertalizationDate { get; set; }
        public DateTime NextTransplantDate { get; set; }
        public DateTime LastWateringDate { get; set; }
        public DateTime LastFertalizationDate { get; set; }
        public DateTime LastTransplantDate { get; set; }
        public void SetNextWateringDate(int userId)
        {
            LastWateringDate = DateTime.Now;
            NextWateringDate = DateTime.Now.AddDays(WateringPeriod);
            Task.Run(() => SqlPlantReader.UpdateNextLastPlantProperty(PlantUserRelationsId, 1, NextWateringDate));
        }
        public void SetNextFertalizationDate(int userId)
        {
            LastFertalizationDate = DateTime.Now;
            NextFertalizationDate = DateTime.Now.AddDays(FertalizationPeriod);
            Task.Run(() => SqlPlantReader.UpdateNextLastPlantProperty(PlantUserRelationsId, 2,
NextFertalizationDate));
        }
        public void SetNextTransplantDate(int userId)
        {
            LastTransplantDate = DateTime.Now;
            NextTransplantDate = DateTime.Now.AddDays(TransplantPeriod);
            Task.Run(() => SqlPlantReader.UpdateNextLastPlantProperty(PlantUserRelationsId, 3, NextTransplantDate));
        }
    }
}
using System;
using System.Collections.Generic;
```

```
using System.Configuration;
using System.Data;
using System.Data.SqlClient;
using System.Linq;
using System.Threading.Tasks;
namespace DataModule.Seeders.Strategies
{
   public class SqlPlantReader
   {
      private static readonly string _connectionString =
ConfigurationManager.ConnectionStrings["SqlConnectionString"].ConnectionString;
      public bool AddPlantToUserPlants(int userId, int plantId)
      {
         string sqlCom = @"AddPlantToUserPlants";
         using (SqlConnection connection = new SqlConnection(_connectionString))
         {
            SqlCommand cmd = new SqlCommand(sqlCom, connection);
            cmd.CommandType = CommandType.StoredProcedure;
            cmd.Parameters.AddWithValue("@UserId", userId);
            cmd.Parameters.AddWithValue("@PlantId", plantId);
            connection.Open();
            cmd.ExecuteNonQuery();
         }
         return true;
      }
      public async Task RemovePlantFromUserAsync(int userId, int plantId)
      {
         string sqlCom = @"Delete from PlantUserRelations where UserId = @UserId and PlantId = @PlantId";
         using (SqlConnection connection = new SqlConnection(_connectionString))
         {
            SqlCommand cmd = new SqlCommand(sqlCom, connection);
            cmd.Parameters.AddWithValue("@UserId", userId);
            cmd.Parameters.AddWithValue("@PlantId", plantId);
            connection.Open();
            await cmd.ExecuteNonQueryAsync();
         }
      }
      public IEnumerable<Plant> GetListOfPlants()
      {
         string sqlCom = @"select * from Plants";
         return ExecuteQuery(sqlCom);
      }
```

```
public List<UsersPlant> GetListOfUserPlantsById(int userId)
{
   string sqlCom = @"select p.*, up.NextTransplantDate, up.NextFertalizationDate, up.NextWateringDate,
                  up.LastTransplantDate, up.LastFertalizationDate, up.LastWateringDate, up.PlantUserRelationsId
                  from Plants p
                  join PlantUserRelations pu on pu.PlantId = p.PlantId
                  join UserPlantsState up on pu.PlantUserRelationsId = up.PlantUserRelationsId
                  where pu.UserId = @Id";
   return ExecuteQueryForUserPlant(sqlCom, userId).ToList();
}
public Plant GetPlant(int id)
{
   string sqlCom = @"select top 1 * from Plants where PlantId = @Id";
   Plant plant = ExecuteQuery(sqlCom, id).FirstOrDefault();
   return plant;
}
public static void UpdateNextLastPlantProperty(int plantUserRelationsId, int propNum, DateTime nextDate)
{
   string sqlCom = $@"UpdatePlantDates";
   using (SqlConnection connection = new SqlConnection(_connectionString))
   {
      SqlCommand cmd = new SqlCommand(sqlCom, connection);
      cmd.CommandType = CommandType.StoredProcedure;
      cmd.Parameters.AddWithValue("@PlantUserRelationsId", plantUserRelationsId);
      cmd.Parameters.AddWithValue("@NextDate", nextDate);
      cmd.Parameters.AddWithValue("@LastDate", DateTime.Now);
      cmd.Parameters.AddWithValue("@UpdateType", propNum);
      connection.Open();
      cmd.ExecuteNonQuery();
   }
}
private IEnumerable<Plant> ExecuteQuery(string sqlCom, int? id = null)
{
   List<Plant> plants = new List<Plant>();
   using (SqlConnection sqlConnection = new SqlConnection(_connectionString))
   {
      SqlCommand cn = new SqlCommand(sqlCom, sqlConnection);
      if (id != null)
         cn.Parameters.AddWithValue("@Id", id);
      sqlConnection.Open();
      using (SqlDataReader reader = cn.ExecuteReader())
      {
```

```
            while (reader.Read())
            {
                int PlantId = reader.GetInt32(0);
                string Name = reader.GetString(1);
                string Descripption = reader.GetString(2);
                int WateringPeriod = reader.GetInt32(3);
                int TransplantPeriod = reader.GetInt32(4);
                int FertalizationPeriod = reader.GetInt32(5);
                string UniqueImageName = reader.GetString(6);
                Plant plant = new Plant()
                {
                    Id = PlantId,
                    Name = Name,
                    Description = Descripption,
                    WateringPeriod = WateringPeriod,
                    TransplantPeriod = TransplantPeriod,
                    FertalizationPeriod = FertalizationPeriod,
                    UniqueImageName = UniqueImageName
                };
                plants.Add(plant);
            }
        }
    }
    return plants;
}
private IEnumerable<UsersPlant> ExecuteQueryForUserPlant(string sqlCom, int? id = null)
{
    List<UsersPlant> plants = new List<UsersPlant>();
    using (SqlConnection sqlConnection = new SqlConnection(_connectionString))
    {
        SqlCommand cn = new SqlCommand(sqlCom, sqlConnection);
        if (id != null)
            cn.Parameters.AddWithValue("@Id", id);
        sqlConnection.Open();
        using (SqlDataReader reader = cn.ExecuteReader())
        {
            while (reader.Read())
            {
                int PlantId = reader.GetInt32(0);
                string Name = reader.GetString(1);
                string Descripption = reader.GetString(2);
                int WateringPeriod = reader.GetInt32(3);
```

```
                int TransplantPeriod = reader.GetInt32(4);
                int FertalizationPeriod = reader.GetInt32(5);
                string UniqueImageName = reader.GetString(6);
                DateTime nextTransplantDate = reader.GetDateTime(7);
                DateTime nextFertalizationDate = reader.GetDateTime(8);
                DateTime nextWateringDate = reader.GetDateTime(9);
                DateTime lastTransplantDate = reader.GetDateTime(10);
                DateTime lastFertalizationDate = reader.GetDateTime(11);
                DateTime lastWateringDate = reader.GetDateTime(12);
                int plantUserRelationsId = reader.GetInt32(13);
                UsersPlant plant = new UsersPlant()
                {
                   Id = PlantId,
                   Name = Name,
                   Description = Descripption,
                   WateringPeriod = WateringPeriod,
                   TransplantPeriod = TransplantPeriod,
                   FertalizationPeriod = FertalizationPeriod,
                   UniqueImageName = UniqueImageName,
                   NextTransplantDate = nextTransplantDate,
                   NextFertalizationDate = nextFertalizationDate,
                   NextWateringDate = nextWateringDate,
                   LastTransplantDate = lastTransplantDate,
                   LastFertalizationDate = lastFertalizationDate,
                   LastWateringDate = lastWateringDate,
                   PlantUserRelationsId = plantUserRelationsId
                };
                plants.Add(plant);
             }
          }
       }
       return plants;
    }
  }
}
using System;
using System.Configuration;
using System.Data;
using System.Data.SqlClient;
namespace DataModule.Seeders.Strategies
{
  public class SqlUserAccess
```

```
{
    private readonly string _connectionString;
    public SqlUserAccess()
    {
        _connectionString = ConfigurationManager.ConnectionStrings["SqlConnectionString"].ConnectionString;
    }
    public User GetUser(string username)
    {
        User user = null;
        string sqlQuery = "Select * from Users where Username = @Username";
        using (SqlConnection sqlConnection = new SqlConnection(_connectionString))
        {
            SqlCommand cmd = sqlConnection.CreateCommand();
            cmd.CommandText = sqlQuery;
            cmd.Parameters.AddWithValue("@Username", username);
            sqlConnection.Open();
            using (SqlDataReader reader = cmd.ExecuteReader())
            {
                while (reader.Read())
                {
                    int id = (int)reader["UserId"];
                    string userName = (string)reader["Username"];
                    string email = (string)reader["Email"];
                    string password = (string)reader["Password"];
                    string salt = (string)reader["Salt"];
                    user = new User()
                    {
                        Id = id,
                        Username = userName,
                        PasswordHash = password,
                        Email = email,
                        Salt = salt,
                    };
                }
            }
        }
        return user;
    }
    public User WriteUser(User user)
    {
        string sqlQuery = "InsertUser";
        int UserId = 0;
```

```
            using (SqlConnection sqlConnection = new SqlConnection(_connectionString))
            {
                SqlCommand cmd = sqlConnection.CreateCommand();
                cmd.CommandType = CommandType.StoredProcedure;
                cmd.CommandText = sqlQuery;
                cmd.Parameters.AddWithValue("@Username", user.Username);
                cmd.Parameters.AddWithValue("@Email", user.Email);
                cmd.Parameters.AddWithValue("@Password", user.PasswordHash);
                cmd.Parameters.AddWithValue("@Salt", user.Salt);
                SqlParameter param = new SqlParameter("@Id", 0);
                param.Direction = ParameterDirection.Output;
                cmd.Parameters.Add(param);
                sqlConnection.Open();
                cmd.ExecuteNonQuery();
                UserId = Convert.ToInt32(cmd.Parameters[4].Value);
            }
            user.Id = UserId;
            return user;
        }
        public string IsEmailOrUserNameExist(string email, string username)
        {
            string result = string.Empty;
            string sqlEmailQuery = "Select Count(Email) from Users where Email = @Email";
            string sqlUsernameQuery = "Select Count(Username) from Users where Username = @Username";
            using (SqlConnection sqlConnection = new SqlConnection(_connectionString))
            {
                SqlCommand cmd = sqlConnection.CreateCommand();
                cmd.CommandText = sqlEmailQuery;
                cmd.Parameters.AddWithValue("@Email", email);
                sqlConnection.Open();
                if (cmd.ExecuteNonQuery() != -1)
                    result = "Email already exists";
                cmd.CommandText = sqlUsernameQuery;
                cmd.Parameters.Clear();
                cmd.Parameters.AddWithValue("@Username", username);
                if (cmd.ExecuteNonQuery() != -1)
                    result = "Username already exists";
            }
            return result;
        }
    }
}
```

```
using DataModule;
using DataModule.Seeders.Strategies;
using ExpertSys.Data;
using ExpertSys.Seeders.SeedStrategies;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
namespace ExpertSys
{
    public class Survey
    {
        private readonly SurveySeederFromDb _surveySeeder;
        private readonly SqlPlantReader _dataSeeder;
        public List<QuestionModel> Questions { get; private set; }
        public int CurrentQuestion { get; set; }
        public Survey()
        {
            _surveySeeder = new SurveySeederFromDb();
            _dataSeeder = new SqlPlantReader();
            Questions = new List<QuestionModel>();
            CurrentQuestion = 0;
        }
        public async Task LoadAsync()
        {
            var questions = await _surveySeeder.SeedQuestionsAsync();
            foreach (QuestionModel questionModel in questions)
            {
                Questions.Add(questionModel);
            }
        }
        public async Task<List<Plant>> GetSimiliarPlantsAsync(PlantScoreModel userPlant)
        {
            List<Plant> similiarPlants = await _surveySeeder.GetSimiliarPlantsAsync(userPlant);
            if (!similiarPlants.Any())
            {
                similiarPlants = await _surveySeeder.GetTop10PlantsAsync();
            }
            return similiarPlants;
        }
        public void AddPlant(int userId, int plantId)
        {
            _dataSeeder.AddPlantToUserPlants(userId, plantId);
```

```
        }
    }
}
using ExpertSys.Data;
using System.Collections.Generic;
using System.Threading.Tasks;
namespace ExpertSys.Seeders.SeedStrategies
{
    public interface ISurveySeedStrategy
    {
        Task<IEnumerable<PlantScoreModel>> SeedPlantsAsync();
        Task<IEnumerable<QuestionModel>> SeedQuestionsAsync();
    }
}
using DataModule;
using ExpertSys.Data;
using System.Collections.Generic;
using System.Configuration;
using System.Data.SqlClient;
using System.Linq;
using System.Threading.Tasks;
namespace ExpertSys.Seeders.SeedStrategies
{
    public class SurveySeederFromDb : ISurveySeedStrategy
    {
        private readonly string _connectionString =
ConfigurationManager.ConnectionStrings["SqlConnectionString"].ConnectionString;
        public async Task<List<Plant>> GetSimiliarPlantsAsync(PlantScoreModel plantScoreModel)
        {
            List<Plant> plants = new List<Plant>();
            string sqlCom = "CompareUserPlant";
            using (SqlConnection sqlConn = new SqlConnection(_connectionString))
            {
                SqlCommand cmd = new SqlCommand(sqlCom, sqlConn);
                cmd.CommandType = System.Data.CommandType.StoredProcedure;
                cmd.Parameters.AddWithValue("@WateringFrequency", plantScoreModel.WateringFrequency);
                cmd.Parameters.AddWithValue("@ComplexityOfCare", plantScoreModel.ComplexityOfCare);
                cmd.Parameters.AddWithValue("@Size", plantScoreModel.Size);
                cmd.Parameters.AddWithValue("@Temperature", plantScoreModel.Temperature);
                cmd.Parameters.AddWithValue("@Lighting", plantScoreModel.Lighting);
                cmd.Parameters.AddWithValue("@ColorSaturation", plantScoreModel.ColorSaturation);
                sqlConn.Open();
```

```
        using (SqlDataReader reader = await cmd.ExecuteReaderAsync())
        {
            while (reader.Read())
            {
                plants.Add(ReadPlant(reader));
            }
        }
    }
    return plants;
}
public async Task<List<Plant>> GetTop10PlantsAsync()
{
    List<Plant> plants = new List<Plant>();
    string sqlCom = "GetTop10Plants";
    using (SqlConnection sqlConn = new SqlConnection(_connectionString))
    {
        SqlCommand cmd = new SqlCommand(sqlCom, sqlConn);
        cmd.CommandType = System.Data.CommandType.StoredProcedure;
        sqlConn.Open();
        using (SqlDataReader reader = await cmd.ExecuteReaderAsync())
        {
            while (reader.Read())
            {
                plants.Add(ReadPlant(reader));
            }
        }
    }
    return plants;
}
public async Task<IEnumerable<PlantScoreModel>> SeedPlantsAsync()
{
    string sqlCom = @"select * from PlantScoreModels";
    List<PlantScoreModel> plants = new List<PlantScoreModel>();
    using (SqlConnection sqlConnection = new SqlConnection(_connectionString))
    {
        SqlCommand cn = new SqlCommand(sqlCom, sqlConnection);
        sqlConnection.Open();
        using (SqlDataReader reader = await cn.ExecuteReaderAsync())
        {
            while (await reader.ReadAsync())
            {
                int PlantId = reader.GetInt32(1);
```

```
                int Size = reader.GetInt32(2);
                int ColorSaturation = reader.GetInt32(3);
                int Temperature = reader.GetInt32(5);
                int PlantType = reader.GetInt32(6);
                int WateringFrequency = reader.GetInt32(7);
                int ComplexityOfCare = reader.GetInt32(8);
                int Lighting = reader.GetInt32(9);
                PlantScoreModel plant = new PlantScoreModel()
                {
                    PlantId = PlantId,
                    Size = Size,
                    ColorSaturation = ColorSaturation,
                    PlantType = PlantType,
                    WateringFrequency = WateringFrequency,
                    ComplexityOfCare = ComplexityOfCare,
                    Temperature = Temperature,
                    Lighting = Lighting
                };
                plants.Add(plant);
            }
        }
    }
    return plants;
}
public async Task<IEnumerable<QuestionModel>> SeedQuestionsAsync()
{
    string sqlCom = @"select * from Questions";
    List<QuestionModel> questions = new List<QuestionModel>();
    using (SqlConnection sqlConnection = new SqlConnection(_connectionString))
    {
        SqlCommand cn = new SqlCommand(sqlCom, sqlConnection);
        sqlConnection.Open();
        using (SqlDataReader reader = await cn.ExecuteReaderAsync())
        {
            int QuestionNumber = 0;
            while (await reader.ReadAsync())
            {
                int QuestionId = reader.GetInt32(0);
                string QuestionText = reader.GetString(1);
                QuestionModel questionModel = new QuestionModel()
                {
                    Id = QuestionId,
```

```
                QuestionText = QuestionText,
                QuestionNumber = QuestionNumber
            };
            QuestionNumber++;
            questions.Add(questionModel);
        }
    }
    sqlCom = @"select * from Answers";
    cn.CommandText = sqlCom;
    using (SqlDataReader reader = await cn.ExecuteReaderAsync())
    {
        while (await reader.ReadAsync())
        {
            int AnswerId = reader.GetInt32(0);
            int QuestionId = reader.GetInt32(1);
            string AnswerText = reader.GetString(2);
            int AnswerValue = reader.GetInt32(3);
            SurveyOption answerModel = new SurveyOption()
            {
                Id = AnswerId,
                OptionText = AnswerText,
                OptionValue = AnswerValue
            };
            questions.Where(x => x.Id == QuestionId).
                First().Options.Add(answerModel);
        }
    }
  }
  return questions;
}
private Plant ReadPlant(SqlDataReader reader)
{
  int PlantId = reader.GetInt32(0);
  string Name = reader.GetString(1);
  string Descripption = reader.GetString(2);
  int WateringPeriod = reader.GetInt32(3);
  int TransplantPeriod = reader.GetInt32(4);
  int FertalizationPeriod = reader.GetInt32(5);
  string UniqueImageName = reader.GetString(6);
  Plant plant = new Plant()
  {
    Id = PlantId,
```

```
                Name = Name,
                Description = Descripption,
                WateringPeriod = WateringPeriod,
                TransplantPeriod = TransplantPeriod,
                FertalizationPeriod = FertalizationPeriod,
                UniqueImageName = UniqueImageName
            };
            return plant;
        }
    }
}
using System.Collections.Generic;
namespace ExpertSys.Data
{
    public class PlantScoreModel
    {
        private List<int> _list = new List<int>();
        public PlantScoreModel()
        {
            PropertyToFill = 0;
            for (int i = 0; i < 7; i++)
            {
                _list.Add(0);
            }
        }
        public int PlantId { get; set; }
        public int PropertyToFill { get; set; }
        public int WateringFrequency
        {
            get => _list[0];
            set => _list[0] = value;
        }
        public int ComplexityOfCare
        {
            get => _list[1];
            set => _list[1] = value;
        }
        public int PlantType
        {
            get => _list[2];
            set => _list[2] = value;
        }
```

```
        public int Size
        {
            get => _list[3];
            set => _list[3] = value;
        }
        public int ColorSaturation
        {
            get => _list[4];
            set => _list[4] = value;
        }
        public int Temperature
        {
            get => _list[5];
            set => _list[5] = value;
        }
        public int Lighting
        {
            get => _list[6];
            set => _list[6] = value;
        }
        public int this[int key]
        {
            get => _list[key];
            set => _list[key] = value;
        }
        public override bool Equals(object obj)
        {
            if (obj is PlantScoreModel)
            {
                var plant = obj as PlantScoreModel;
                if (plant.ColorSaturation == ColorSaturation &&
                    plant.PlantType == PlantType) return true;
            }
            return false;
        }
    }
}
using System.Collections.Generic;
namespace ExpertSys.Data
{
    public class QuestionModel
    {
```

```
        public int Id { get; set; }
        public string QuestionText { get; set; }
        public List<SurveyOption> Options { get; set; } = new List<SurveyOption>();
        public int QuestionNumber { get; set; }
    }
}
namespace ExpertSys.Data
{
    public class SurveyOption
    {
        public int Id { get; set; }
        public string OptionText { get; set; }
        public int OptionValue { get; set; }
    }
}
using ExpertSys.Data;
namespace PlantSite.Models
{
    public class OptionPlantModel
    {
        public QuestionModel Question { get; set; }
        public PlantScoreModel PlantScore { get; set; }
    }
}
using System;
using System.Security.Cryptography;
using System.Text;
namespace PlantSite.Models
{
    public static class PasswordHelper
    {
        public static string GenerateSalt()
        {
            var rng = new RNGCryptoServiceProvider();
            var buffer = new byte[16];
            rng.GetBytes(buffer);
            return Convert.ToBase64String(buffer);
        }
        public static string HashPassword(string password, string salt)
        {
            using (var sha256 = SHA256.Create())
            {
```

```
            var saltedPassword = string.Concat(password, salt);
            var saltedPasswordAsBytes = Encoding.UTF8.GetBytes(saltedPassword);
            var hash = sha256.ComputeHash(saltedPasswordAsBytes);
            return Convert.ToBase64String(hash);
        }
    }
    public static bool VerifyPassword(string enteredPassword, string storedHash, string storedSalt)
    {
        var hashOfEnteredPassword = HashPassword(enteredPassword, storedSalt);
        return hashOfEnteredPassword.Equals(storedHash);
    }
  }
}
namespace PlantSite.Models
{
  public class RegisterViewModel
  {
    public string Username { get; set; }
    public string Email { get; set; }
    public string Password { get; set; }
  }
}
using DataModule;
using DataModule.Seeders.Strategies;
using PlantSite.Models;
using System.Web.Mvc;
using System.Web.Security;
namespace PlantSite.Controllers
{
  [AllowAnonymous]
  public class AccountController : Controller
  {
    private SqlUserAccess _sqlUserAccess = new SqlUserAccess();
    [HttpGet]
    public ActionResult Login()
    {
        return View();
    }
    [HttpPost]
    public ActionResult Login(string username, string password)
    {
        var user = _sqlUserAccess.GetUser(username);
```

```
    if (user != null && PasswordHelper.VerifyPassword(password, user.PasswordHash, user.Salt))
    {
        FormsAuthentication.SetAuthCookie(user.Id.ToString(), false);
        return Json(new { success = true });
    }
    string error = "Invalid username or password";
    return Json(new { success = false, message = error }); ;
}
[HttpGet]
public ActionResult Register()
{
    return View();
}
[HttpPost]
public ActionResult Register(string username, string email, string password)
{
    if (!IsEmailOrUsernameExist(username, email, out string error))
    {
        return Json(new { success = false, message = error });
    }
    var salt = PasswordHelper.GenerateSalt();
    var hash = PasswordHelper.HashPassword(password, salt);
    User user = _sqlUserAccess.WriteUser(new User
    {
        Username = username,
        Email = email,
        PasswordHash = hash,
        Salt = salt
    });
    FormsAuthentication.SetAuthCookie(user.Id.ToString(), false);
    return Json(new { success = true });
}
public ActionResult Logout()
{
    FormsAuthentication.SignOut();
    return RedirectToAction("Index", "Home");
}
private bool IsEmailOrUsernameExist(string username, string email, out string error)
{
    error = string.Empty;
    error = _sqlUserAccess.IsEmailOrUserNameExist(username, email);
    if (string.IsNullOrEmpty(error))
```

```
            {
                return true;
            }
            return false;
        }
    }
}
using System.Web.Mvc;
namespace PlantSite.Controllers
{
    [AllowAnonymous]
    public class HomeController : Controller
    {
        [AllowAnonymous]
        public ActionResult Index()
        {
            if (User.Identity.IsAuthenticated)
            {
                return RedirectToAction("Index", "Main");
            }
            return View();
        }
    }
}
using DataModule;
using DataModule.Seeders.Strategies;
using System.Collections.Generic;
using System.Web.Mvc;
namespace PlantSite.Controllers
{
    [Authorize]
    public class MainController : Controller
    {
        SqlPlantReader _sqlPlantReader;
        public MainController()
        {
            _sqlPlantReader = new SqlPlantReader();
        }
        public ActionResult Index()
        {
            return View();
        }
```

```
    public ActionResult GetUpdatedPlants()
    {
        var model = GetDashboardItems();
        return PartialView("_PlantsPartial", model);
    }
    public ActionResult RemovePlant(List<UsersPlant> model, int plantNum)
    {
        model.RemoveAt(plantNum);
        return PartialView("_PlantsPartial", model);
    }
    public ActionResult GetPlantInfo(Plant plant)
    {
        return PartialView("_PlantDetails", plant);
    }
    public ActionResult UpdatePlant(List<UsersPlant> model, int plantNum, int propNumber)
    {
        ModelState.Clear();
        switch (propNumber)
        {
            case 1:
                model[plantNum].SetNextWateringDate(int.Parse(User.Identity.Name));
                break;
            case 2:
                model[plantNum].SetNextFertalizationDate(int.Parse(User.Identity.Name));
                break;
            case 3:
                model[plantNum].SetNextTransplantDate(int.Parse(User.Identity.Name));
                break;
            default:
                break;
        }
        return PartialView("UpdatePlantsView", model);
    }
    private List<UsersPlant> GetDashboardItems()
    {
        List<UsersPlant> plantsModel;
        int userId = int.Parse(User.Identity.Name);
        plantsModel = _sqlPlantReader.GetListOfUserPlantsById(userId);
        if (plantsModel == default) plantsModel = new List<UsersPlant>();
        return plantsModel;
    }
}
```

```
}
using DataModule;
using ExpertSys;
using ExpertSys.Data;
using PlantSite.Models;
using System.Collections.Generic;
using System.Threading.Tasks;
using System.Web.Mvc;
namespace PlantSite.Controllers
{
    [Authorize]
    public class SurveyController : Controller
    {
        private static Survey _survey;
        public async Task<ActionResult> Index()
        {
            _survey = new Survey();
            await _survey.LoadAsync();
            return View("Index", new OptionPlantModel()
            {
                Question = _survey.Questions[0],
                PlantScore = new PlantScoreModel()
                {
                    WateringFrequency = 0,
                    ComplexityOfCare = 0,
                    PlantType = 0,
                    Size = 0,
                    ColorSaturation = 0,
                    Temperature = 0,
                    Lighting = 0
                }
            }
            );
        }
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Index(OptionPlantModel optionPlantModel, string SelectedOption)
        {
            if (!ModelState.IsValid)
            {
                return PartialView("Index", optionPlantModel);
            }
```

```
            optionPlantModel.PlantScore[optionPlantModel.Question.QuestionNumber] = int.Parse(SelectedOption);
            int nextQuestionNumber = ++optionPlantModel.Question.QuestionNumber;
            if (optionPlantModel.Question.QuestionNumber < _survey.Questions.Count)
            {
                ModelState.Clear();
                optionPlantModel = new OptionPlantModel()
                {
                    Question = _survey.Questions[nextQuestionNumber],
                    PlantScore = optionPlantModel.PlantScore
                };
                return PartialView(optionPlantModel);
            }
            return Json(new { success = true, redirectUrl = Url.Action("SurveyResultAsync", optionPlantModel.PlantScore)
});
        }
        public async Task<ActionResult> SurveyResultAsync(PlantScoreModel plantScoreModel)
        {
            List<Plant> plants = await _survey.GetSimiliarPlantsAsync(plantScoreModel);
            if (plants == null)
            {
                plants = new List<Plant>();
            }
            return PartialView("SurveyResult", plants);
        }
        [HttpPost]
        public ActionResult AddPlant(int plantId)
        {
            _survey.AddPlant(int.Parse(User.Identity.Name), plantId);
            return Json(new { success = true });
        }
    }
}
USE [PlantHelper]
GO
/****** Object:  StoredProcedure [dbo].[UpdatePlantDates]    Script Date: 09.11.2024 10:39:46 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[UpdatePlantDates]
    @PlantUserRelationsId INT,
    @NextDate datetime,
```

```
    @LastDate datetime,
    @UpdateType INT
AS
BEGIN
    IF @UpdateType = 1
    BEGIN
        UPDATE UserPlantsState
        SET
            NextWateringDate = @NextDate,
            LastWateringDate = @LastDate
        WHERE PlantUserRelationsId = @PlantUserRelationsId;
    END
    ELSE IF @UpdateType = 2
    BEGIN
        UPDATE UserPlantsState
        SET
            NextFertalizationDate = @NextDate,
            LastFertalizationDate = @LastDate
        WHERE PlantUserRelationsId = @PlantUserRelationsId;
    END
    ELSE IF @UpdateType = 3
    BEGIN
        UPDATE UserPlantsState
        SET
            NextTransplantDate = @NextDate,
            LastTransplantDate = @LastDate
        WHERE PlantUserRelationsId = @PlantUserRelationsId;
    END
END;
USE [PlantHelper]
GO
/****** Object:  StoredProcedure [dbo].[GetTop10Plants]    Script Date: 09.11.2024 10:39:40 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER Procedure [dbo].[GetTop10Plants]
as
        SELECT TOP 10 p.*
FROM Plants p
JOIN (
    SELECT PlantId, COUNT(*) as PlantCount
```

```
    FROM PlantUserRelations
    GROUP BY PlantId
) pu ON p.PlantId = pu.PlantId
ORDER BY pu.PlantCount DESC;
USE [PlantHelper]
GO
/****** Object:  StoredProcedure [dbo].[CompareUserPlant]    Script Date: 09.11.2024 10:39:36 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[CompareUserPlant]
    @WateringFrequency INT,
    @ComplexityOfCare INT,
    @Size INT,
    @Temperature INT,
    @Lighting INT,
    @ColorSaturation INT
AS
BEGIN
    SELECT top 10 pl.*
    FROM PlantScoreModels p
         join Plants pl on pl.PlantId = p.PlantId
    WHERE (@WateringFrequency - 1 = p.WateringFrequency
         OR @WateringFrequency + 1 = p.WateringFrequency
         OR @WateringFrequency = p.WateringFrequency)
      AND (@ComplexityOfCare - 1 = p.ComplexityOfCare
         OR @ComplexityOfCare + 1 = p.ComplexityOfCare
         OR @ComplexityOfCare = p.ComplexityOfCare)
      AND (@Size - 1 = p.Size
         OR @Size + 1 = p.Size
         OR @Size = p.Size)
      AND (@Temperature - 1 = p.Temperature
         OR @Temperature + 1 = p.Temperature
         OR @Temperature = p.Temperature)
      AND (@Lighting - 1 = p.Lighting
         OR @Lighting + 1 = p.Lighting
         OR @Lighting = p.Lighting)
      AND (@ColorSaturation - 1 = p.ColorSaturation
         OR @ColorSaturation + 1 = p.ColorSaturation
         OR @ColorSaturation = p.ColorSaturation)
         ORDER BY
```

```
    CASE WHEN @WateringFrequency = p.WateringFrequency THEN 1 ELSE 0 END +
    CASE WHEN @ComplexityOfCare = p.ComplexityOfCare THEN 1 ELSE 0 END +
    CASE WHEN @Size = p.Size THEN 1 ELSE 0 END +
    CASE WHEN @Temperature = p.Temperature THEN 1 ELSE 0 END +
    CASE WHEN @Lighting = p.Lighting THEN 1 ELSE 0 END +
    CASE WHEN @ColorSaturation = p.ColorSaturation THEN 1 ELSE 0 END DESC;
END;
USE [PlantHelper]
GO
/****** Object:  StoredProcedure [dbo].[AddPlantToUserPlants]    Script Date: 09.11.2024 10:39:24 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[AddPlantToUserPlants]
    @UserId INT,
    @PlantId INT
AS
BEGIN
    INSERT INTO PlantUserRelations (UserId, PlantId)
    VALUES (@UserId, @PlantId);
    DECLARE @Id INT;
    SELECT TOP 1 @Id = PlantUserRelationsId
    FROM PlantUserRelations
    ORDER BY PlantUserRelationsId DESC;
        INSERT INTO UserPlantsState
    VALUES (@Id, CAST(GETDATE() AS DATE), CAST(GETDATE() AS DATE), CAST(GETDATE() AS DATE),
        CAST(GETDATE() AS DATE), CAST(GETDATE() AS DATE), CAST(GETDATE() AS DATE));
END;
```

Додаток В (обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

«ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ НАДАННЯ РЕКОМЕНДАЦІЙ ДЛЯ ДОГЛЯДУ ЗА КІМНАТНИМИ РОСЛИНАМИ»

Виконав: студент 2-го курсу, групи 2КН-23м,
спеціальності 122 – «Комп'ютерні науки»

Бортник А. О.
(прізвище та ініціали)

Керівник: к.т.н., доцент каф. КН

Крилик Л. В.
(прізвище та ініціали)

«05» 12 2024 р.

Вінниця ВНТУ - 2024 рік

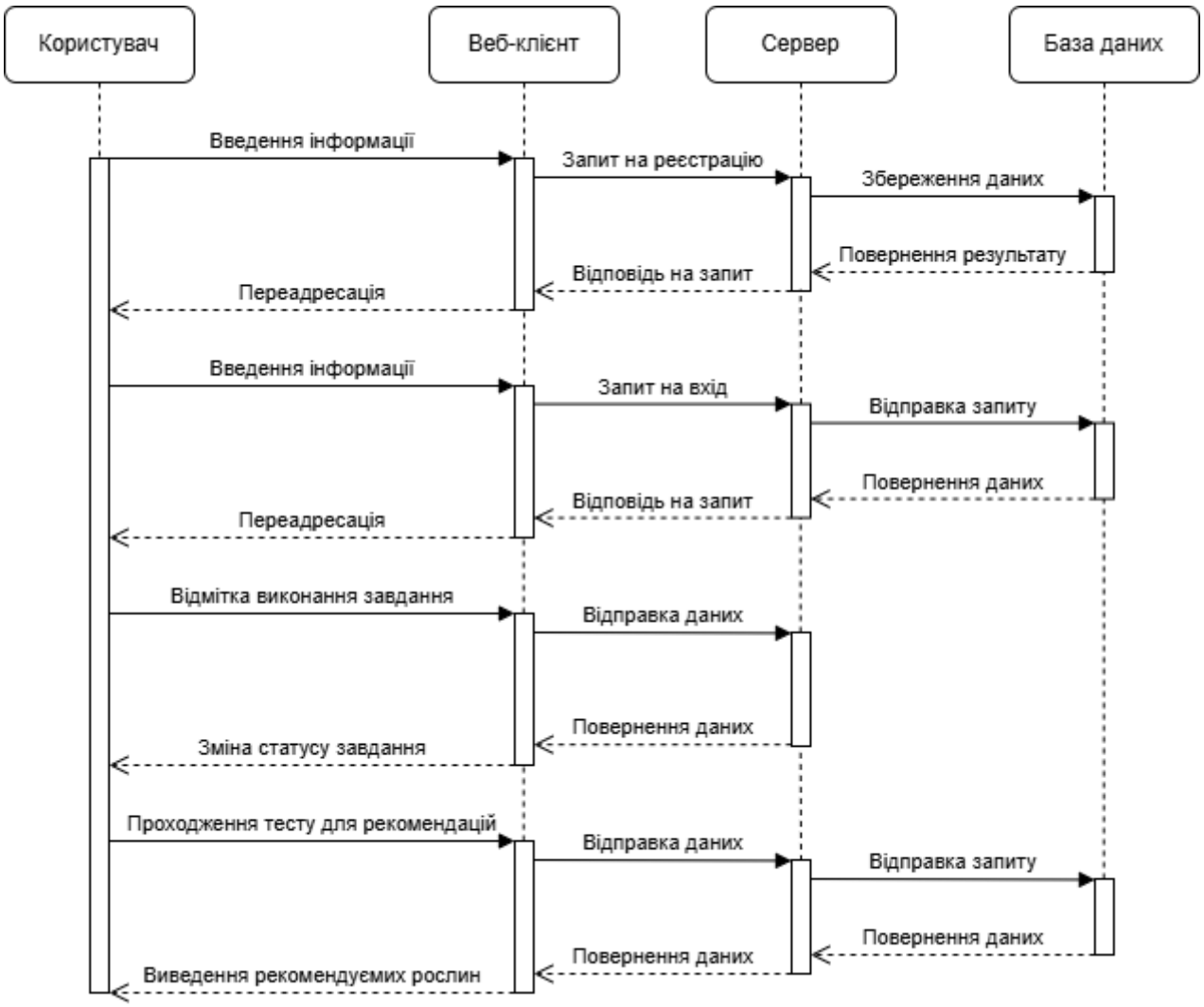

Рисунок В.1 – UML-діаграма послідовності взаємодії модулів інформаційної технології надання рекомендацій для догляду за кімнатними рослинами

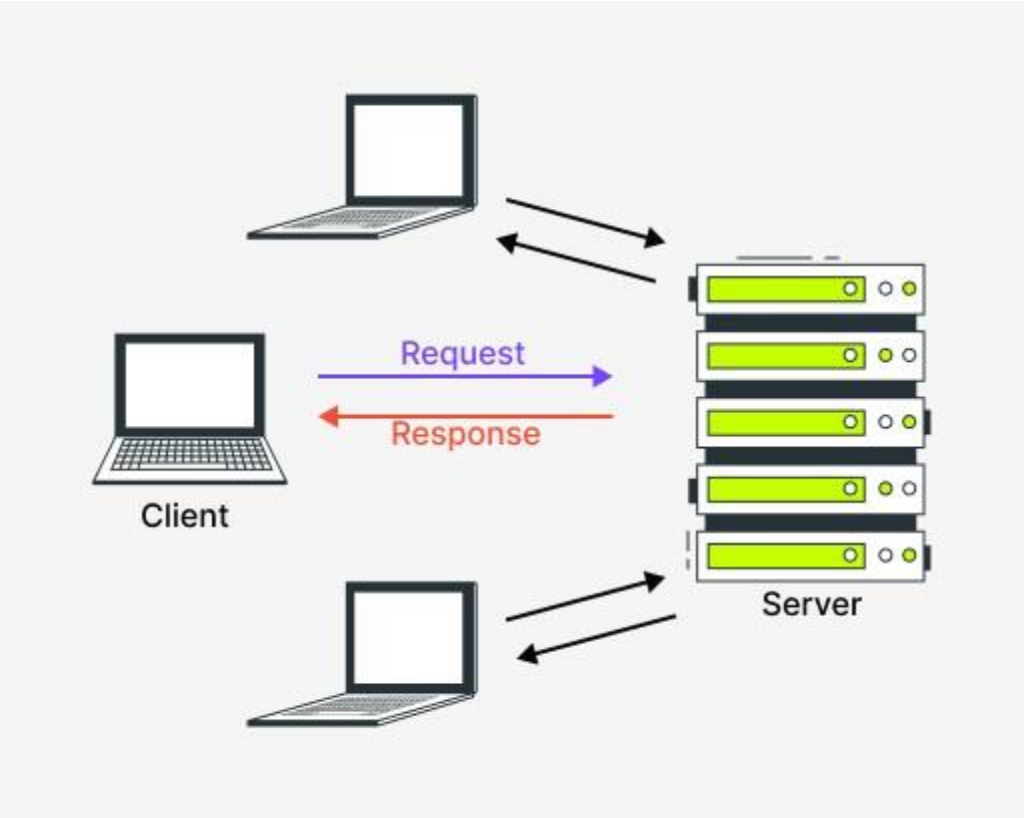

Рисунок В.2 − Клієнт-серверна архітектура

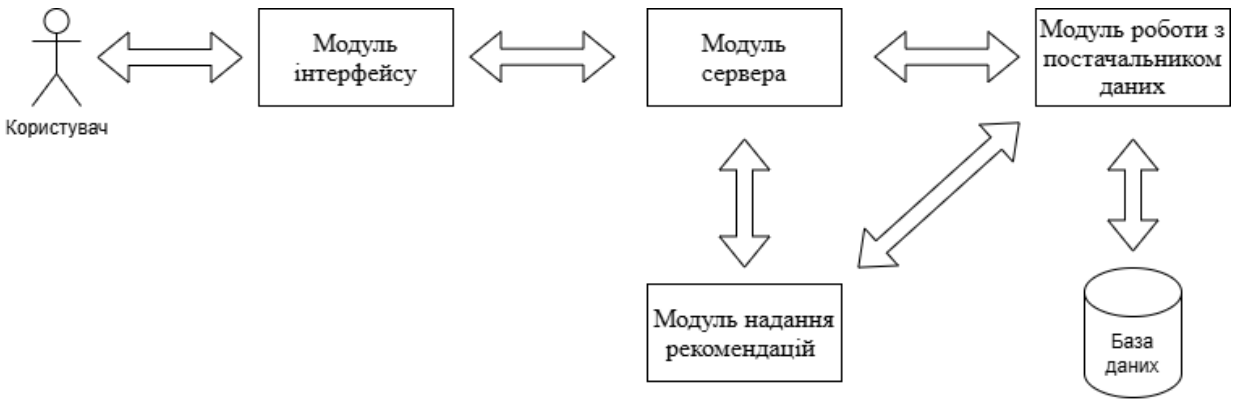

Рисунок В.3 – Загальна структурна модель програмного модуля

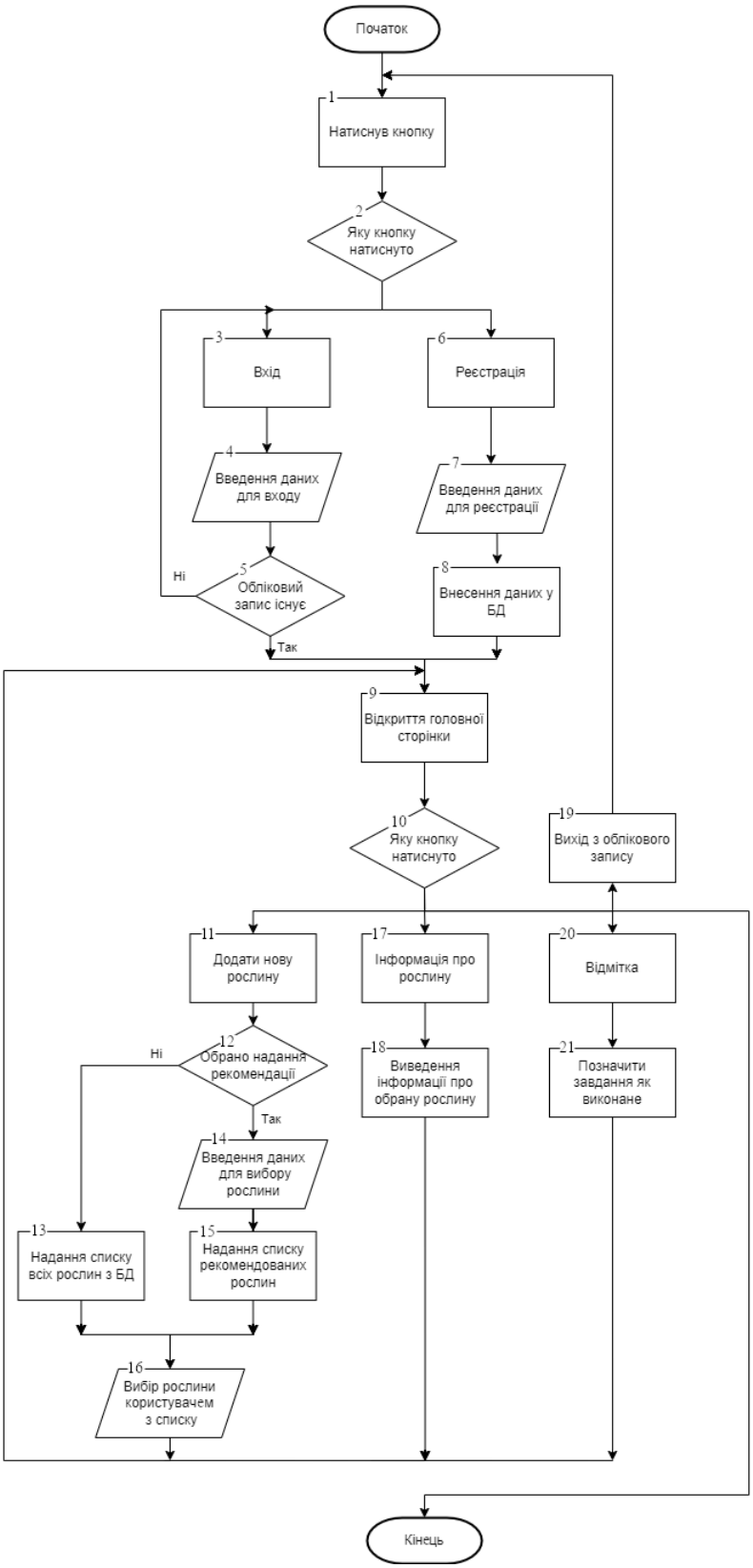

Рисунок В.4 – Схема алгоритму роботи інформаційної технології надання рекомендацій для догляду за кімнатними рослинами

Рисунок В.5 – Схема алгоритму роботи системи надання рекомендацій по вибору кімнатної рослини

Рисунок В.6 – Діаграма діяльності серверу під час запиту інформації про рослину

Рисунок В.7 – Діаграма діяльності серверу під час перевірки чи існує користувач у системі

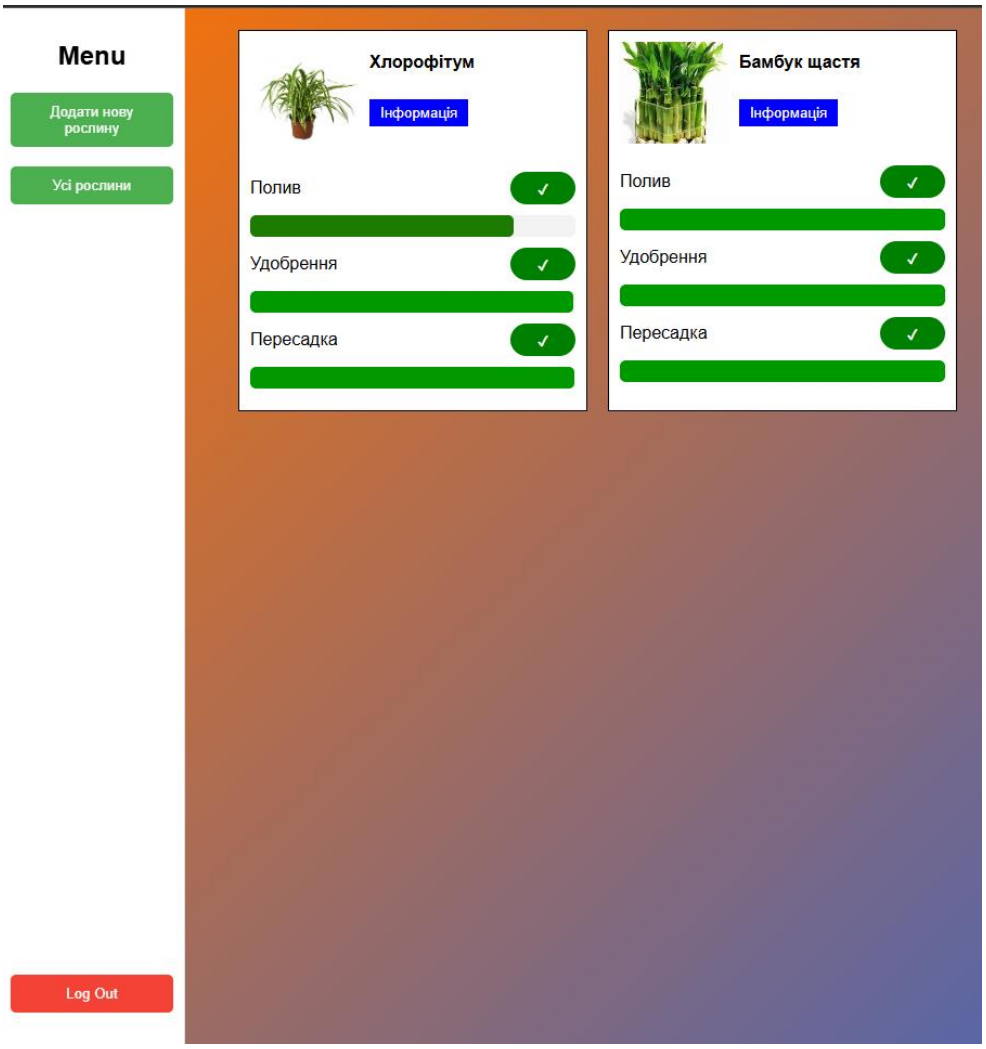

Рисунок В.8 – Загальний вигляд інтерфейсного вікна роботи програми

Додаток Г (довідниковий)

Інструкція користувача

Для того, щоб почати роботу з інформаційною технологією, потрібно мати доступ до інтернет браузера. Головною умовою є стабільне підключення до мобільної мережі інтернету або Wifi (рис. Г.1).

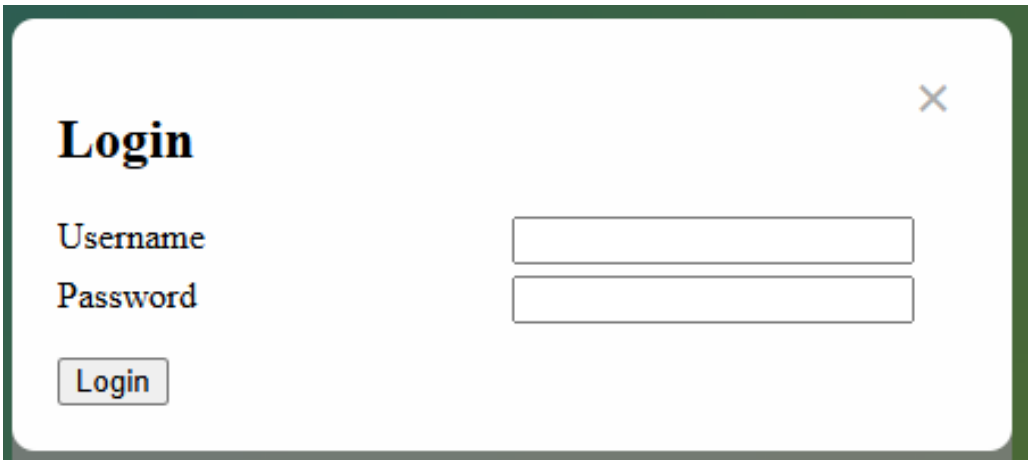

Рисунок Г.1 – Загальний вигляд інтерфейсного вікна авторизації користувача

Після успішної авторизації користувач переадресовується на вікно головного меню програми. Це вікно містить всю необхідну користувачу інформацію. У головному вікні можна побачити рослини та прогрес по догляду рослини. Кожну рослину представлено у вигляді блока на якому і зображена основна інформація. Для помітки завдання як виконаного, користувач має натиснути зелену кнопку, також натиснувши на кнопку «Інформація» можна отримати більш детальну інформацію про рослину. На рисунку Г.2 зображено вікно блоку рослини.

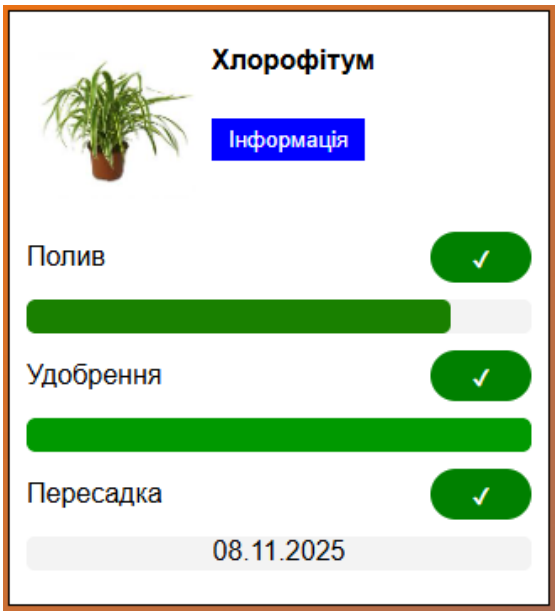

Рисунок Г.2 – Загальний вигляд блоку «Рослини»

Також у лівій частині зображено можливості користувача такі як додати нову рослину, вийти з акаунта або переглянути всі рослини (рис. Г.3).

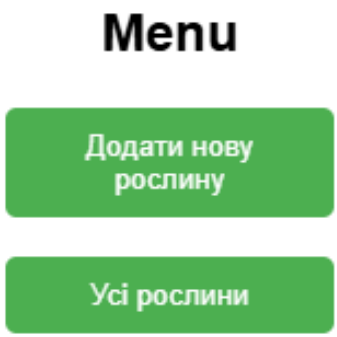

Рисунок Г.3 – Загальний вигляд блоку «Меню»

При натисненні на кнопку «Додати нову рослину» відкривається вікно з питаннями на які користувач має дати відповіді, після чого він отримає результати у вигляді списка рослин, які система надання рекомендацій вважає найбільш прийнятними для користувача. На вікні результату користувач може додати рослину до себе у колекцію та подивитись більш детальну інформацію про неї. Вікно «Питання» зображено на рисунку Г.4, вікно результату зображено на рисунку Г.5.

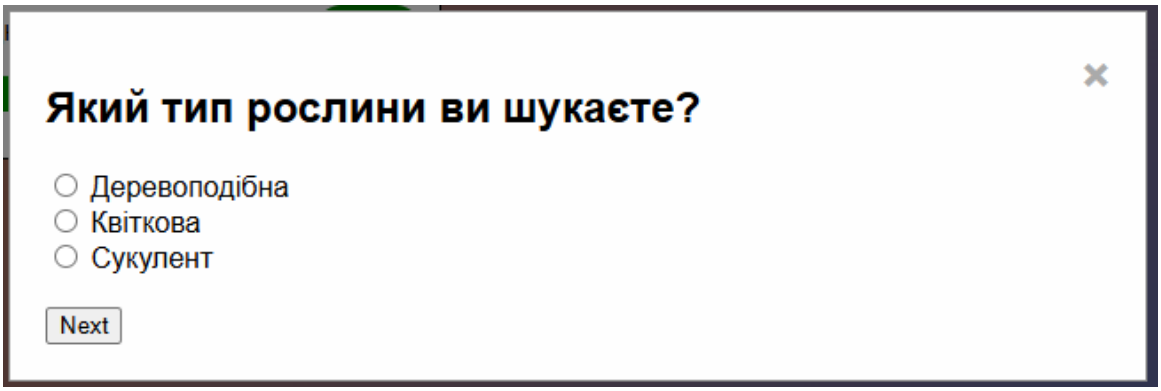

Рисунок Г.4 – Загальний вигляд вікна «Питання»

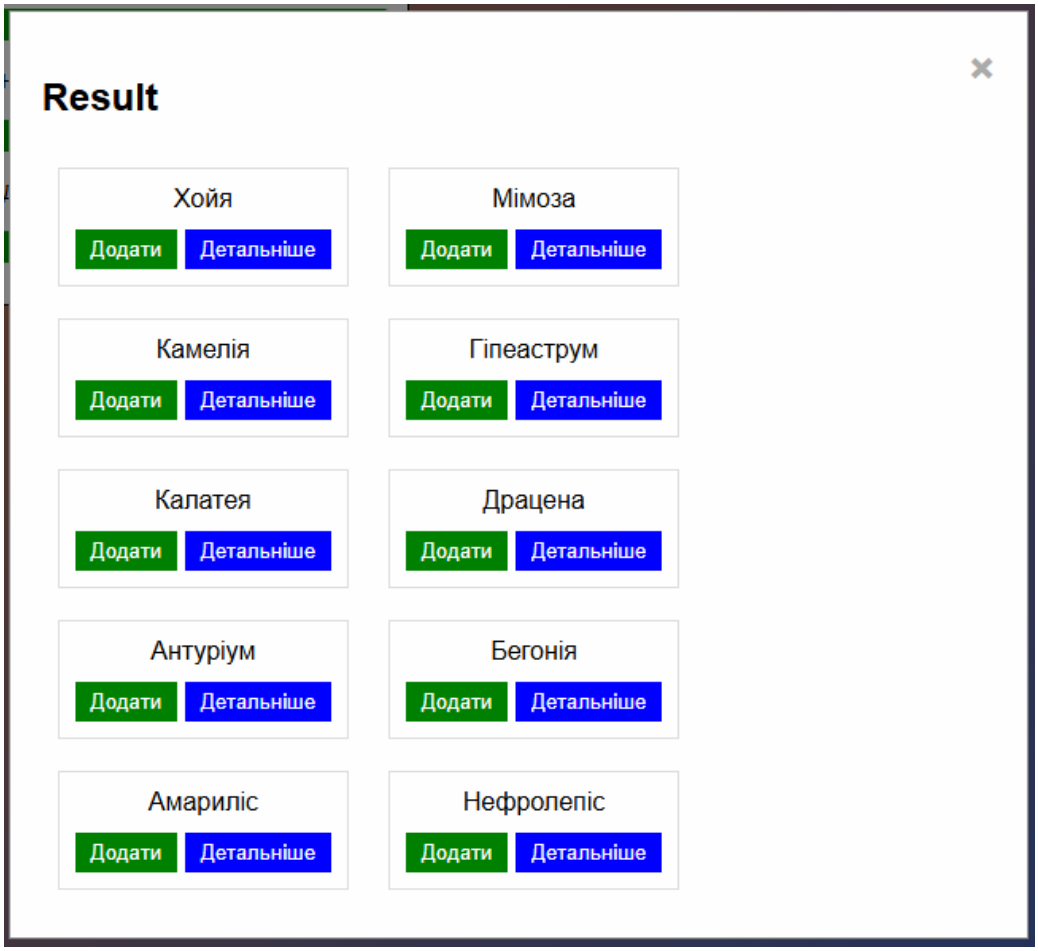

Рисунок Г.5 – Загальний вигляд вікна результату