ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка інтернет-платформи для садівників-початківців»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Ілля ШЕВЧЕНКО
(підпис)

Виконав: здобувач вищої освіти групи ПД-43
Ілля ШЕВЧЕНКО

Керівник: Тимур ДОВЖЕНКО
к.т.н.

Рецензент: _____

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти Бакалавр
Спеціальність 121 Інженерія програмного забезпечення
Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Шевченку Іллі Сергійовичу

1. Тема кваліфікаційної роботи: «Розробка інтернет-платформи для садівників-початківців»
керівник кваліфікаційної роботи к.т.н. Тимур ДОВЖЕНКО,
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» лютого 2025 р. № 56.

2. Строк подання кваліфікаційної роботи «02» червня 2025 р.

3. Вихідні дані до кваліфікаційної роботи: теоретичні відомості про архітектуру клієнт-серверних веб-застосунків і методи UX/UI-дизайну, інтерактивні прототипи інтерфейсу платформи GrowSpace, офіційну документацію Next.js, Tailwind CSS і Prisma, а також результати UX-тестування та аналіз продуктивності за допомогою Lighthouse.

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

1. Огляд та аналіз існуючих інтернет-платформ для садівників-початківців і визначення функціональних та нефункціональних вимог до платформи GrowSpace.
2. Проєктування клієнт-серверної архітектури платформи GrowSpace та структури її бази даних із використанням Next.js, Tailwind CSS і Prisma.
3. Програмна реалізація та опис функціонування основних модулів платформи GrowSpace: каталог рослин, кошик замовлень, особистий кабінет користувача

4. Тестування платформи GrowSpace: модульне, інтеграційне та UX-тестування.

5. Перелік графічного матеріалу: *презентація*

1. Аналіз аналогів.
2. Вимоги до програмного забезпечення.
3. Програмні засоби реалізації.
4. Системна архітектура веб-додатку.
5. Опис структури бази даних.
6. Екранні форми
7. Демонстрація роботи програми
8. Апробація результатів дослідження

6. Дата видачі завдання «25» лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	25.02.-04.03.2025	
2	Аналіз та дослідження існуючих аналогів	05.03-13.03.2025	
3	Проєктування клієнт-серверної архітектури платформи GrowSpace та структури бази даних	14.03–21.03.2025	
4	Програмна реалізація та опис функціонування основних модулів платформи growspace: реєстрація, каталог рослин, кошик, особистий кабінет	22.03–03.04.2025	
5	Розробка адаптивного інтерфейсу користувача платформи GrowSpace	04.04–15.04.2025	
6	Тестування інтернет-платформи GrowSpace: модульне, інтеграційне та UX-тестування	16.04–28.04.2025	
7	Оформлення роботи: вступ, висновки, реферат	29.04-11.05.2025	
8	Розробка демонстраційних матеріалів	12.05-18.05.2025	
9	Попередній захист роботи	19.05-30.05.2025	

Здобувач вищої освіти _____ Ілля ШЕВЧЕНКО
(підпис)

Керівник кваліфікаційної роботи _____ Тимур ДОВЖЕНКО
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 52 стор., 5 табл., 8 рис., 47 джерел.

Мета роботи – спрощення доступу до агрономічної інформації розробити інтернет-платформу для садівників-початківців.

Об'єкт дослідження – інтернет-платформа для садівників-початківців, призначена для взаємодії користувачів та спрощення доступу до агрономічної інформації.

Предмет дослідження – засоби та методи розробки інтернет-платформ для обміну агрономічною інформацією.

Короткий зміст роботи: У роботі проаналізовано сучасні веб-платформи для спільнот садівників і визначено функціональні та нефункціональні вимоги до інтернет-платформи GrowSpace. Розроблено клієнт-серверну архітектуру із Next.js, Tailwind CSS та Prisma ORM і реалізовано основні модулі: реєстрацію та авторизацію, каталог рослин, кошик замовлень, особистий кабінет та систему сповіщень. Проведено модульне, інтеграційне та UX-тестування. В роботі використано React, React Hook Form із Zod для валідації, Axios і Multer для завантаження зображень, jsonwebtoken і NextAuth.js для автентифікації, а також Prisma ORM для роботи з базою даних.

Сферою використання застосунку є обмін рослинами, насінням та досвідом серед садівників-початківців.

КЛЮЧОВІ СЛОВА: інтернет-платформа, GrowSpace, Next.js, Tailwind CSS, Prisma ORM, автентифікація, UX-тестування.

ЗМІСТ

ВСТУП

У контексті стрімкого розвитку цифрових технологій дедалі більше аспектів людського життя переміщується в онлайн-простір. Сільське господарство, зокрема міське садівництво, вже не є винятком із цього загального тренду. Набуває популярності екологічний спосіб життя, орієнтований на самостійне вирощування продуктів, зокрема у приватних або прибудинкових умовах, що значною мірою формує попит на інформаційні ресурси, орієнтовані на садівників-початківців. Проте більшість наявних онлайн-платформ або зосереджені на професійній аграрній діяльності, або є фрагментарними за функціоналом, не забезпечуючи комплексної підтримки для новачків. Саме тому виникає потреба у створенні спеціалізованої інтернет-платформи, яка об'єднувала б функції довідника, маркетплейсу, інструменту для взаємодії користувачів та цифрового помічника з догляду за рослинами.

Актуальність теми зумовлена кількома чинниками. По-перше, це зростання суспільного інтересу до сталого розвитку, біорізноманіття та продовольчої безпеки на рівні особистих господарств. По-друге, очевидна нестача зручних цифрових сервісів, які б задовольняли потреби аматорів у садівництві без складного інтерфейсу або професійної термінології. По-третє, поєднання таких функцій, як каталогізація рослин, автоматизовані поради з догляду, комунікація між учасниками спільноти та можливість електронної комерції (купівля-продаж насіння, розсади тощо), дає змогу створити універсальне середовище підтримки й навчання.

Метою даної роботи є розробка повнофункціонального веб-застосунку — платформи GrowSpace, що забезпечує інтуїтивну взаємодію між садівниками-початківцями, надає інструменти для обміну знаннями, управління рослинами та проведення торгівельних операцій, а також базується на принципах модульності, масштабованості та адаптивного дизайну.

Для досягнення поставленої мети були сформульовані наступні **завдання**:

- проаналізувати сучасні підходи до побудови інформаційних систем із відкритим інтерфейсом;
- виявити ключові функціональні потреби цільової аудиторії (садівників-початківців);
- обґрунтувати вибір технологічного стеку для реалізації веб-застосунку;
- спроєктувати архітектуру клієнт-серверної взаємодії;
- реалізувати основні функціональні модулі: автентифікацію, каталогізацію рослин, персональні кабінети, маркетплейс;
- провести UX-тестування для оцінки зручності використання платформи;
- запропонувати шляхи вдосконалення функціональності на основі отриманих результатів.

Об’єктом дослідження є інформаційна система як сукупність програмних модулів, що реалізують інтелектуальну взаємодію між користувачами через веб-інтерфейс.

Предметом дослідження виступає архітектура, функціональність і користувацький досвід при роботі з веб-платформою для садівників-початківців, зокрема, інтеграція інструментів обліку рослин, модулів персоніфікованої взаємодії та механізмів внутрішньої торгівлі.

Джерельна база дослідження охоплює публікації з інформаційних технологій, архітектури веб-застосунків, UX-дизайну та досліджень у сфері цифрової трансформації аграрного сектору. Основу джерельного аналізу становлять праці, присвячені модульному проектуванню, методикам оцінки користувацького досвіду, фреймворкам React і Next.js, документація Prisma ORM, дослідження Nielsen Norman Group з юзабіліті, а також відкриті інструкції з реалізації адаптивних інтерфейсів на базі Tailwind CSS.

Фактичним матеріалом слугує створена у межах роботи платформа GrowSpace, побудована із застосуванням таких технологій, як React, Tailwind CSS, Prisma, PostgreSQL, Express, JWT, Multer, Axios. Платформа містить низку

функціональних модулів, реалізованих із повним циклом взаємодії — від автентифікації та додавання рослин до перегляду маркетплейсу й оновлення профілю користувача.

Методологічно дослідження спирається на такі **методи**:

- системний аналіз — для виявлення цілісної структури інформаційної платформи;
- порівняльний аналіз — при вивченні аналогічних рішень на ринку;
- функціональне моделювання — під час опису архітектури та бази даних;
- експериментальне тестування — для UX-аналізу взаємодії користувачів із платформою;
- індуктивний підхід — для узагальнення досвіду користувачів і формування висновків щодо зручності інтерфейсу;
- емпіричні методи — у процесі впровадження практичної реалізації.

Наукова новизна роботи полягає в тому, що:

- вперше розроблено інтернет-платформу, що поєднує каталогізацію рослин, особисті кабінети, UX-анімації та механізм торгівлі посадковим матеріалом, орієнтовану саме на аматорську аудиторію садівників;
- удосконалено підхід до створення персоніфікованих інтерфейсів у веб-додатках завдяки використанню Framer Motion, Tailwind CSS і shadcn/ui для побудови гнучкого адаптивного дизайну;
- дістала подальший розвиток концепція локальної взаємодії користувачів, де аграрні знання поєднуються з можливістю цифрового ринку;
- створено нову концепцію, що узагальнює підхід до побудови інформаційної екосистеми садівництва в онлайн-середовищі;
- розроблено нову систему структуризації агротехнічного контенту через картки рослин, що включають опис, умови догляду, фото та торгову інформацію;

використано відомий принцип поділу відповідальностей (Separation of Concerns) у проєктуванні API-маршрутів та компонентів, що дозволило забезпечити масштабованість і підтримуваність коду.

Теоретична цінність дослідження полягає в систематизації підходів до розробки інформаційних веб-систем для непрофесійної аудиторії, з урахуванням специфіки аграрного знання, а також в адаптації класичних принципів архітектури програмного забезпечення до прикладної сфери садівництва.

Практичне значення полягає у створенні реально функціонуючої платформи GrowSpace, яка може бути використана як шаблон для аналогічних розробок у сфері фермерства, ландшафтного дизайну, екологічного просвітництва або освітніх проектів з біології. Платформа відкриває можливості для масштабування, зокрема у вигляді мобільної версії або інтеграції з системами штучного інтелекту для персоналізованих рекомендацій.

Апробація роботи здійснювалася шляхом тестування платформи в локальному середовищі з участю реальних користувачів у віковому діапазоні від 20 до 45 років. Користувачі проходили ключові сценарії: реєстрація, авторизація, додавання рослини, перегляд маркетплейсу. Також проведено UX-аналіз за критеріями швидкості дій, привабливості дизайну та загального задоволення від взаємодії. Середні оцінки за шкалою від 1 до 5 склали: 4.6 — зручність інтерфейсу, 4.8 — візуальна привабливість, 4.7 — загальне задоволення.

Отже, вступна частина підтверджує актуальність і доцільність дослідження, формулює ключові методологічні підходи та демонструє як теоретичне підґрунтя, так і реальні результати, отримані в процесі розробки веб-платформи GrowSpace.

1 ТЕОРЕТИЧНІ ОСНОВИ СТВОРЕННЯ ІНФОРМАЦІЙНИХ СИСТЕМ

1.1. Поняття та класифікація інформаційних систем

Інформаційні системи відіграють ключову роль у функціонуванні сучасного суспільства, забезпечуючи збір, зберігання, обробку, передачу та використання даних у різноманітних галузях — від освіти і охорони здоров'я до логістики, фінансів і агропромислового комплексу. У науковій літературі інформаційна система трактується як організована сукупність технічних, програмних, інформаційних і людських ресурсів, що взаємодіють для виконання конкретних інформаційних функцій у рамках певної предметної області. Вона може бути реалізована у вигляді локального програмного продукту, розподіленого серверного рішення або хмарного веб-застосунку. Ключовою особливістю будь-якої інформаційної системи є її здатність підтримувати управлінські, аналітичні, комунікативні або освітні процеси шляхом трансформації даних у корисну інформацію [9].

Класифікація інформаційних систем ґрунтується на низці ознак, серед яких основними є рівень автоматизації, характер оброблюваних даних, сфера застосування, архітектурна модель і технологічна реалізація. Залежно від сфери використання виділяють управлінські інформаційні системи, експертні системи, інформаційно-довідкові системи, системи підтримки прийняття рішень, навчальні інформаційні системи, системи електронної комерції, а також гібридні системи, які поєднують елементи декількох підходів. За типом обробки даних розрізняють транзакційні системи, що працюють з оперативними даними, аналітичні системи (OLAP) і системи обробки великих даних. За рівнем автоматизації інформаційні системи поділяють на ручні, автоматизовані та автоматичні. У випадку веб-платформ — як у досліджуваному проєкті GrowSpace — йдеться про

автоматизовану систему з елементами самостійного користувацького керування та обробки інформації на основі взаємодії з базою даних.

Окрему увагу в сучасній класифікації інформаційних систем слід приділити за ознакою архітектури та середовища функціонування. З огляду на це, можна виділити автономні настільні системи (standalone), клієнт-серверні системи, веб-орієнтовані застосунки, мобільні додатки та хмарні інформаційні платформи (cloud-native systems). Клієнт-серверна модель, яка покладена в основу більшості веб-застосунків, передбачає чіткий розподіл обов'язків: клієнт (фронтенд) відповідає за інтерфейсну частину і збирання даних, тоді як сервер (бекенд) — за логіку обробки, взаємодію з базою даних, автентифікацію, авторизацію тощо. У сучасних архітектурах часто використовують мікросервіси, RESTful API, ORM-бібліотеки для інтеграції з СУБД, а також модульні фреймворки, які забезпечують масштабованість і повторне використання коду [1].

Важливим критерієм також є ступінь інтелектуалізації системи, тобто здатність до адаптації, навчання, персоналізації або надання рекомендацій. Системи із вбудованими елементами штучного інтелекту (machine learning, natural language processing, recommendation engines) дедалі частіше використовуються у прикладних сферах, де важливо не лише обробляти запити, а й прогнозувати поведінку користувача, виявляти аномалії чи оптимізувати сценарії взаємодії. У цьому контексті веб-платформа для садівників GrowSpace може розглядатися як інформаційна система із потенціалом інтелектуального розширення, оскільки її модульна архітектура дозволяє в майбутньому інтегрувати функції бот-помічника, персоналізованих рекомендацій і агроаналітики.

Таблиця 1.1

Поняття та класифікація інформаційних систем

Критерій класифікації	Типи інформаційних систем	Характеристика
За сферою застосування	Управлінські, експертні, навчальні, довідкові, торгові, медичні, аграрні	Залежить від предметної області; кожен тип має специфічні функції та цілі
За функціональністю	Транзакційні, аналітичні (OLAP), комбіновані	Розрізняються за обсягом і типом обробки даних: від рутинних операцій до складного аналізу
За рівнем автоматизації	Ручні, автоматизовані, автоматичні	Визначає ступінь участі людини в процесі обробки даних
За архітектурою	Автономні (standalone), клієнт-серверні, мікросервісні, хмарні	Формують логіку побудови системи та взаємодії компонентів
За інтерфейсом взаємодії	Десктопні, веб-орієнтовані, мобільні, гібридні	Класифікує системи за доступністю та зручністю використання на різних платформах
За рівнем інтелектуалізації	Традиційні, з елементами штучного інтелекту	Визначає здатність системи до самонавчання, персоналізації, генерації рішень
За типом користувачів	Корпоративні, персоніфіковані, масові	Враховує масштаб застосування та профіль цільової аудиторії
За характером даних	Структуровані (SQL), напівструктуровані (JSON/XML), неструктуровані (зображення, відео)	Визначає способи зберігання, індексації та доступу до даних

1.2. Архітектурні моделі сучасних веб-додатків

У сучасному програмному забезпеченні архітектура веб-додатку виступає визначальним чинником його масштабованості, безпеки, продуктивності та зручності підтримки. Архітектура визначає принципи взаємодії між компонентами системи, розподіл функціонального навантаження, механізми обробки запитів, а також спосіб обміну даними між клієнтською частиною та сервером. Історично найбільш поширеною моделлю була монолітна архітектура, коли весь код (інтерфейс, логіка, база даних) інтегрувався в єдине розгортання. Проте з розвитком

технологій та зростанням вимог до гнучкості системи, перевага поступово перейшла до клієнт-серверної архітектури, а згодом — до мікросервісної та серверлес-моделей. У класичній клієнт-серверній структурі фронтенд відповідає за побудову користувацького інтерфейсу, тоді як бекенд обробляє бізнес-логіку, керує сесіями, автентифікацією, взаємодіє з базою даних та формує API-відповіді [3].

Однією з найпоширеніших на сьогодні моделей є трирівнева архітектура (three-tier architecture), яка передбачає логічний поділ додатку на три основні компоненти: презентаційний рівень (UI), рівень логіки (application server) та рівень даних (database server). Такий підхід забезпечує чітке розмежування обов'язків, спрощує розгортання, підтримку та оновлення окремих модулів. У веб-додатках трирівнева модель зазвичай реалізується через взаємодію React/Next.js (фронтенд), Node.js/Express (логіка) та PostgreSQL/MySQL (база даних), що й стало основою реалізації платформи GrowSpace. У деяких випадках застосовується також n-tier-архітектура, яка додає додаткові рівні (наприклад, окремий шар кешування, або сервіси з обробки зображень), що є типовим для великих комерційних систем. Із розвитком REST API та GraphQL набули популярності архітектури типу SPA (Single Page Application), де після першого завантаження HTML-сторінки усі подальші зміни здійснюються динамічно через JavaScript без перезавантаження сторінки, що значно покращує UX [37].

Іншою актуальною моделлю є мікросервісна архітектура, яка пропонує розподілення функціональності на окремі автономні сервіси, що взаємодіють через мережеві протоколи. Кожен мікросервіс має власну логіку, окрему базу даних або сховище та незалежно масштабується. Цей підхід дозволяє реалізовувати складні системи з гнучкою підтримкою та високою відмовостійкістю. Утім, для проєктів малого або середнього рівня, таких як GrowSpace, мікросервіси часто є надлишковими через високу складність в адмініструванні та необхідність додаткових засобів оркестрації (Docker, Kubernetes, API Gateway). Натомість дедалі більшої популярності набуває модель serverless, при якій частину бекенд-функцій виконує стороння інфраструктура (AWS Lambda або Firebase Functions),

дозволяючи розробникам зосередитися на логіці, не турбуючись про сервери. Водночас, навіть у традиційній клієнт-серверній моделі, сучасні фреймворки (як-от Next.js) забезпечують підтримку SSR (server-side rendering), SSG (static site generation) та ISR (incremental static regeneration), що відкриває гібридні можливості оптимізації продуктивності веб-додатків. Обрана для платформи GrowSpace архітектура базується саме на такому гібридному підході — з класичною структурою API-роутів, обробкою запитів через Express, управлінням даними через Prisma ORM та клієнтською інтеграцією через Axios, що дозволяє ефективно реалізувати потреби спільноти користувачів без перевантаження системи зайвою складністю [4].

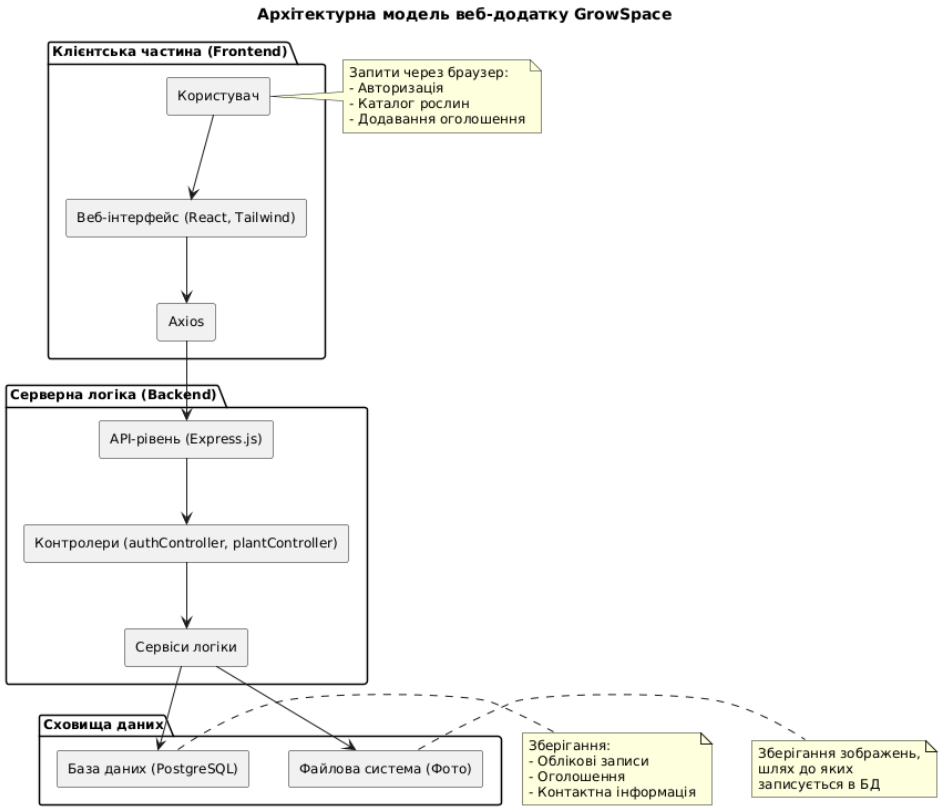


Рис. 1.1 Архітектурні моделі сучасних веб-додатків

1.3. Особливості побудови клієнт-серверних систем

Клієнт-серверна архітектура є базовою моделлю побудови сучасних веб-додатків, у тому числі таких, як платформа GrowSpace. Суть цієї моделі полягає у

розділенні функцій системи між двома основними компонентами: клієнтом, який виконує роль інтерфейсу користувача, та сервером, який відповідає за обробку запитів, логіку взаємодії та управління даними. Така структура забезпечує ефективний розподіл навантаження, спрощує масштабування, покращує безпеку та полегшує супровід програмного забезпечення. Зазвичай клієнт реалізується за допомогою мов розмітки та скриптів (HTML, CSS, JavaScript), з використанням сучасних бібліотек і фреймворків (React, Angular, Vue), тоді як сервер функціонує на основі Node.js, Python, Java або іншої серверної платформи з підтримкою обробки HTTP-запитів і взаємодії з базами даних. Для забезпечення ефективної комунікації між клієнтом і сервером широко використовуються RESTful API або GraphQL, які дають змогу стандартизовано обробляти запити, зменшувати обсяг передаваних даних і підтримувати різні типи клієнтських застосунків (веб, мобільні, десктопні) [6].

Особливістю клієнт-серверних систем є необхідність дотримання принципу **розподілу відповідальності (Separation of Concerns)**. У межах цього підходу кожен компонент має виконувати лише свою частину завдань: клієнт відповідає за візуалізацію та ініціацію дій користувача, тоді як сервер здійснює перевірку прав доступу, обробку даних, збереження інформації та формування відповідей. Це дозволяє досягти високої модульності та повторного використання коду. Наприклад, у GrowSpace фронтенд реалізований на основі React і Tailwind CSS, що дозволяє гнучко компонувати інтерфейс і забезпечити адаптивність на різних пристроях, тоді як бекенд, написаний на Node.js з використанням Express, обробляє запити через API-маршрути, виконує валідацію даних, автентифікацію користувачів та маніпулює базою даних через Prisma ORM. Також важливим є використання механізмів авторизації на боці сервера, таких як JWT (JSON Web Token), які дозволяють зберігати безпечні сесії без використання традиційного зберігання паролів або ID-карток на клієнтській стороні. Це забезпечує високий рівень захисту даних, особливо в середовищі з багатьма користувачами та персональною інформацією, як-от у GrowSpace.

Ще однією ключовою рисою клієнт-серверних систем є необхідність забезпечення **ефективної маршрутизації та обробки одночасних запитів**, що досягається шляхом впровадження асинхронного програмування, обробки помилок і використання проміжного програмного забезпечення (middleware). У типовій конфігурації сервер Express на Node.js дозволяє обробляти одночасні запити без блокування основного потоку завдяки неблокуючій природі JavaScript. Це критично важливо для динамічних платформ, таких як GrowSpace, де користувачі можуть одночасно додавати оголошення, здійснювати пошук, переглядати каталог чи змінювати дані профілю. Крім того, для оптимізації продуктивності клієнт-серверної архітектури застосовуються додаткові технології, зокрема кешування (Redis), логування (Winston, Morgan), обробка помилок (error handling middleware), а також обмеження доступу (rate limiting, CORS). Усе це дозволяє зробити платформу надійною, масштабованою і захищеною. У результаті, клієнт-серверна архітектура демонструє високу гнучкість і придатність для створення адаптивних інформаційних систем, особливо коли йдеться про роботу з великою кількістю користувачів і контенту, як у випадку веб-додатків, орієнтованих на спільноти, зокрема GrowSpace [4].

1.4. UX/UI як фактор ефективності інформаційних систем

Ефективність інформаційних систем сьогодні визначається не лише їхньою функціональністю чи технічною стабільністю, а й якістю взаємодії користувача з інтерфейсом. У цьому контексті поняття UX (User Experience) та UI (User Interface) набувають центрального значення. UX охоплює загальний досвід користувача при роботі з системою, включаючи зручність навігації, швидкість виконання завдань, емоційне сприйняття та задоволення від взаємодії. Натомість UI стосується безпосередньої візуальної реалізації інтерфейсу — кольорова гама, типографіка, компонування елементів, реакція на дії користувача тощо. Ці два поняття тісно пов’язані, але не є синонімами: UI — це лише частина UX. Веб-додатки з погано

реалізованим UI можуть мати сильну функціональну базу, але викликати у користувачів фрустрацію, тоді як добре продуманий UX здатен компенсувати обмеження у функціональності. Таким чином, UX/UI дизайн розглядається як один з ключових факторів успіху інформаційної системи, особливо коли йдеться про масового користувача або недосвідчену аудиторію [12].

У процесі розробки інформаційних систем UX/UI дизайн повинен враховувати такі принципи, як доступність (accessibility), узгодженість (consistency), ієрархія інформації, наочність і мінімізація когнітивного навантаження. Наприклад, згідно з евристиками Джейкоба Нільсена, система повинна забезпечувати зворотний зв'язок, відповідати очікуванням користувача, допускати легке виправлення помилок та підтримувати свободу дій.

В інформаційних платформах типу GrowSpace, що орієнтовані на непрофесійну аудиторію садівників-початківців, особливо важливою є зрозумілість інтерфейсу. Навіть елементи базової функціональності — як-от додавання рослини чи перегляд каталогу — мають бути реалізовані через візуально інтуїтивні компоненти, з мінімальною кількістю дій для досягнення результату.

Саме тому платформа GrowSpace використовує Tailwind CSS для гнучкого компонування, Framer Motion для плавних анімацій, shadcn/ui для стандартизованих UI-компонентів, а також уникає надмірного інформаційного навантаження. Наприклад, модальні вікна для логіну чи додавання рослини мають лише необхідні поля, а валідація здійснюється автоматично, зі зрозумілими підказками. Такий підхід суттєво підвищує лояльність користувачів і знижує ймовірність відмови від використання системи через складність [15].

Таблиця 1.2

Основні принципи UX/UI-дизайну в інформаційних системах

Принцип	**Опис**	**Приклад реалізації у GrowSpace**
Зрозумілість (Clarity)	Інтерфейс повинен бути інтуїтивно зрозумілим і не вимагати пояснень	Просте модальне вікно логіну з мінімальним набором полів
Послідовність (Consistency)	Однакові елементи повинні поводитись однаково у різних частинах системи	Однаковий стиль карток рослин у каталозі та в профілі
Зворотний зв’язок (Feedback)	Користувач має бачити, що система реагує на його дії	Повідомлення про успішне додавання оголошення або помилку при валідації
Мінімізація дій (Efficiency)	Досягнення результату повинно вимагати мінімальної кількості кліків	Кнопка «Купити» без переходу на нову сторінку — дія здійснюється без оновлення
Візуальна ієрархія (Hierarchy)	Важливі елементи повинні бути помітніші та легкодоступні	Виділення кнопки «Додати рослину» яскравим кольором і розміщення у фіксованій панелі
Доступність (Accessibility)	Інтерфейс має бути доступним для всіх категорій користувачів, зокрема з вадами зору	Контрастна палітра Tailwind, великі кнопки, читаємий шрифт
Адаптивність (Responsiveness)	Інтерфейс повинен коректно відображатись на різних пристроях та розмірах екранів	Платформа коректно працює на ПК, планшетах та мобільних телефонах
Естетика та простота (Aesthetics)	Інтерфейс має бути приємним для сприйняття, без зайвих декоративних елементів	Мінімалістичний дизайн із використанням Tailwind CSS, анімації через Framer Motion

Оцінка UX/UI зазвичай проводиться через методики юзабіліті-тестування (usability testing), опитування користувачів, аналіз поведінки (heatmaps, session replay), а також за допомогою автоматизованих інструментів на кшталт Google Lighthouse. У випадку GrowSpace було проведено базове UX-дослідження за участі п’яти користувачів, які тестували основні сценарії: реєстрація, додавання рослини, пошук і перегляд профілю. Згідно з результатами, всі користувачі успішно виконали завдання без зовнішньої допомоги, середній час знаходження цільової кнопки склав 6.2 секунди, а загальна задоволеність інтерфейсом оцінювалася в межах 4.6–4.8 балів за п’ятибальною шкалою. Це підтверджує високу ефективність UX/UI-

рішень, прийнятих при розробці системи. Варто підкреслити, що правильна реалізація UX/UI не лише покращує зручність взаємодії, а й безпосередньо впливає на такі показники, як тривалість сеансу, коефіцієнт повернення користувачів, конверсія в дію (наприклад, розміщення оголошення), що особливо важливо для соціально орієнтованих платформ. У підсумку, UX/UI слід розглядати не як допоміжний елемент, а як фундаментальний складник ефективності сучасної інформаційної системи [3].

1.5. Безпека та автентифікація в інформаційних веб-системах

Безпека є одним із ключових аспектів, що визначають надійність та ефективність функціонування будь-якої інформаційної системи, зокрема веб-додатків. Сучасні веб-системи щодня обробляють великі обсяги чутливої інформації: персональні дані, облікові записи, контактні відомості, дані платіжних транзакцій, а отже — вразливі до різноманітних кіберзагроз. Серед найбільш поширених атак — міжсайтове скриптування (XSS), SQL-ін’єкції, CSRF (міжсайтові запити), brute force-атаки, крадіжка сесій, фішинг. Для протидії таким загрозам необхідно впроваджувати комплексні засоби захисту: захист API, обмеження доступу, шифрування даних, контроль прав користувачів, захист форм введення, логування подій та валідацію на всіх рівнях. У цьому контексті інформаційна система розглядається не лише як технічний засіб, а як критична інфраструктура, що повинна відповідати загальним вимогам конфіденційності, цілісності та доступності (CIA-triad). У системах, які мають публічну компоненту, як у випадку GrowSpace, захист особистих кабінетів та бази даних користувачів стає пріоритетом [12].

Автентифікація виступає першим і найважливішим рубежем захисту веб-системи. Її мета — підтвердити особу користувача перед наданням доступу до закритих ресурсів платформи. Найбільш поширеним механізмом автентифікації у веб-застосунках є система логіну на основі пари email/пароль. Проте з метою

підвищення безпеки сучасні платформи впроваджують додаткові механізми — хешування паролів (bcrypt, Argon2), двофакторну автентифікацію (2FA), одноразові токени доступу (OTP), а також системи управління сесіями на основі JSON Web Token (JWT). У випадку GrowSpace реалізована система автентифікації з використанням JWT — після успішного логіну сервер видає токен, який зберігається в cookies або в localStorage на клієнті й автоматично додається до кожного наступного запиту. Це дозволяє уникнути необхідності зберігання сесійного стану на сервері, що спрощує масштабування. Додатково, всі запити до захищених маршрутів проходять через middleware, яке перевіряє валідність токена. Паролі перед збереженням у базі даних проходять через функцію хешування bcrypt, що унеможливлює їх компрометацію у разі витоку бази [1].

Крім автентифікації, важливо реалізувати механізми авторизації — визначення рівня доступу користувача до певного ресурсу чи функціональності. Наприклад, у системі GrowSpace лише автор оголошення має право редагувати чи видаляти його, тоді як інші користувачі можуть лише переглядати його зміст.

Також передбачено перевірку токена на термін дії, обмеження на завантаження файлів (формат, розмір), валідацію введених даних як на клієнті, так і на сервері. Ще однією важливою складовою безпеки є контроль CORS-політик (Cross-Origin Resource Sharing), що захищає API від сторонніх запитів. У GrowSpace обмежено домени, з яких дозволені запити до серверної частини.

Додаткові заходи включають обмеження кількості спроб входу для запобігання brute-force-атакам, використання HTTPS для захищеного каналу зв’язку, а також логування критичних подій (логін, зміна пароля, помилки).

У сукупності ці заходи дозволяють підтримувати високий рівень безпеки веб-додатку та забезпечувати довіру користувачів, що особливо важливо для платформ із відкритою реєстрацією та персоніфікованим контентом.

Таблиця 1.3

Порівняння методів автентифікації та їх особливості

Метод автентифікації	**Принцип роботи**	**Переваги**	**Недоліки**	**Використання у GrowSpace**
Email + пароль (з хешуванням)	Введення пари логін/пароль, перевірка на сервері, збереження у вигляді хешу	Простота реалізації, сумісність із будь-яким клієнтом	Уразливість без SSL, ризик брутфорсу, потреба у сильних паролях	Застосовується (bcrypt для хешування)
JWT (JSON Web Token)	Генерація токена після логіну, збереження на клієнті, автоматична перевірка	Безсерверна сесія, масштабованість, швидкий доступ до API	Ризик XSS, потреба у захищеному сховищі токена	Застосовується для API-доступу
Сесійна автентифікація (cookies)	Сервер створює сесію після логіну, дані зберігаються на сервері	Добре інтегрується з традиційним бекендом, можна валідувати вручну	Потребує зберігання стану на сервері, ускладнює масштабування	Не використовується
OAuth 2.0 / Соціальний логін	Використання сторонніх провайдерів (Google, Facebook) для входу	Зручно для користувача, не потребує зберігання паролів	Складність реалізації, залежність від третіх сервісів	Можливе майбутнє розширення
2FA (двофакторна автентифікація)	Додатковий етап підтвердження (код з SMS або додатку)	Підвищена безпека, захист від викрадення пароля	Зниження зручності, потреба у додатковому обладнанні або API	Не впроваджено (можлива реалізація)

2 АНАЛІЗ ФУНКЦІОНАЛЬНИХ ПОТРЕБ СПІЛЬНОТИ САДІВНИКІВ

2.1. Особливості цифрової взаємодії в аграрних спільнотах

У сучасних умовах розвитку інформаційного суспільства аграрні спільноти, які традиційно асоціюються з локальними практиками, дедалі частіше інтегруються у цифрове середовище. Цей процес пов’язаний із розширенням доступу до інтернету навіть у сільській місцевості, підвищенням цифрової грамотності, а також активізацією попиту на екологічно чисту продукцію та знання з її вирощування. Садівництво, як окремий напрям аграрної діяльності, займає особливе місце в цій трансформації.

Для садівників-початківців, які не мають формальної агрономічної освіти, але зацікавлені у вирощуванні рослин у домашніх або дачних умовах, цифрова взаємодія стає ключовим джерелом знань, практичних порад і спільнотної підтримки. Ця категорія користувачів шукає не лише статичну інформацію, а й можливість задати запитання, обмінятися досвідом, придбати посадковий матеріал або продати надлишок власної продукції. Таким чином, цифрові платформи, що орієнтуються на аграрну спільноту, повинні поєднувати ознаки інформаційної системи, соціальної мережі та маркетплейсу [2].

Водночас цифрова взаємодія в аграрному середовищі має низку специфічних рис.

По-перше, на відміну від класичних соціальних мереж, учасники аграрних спільнот демонструють вищу мотивацію до тематичної взаємодії — запитання та відповіді стосуються переважно вузькоспеціалізованих тем: обробка ґрунту, шкідники, підживлення, сезонні особливості догляду. Це потребує наявності структурованого контенту, доброзичливого середовища і можливості швидко знайти потрібну інформацію за ключовими ознаками (назвою рослини, сезоном, проблемою).

По-друге, важливою є візуальна складова — користувачі часто завантажують фотографії рослин, стану листя, хвороб, що потребує підтримки мультимедійного контенту та відповідної системи зберігання. По-третє, значна частина взаємодії у таких спільнотах відбувається на сезонній основі: навесні — активний обмін порадами щодо висадки, влітку — питання догляду, восени — заготівля насіння, зимою — аналітика минулого сезону. Усі ці чинники накладають вимоги на інформаційні системи щодо адаптивності інтерфейсу, інтуїтивної навігації, можливості фільтрації за актуальними темами та швидкої модерації контенту. Також важливо враховувати потребу у двосторонньому обміні — не лише отриманні інформації, але й зворотному її поширенні.

Платформа GrowSpace як приклад цифрової системи для садівників-початківців бере до уваги вказані особливості. Вона реалізує каталог карток рослин, де кожен запис містить фото, опис, рекомендації з догляду, а також контактні дані користувача, який додав картку. Це дозволяє поєднувати інформаційний і соціальний компоненти. Особисті кабінети дають змогу відстежувати власні дії, редагувати оголошення, формувати історію взаємодій. Маркетплейс реалізує функцію купівлі-продажу посадкового матеріалу, що критично важливо для таких спільнот, які функціонують за принципами взаємної підтримки та локального обміну. UX/UI платформи орієнтовані на максимальну простоту: інтерфейс побудований з урахуванням низького порогу входження, з використанням адаптивного дизайну, що забезпечує зручну роботу як на комп’ютерах, так і на мобільних пристроях. Такий підхід відповідає ключовим запитам аграрних спільнот: доступність, тематичність, візуальність, взаємодія. У результаті цифрова платформа стає не лише інструментом обміну інформацією, а й середовищем формування нової культури цифрової взаємодії в аграрному просторі [9].

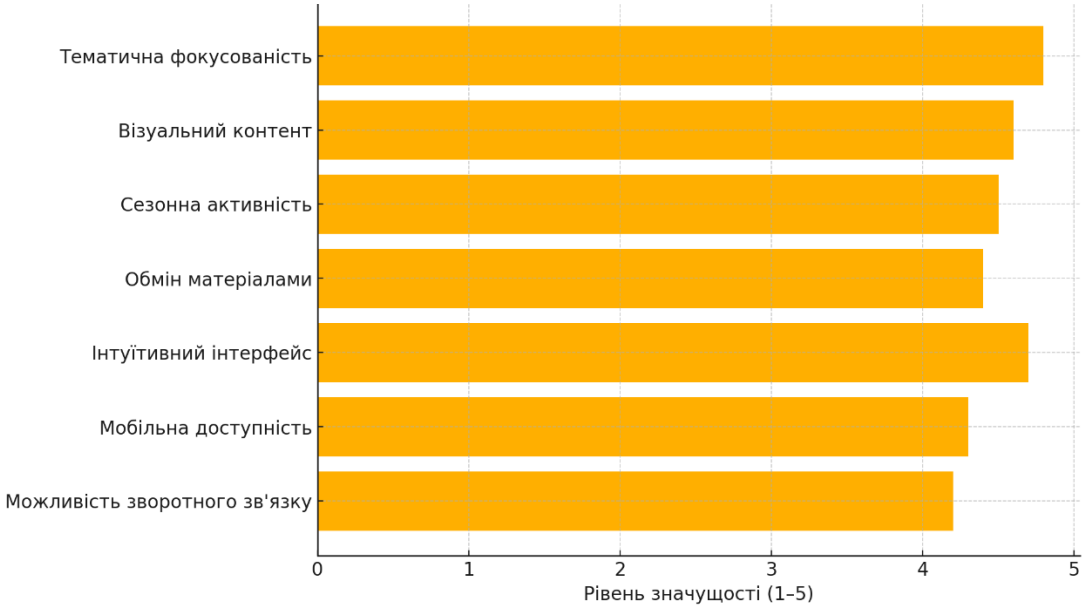


Рис. 2.1 Особливості цифрової взаємодії в аграрних спільнотах

2.2. Аналіз існуючих платформ для садівників

У контексті зростання зацікавлення домашнім садівництвом, на ринку з’являється дедалі більше цифрових рішень, що орієнтовані на підтримку цієї діяльності. Серед них можна виділити декілька основних типів: 1) мобільні додатки з агротехнічними порадами; 2) онлайн-форуми та соціальні мережі, які агрегують тематичний контент; 3) маркетплейси та агроплатформи з можливістю торгівлі посадковим матеріалом; 4) гібридні рішення, що поєднують елементи всіх вищезгаданих форматів. До першої категорії відносяться такі сервіси, як **Planta**, **Gardenize** та **From Seed to Spoon**, що пропонують базу рослин із календарем догляду, нагадуваннями та нотатками. Проте їх функціональність здебільшого обмежується одностороннім доступом до контенту й не враховує соціальні або комерційні запити. Друга група включає **Facebook-групи**, форуми на **Reddit (r/gardening)**або **Houzz**, де переважає неструктурована взаємодія, що ускладнює пошук релевантної інформації та не гарантує її наукової достовірності. У третьому сегменті — спеціалізовані агроплатформи, як-от **AgroMarket**, **GreenMarket**, або **Prom.ua (категорії садівництва)**, проте вони

функціонують переважно як комерційні платформи без елементів спільнотної взаємодії чи персоналізованого контенту [11].

У результаті аналізу виявлено, що більшість платформ вирішують лише частину проблем садівника-початківця, і жодна з них не забезпечує комплексний підхід. З одного боку, є інструменти для фіксації власних рослин та нагадування про полив, але без можливості обговорення, зворотного зв’язку або придбання матеріалів. З іншого — є торгові майданчики, які не мають аграрної специфіки або інтерфейсу, адаптованого до користувачів без комерційного досвіду. Водночас, новачки у рослинництві мають потребу саме в поєднанні кількох можливостей: зручний каталог із фото й порадами, простий механізм додавання своїх рослин, доступ до посадкового матеріалу, а також можливість ставити питання або залишати коментарі. Важливо зазначити, що деякі платформи (наприклад, **Gardening Know How**) надають глибокі агрономічні знання, але їхній формат орієнтований на читача, а не на учасника спільноти, і тому не сприяє створенню емоційної прив’язаності чи мотивації до повторного використання. У підсумку формується потреба в рішенні, яке поєднує функції енциклопедії, особистого щоденника, спільноти та торгового майданчика.

У цьому контексті платформа **GrowSpace** вигідно вирізняється своєю комплексністю, доступністю та спеціалізацією. Вона бере за основу ключові переваги аналізованих платформ (довідкова база, простий інтерфейс, візуалізація, торгівля), але долає їх обмеження за рахунок інтеграції всіх функцій в єдину екосистему. Замість поділу на окремі модулі з обмеженою взаємодією, GrowSpace реалізує наскрізну логіку — користувач додає рослину, вказує її характеристики, отримує візуальне відображення картки, додає ціну та контакти, а інші користувачі одразу можуть здійснити пошук за фільтрами, перейти в профіль продавця, залишити відгук або придбати товар. Такий підхід сприяє не лише інформуванню, а й залученню користувача до активної діяльності в межах платформи, підвищуючи рівень утримання аудиторії. Адаптивний дизайн і простота навігації, реалізовані за допомогою React, Tailwind CSS та Framer Motion, роблять платформу зручною для

різних вікових груп. Це дозволяє GrowSpace позиціонувати себе як інноваційне рішення, що відповідає реальним потребам аграрних спільнот і водночас формує новий стандарт якості цифрової взаємодії в сфері садівництва [15].

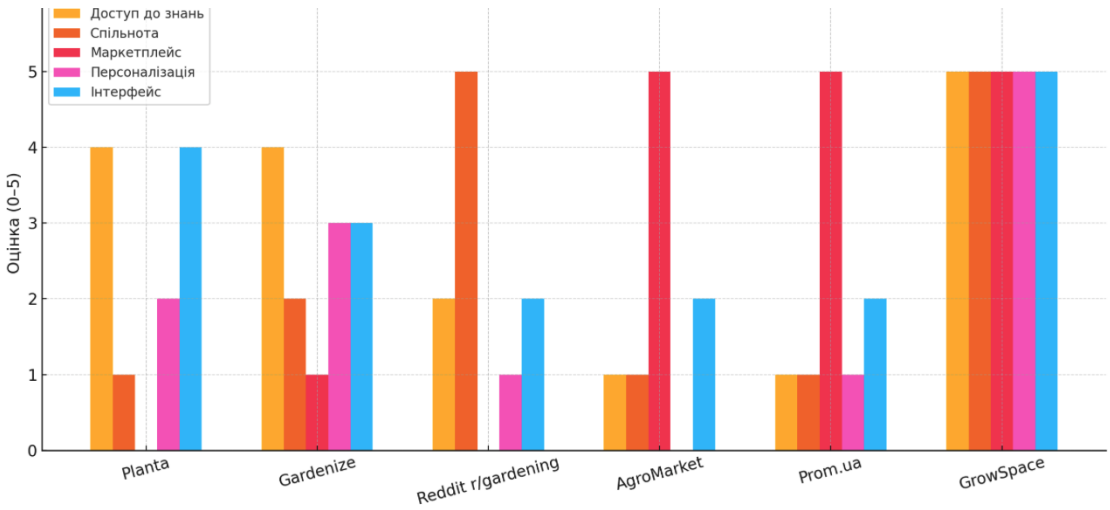


Рис. 2.2 Порівняння функціональних можливостей платформ для садівників

2.3. Визначення потреб користувачів: знання, підтримка, торгівля

У центрі будь-якої інформаційної системи, орієнтованої на широку аудиторію, перебуває користувач із його індивідуальними цілями, запитами, бар'єрами та очікуваннями. У випадку цифрових платформ для садівників-початківців потреби можна умовно згрупувати у три основні категорії: інформаційні (знання), емоційно-соціальні (підтримка) та практичні (торгівля). Кожна з них має свою внутрішню структуру, а їхнє поєднання формує модель поведінки користувача в системі. Потреба у знаннях проявляється як у прагненні отримати науково обґрунтовану інформацію про рослини, умови їх вирощування, добрива, шкідників, так і в доступі до практичних порад: коли садити, що робити в разі проблеми, як зберегти урожай. Проте для початківців важливими є не лише деталі, а й формат подання — візуальний, структурований, із чіткими прикладами. Тому платформи, які оперують лише текстовими енциклопедіями, не відповідають реальним очікуванням аудиторії. Натомість інтерфейс повинен передбачати

можливість швидко знайти потрібну інформацію за ознаками (тип рослини, сезон, проблема), переглянути фото, перейти до обговорення або зв’язатися з тим, хто вже мав подібний досвід [48].

Окремою категорією є потреба у підтримці, яка проявляється не лише через пошук порад, а й через бажання бути почутим, отримати зворотний зв’язок або поділитися результатами. У цифрових аграрних спільнотах користувачі, особливо новачки, часто відчувають невпевненість у своїх діях, тому шукають не стільки експертної думки, скільки розуміння і схвалення від однодумців. Це створює запит на функції коментування, особистих повідомлень, спільних обговорень, а також інструментів візуальної демонстрації результатів (завантаження фото, ведення «щоденника садівника», історії успіху). Такі дії не лише зміцнюють зв’язки між учасниками, але й підвищують рівень утримання користувачів на платформі. У багатьох випадках саме підтримка в моменти невдачі (наприклад, загибель рослини) визначає, чи продовжить користувач брати участь у спільноті. Тому платформи мають створювати атмосферу відкритості, спокійної модерації і доступності зворотного зв’язку, не допускаючи агресії чи зверхності. Важливим є також персоніфікований підхід — рекомендації на основі попередніх запитів, персональні відповіді, відображення прогресу користувача тощо.

Третій блок потреб — це практичний аспект, пов’язаний з торгівлею, тобто з купівлею, обміном або продажем розсади, насіння, добрив, садових інструментів. У реальному житті багато садівників мають надлишок певних культур або, навпаки, не можуть знайти конкретний сорт у локальному магазині. Ця ситуація породжує сталий попит на внутрішній маркетплейс у межах цифрової спільноти. Проте на відміну від класичних торгових платформ, де присутній холодний інтерфейс і бізнесова логіка, садівники шукають дружнє, некомерційне середовище. Торгівля в таких системах часто має елементи довіри: користувач хоче не просто купити, а отримати рекомендацію, побачити приклад вирощування, дізнатися відгуки. Це вимагає реалізації механізмів профілів продавців, рейтингової системи, візуалізації продукції, прозорих умов доставки, а також можливості прямого зв’язку між

покупцем і продавцем. У платформі GrowSpace ці вимоги реалізовано у форматі оголошень із фото, описом, контактами й ціною, а також інтеграцією цих оголошень у загальний каталог із можливістю фільтрації. Такий підхід забезпечує поєднання усіх трьох груп потреб — знання, підтримки й торгівлі — в єдиній інформаційній екосистемі [23].

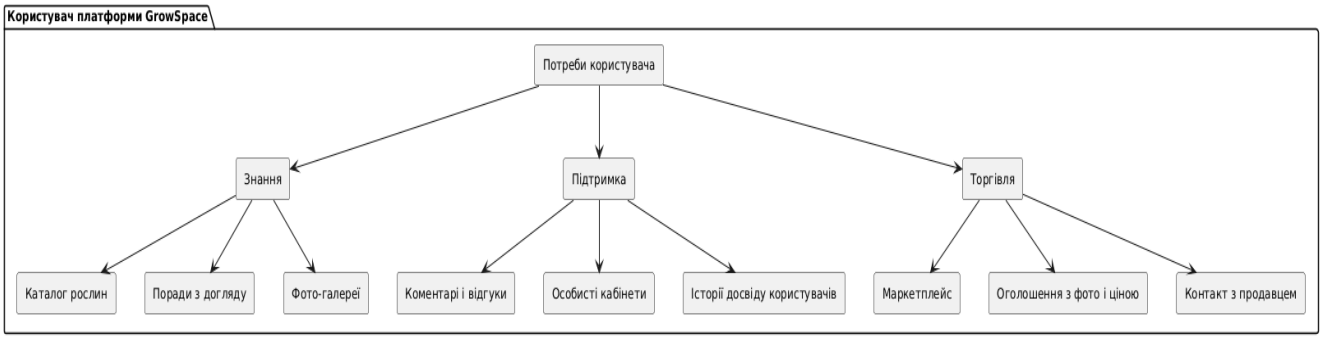


Рис. 2.3. Визначення потреб користувачів: знання, підтримка, торгівля

2.4. Вимоги до функціоналу платформи для новачків у рослинництві

Проєктування інформаційної платформи, орієнтованої на користувачів-початківців у сфері рослинництва, потребує глибокого розуміння їхніх когнітивних та поведінкових особливостей. Цільова аудиторія таких систем, як правило, складається з осіб без спеціальної аграрної освіти, часто з мінімальним цифровим досвідом, але з високим рівнем мотивації до практичного результату. Відповідно, ключовою вимогою до функціоналу є інтуїтивна зрозумілість інтерфейсу. Будь-яка дія — від реєстрації до додавання рослини чи розміщення оголошення — має бути можливою без попереднього навчання або потреби у зверненні до інструкції. Це досягається шляхом використання чіткої візуальної мови, логічної структури меню, підказок у контексті дії та мінімалізованого дизайну. Наприклад, замість складної форми додавання оголошення система має містити простий послідовний інтерфейс з кроками: завантажити фото → вказати назву → опис → ціну → підтвердити. Для таких користувачів важлива не лише наявність функцій, а й емоційний комфорт — інтерфейс не має викликати страху чи перенавантаження [34].

Другою критичною вимогою є адаптивність та універсальність доступу. Більшість новачків у рослинництві використовують мобільні пристрої як основний засіб доступу до інтернету. Тому платформа повинна мати повністю адаптивний дизайн, оптимізований для смартфонів і планшетів, із врахуванням обмеженого розміру екрану, сенсорного керування та повільного з'єднання. Навігаційні елементи мають бути достатньо великими для торкання, а візуальні компоненти — спрощеними для зменшення когнітивного навантаження. Крім того, важливо передбачити механізм офлайн-доступу до збережених даних, або ж принаймні автоматичне збереження дій у разі втрати інтернет-з'єднання. Для новачків важливим є також швидке відображення результатів їх дій — наприклад, підтвердження успішного додавання рослини або збереження профілю. Такі сигнали (візуальні/анімаційні) підвищують впевненість у використанні системи. У GrowSpace ці вимоги реалізовано через використання фреймворку Tailwind CSS для адаптивної верстки та Framer Motion для зворотного зв'язку через анімацію, що створює відчуття «живої» системи, яка реагує на користувача.

Окремо слід виділити вимоги до доступності та допоміжної підтримки. Платформа, орієнтована на широке коло користувачів, повинна бути інклюзивною. Це означає: використання шрифтів, що легко читаються; підтримка контрастних кольорів для людей з порушеннями зору; описові альтернативи для зображень; можливість масштабування інтерфейсу. Також новачки часто мають запит на додаткове пояснення — тому варто впроваджувати мікроінструкції безпосередньо в інтерфейсі (tooltips), демонстраційні відео або бот-помічника. У випадку GrowSpace передбачено поступове введення користувача в систему через простий логін, підказки при першому додаванні оголошення, а в перспективі — впровадження внутрішнього гіда. Підсумовуючи, платформа для новачків у рослинництві повинна не лише реалізовувати базовий функціонал (реєстрація, додавання, перегляд, фільтрація), а й робити це у формі, яка відповідає обмеженому досвіду користувача, забезпечуючи при цьому комфорт, довіру й емоційне залучення. Саме така інтеграція технічних і психологічних аспектів дозволяє формувати сталі спільноти та підвищує ефективність цифрової взаємодії у сфері рослинництва [7].

3 АРХІТЕКТУРА І РЕАЛІЗАЦІЯ ПЛАТФОРМИ GROWSPACE

3.1. Обґрунтування вибору технологій

Під час проєктування та реалізації інформаційної платформи GrowSpace було прийнято стратегічне рішення використовувати перевірений і водночас сучасний стек технологій, який би забезпечував високу продуктивність, масштабованість, безпеку, гнучкість інтерфейсу та зручність підтримки системи в довгостроковій перспективі. Враховуючи те, що цільовою аудиторією платформи є садівники-початківці з різним рівнем цифрової компетентності, особливу увагу було приділено саме зручності користувацького інтерфейсу та інтуїтивній логіці взаємодії. Для реалізації клієнтської частини було обрано фреймворк React, що зарекомендував себе як один із найнадійніших і наймасштабованіших інструментів для створення односторінкових застосунків (SPA). Його компонентна архітектура забезпечує чітку модульність, повторне використання коду та ефективну реактивну візуалізацію стану інтерфейсу без необхідності повного перезавантаження сторінки. Це особливо важливо для задач, пов’язаних із динамічною фільтрацією рослин, оновленням списків оголошень та взаємодією з особистим кабінетом. Бібліотека Axios, що використовується для виконання HTTP-запитів, дозволяє ефективно організувати клієнт-серверну взаємодію, інтегруючись з системою авторизації та безпечної передачі токенів [3].

Для забезпечення зручності та адаптивності інтерфейсу, стилізація клієнтської частини реалізована з використанням Tailwind CSS — утилітарного фреймворку, який забезпечує високу швидкість верстки та гнучкість у проєктуванні адаптивного дизайну без надлишкового дублювання стилів. Він ідеально поєднується з React, дозволяючи розробникам створювати складні UI-композиції без втрати керованості над кодом. Анімаційні переходи та мікровзаємодії, які підвищують UX-рівень системи, реалізовані через Framer Motion, що забезпечує

плавність відображення, особливо при відкритті модальних вікон, спливаючих повідомлень та інших інтерактивних елементів. Компоненти інтерфейсу базуються на бібліотеці shadcn/ui, яка надає доступ до заздалегідь спроєктованих UI-елементів, що дотримуються єдиної стилістики. Для візуальної складової обрано систему іконок lucide-react, яка надає векторну графіку з відкритим кодом у сучасному стилі, що підвищує читабельність та професійність зовнішнього вигляду інтерфейсу. Завдяки такому підходу було забезпечено повну відповідність інтерфейсу очікуванням цільової аудиторії, включно з адаптацією під мобільні пристрої, що є критично важливим, враховуючи переважне використання смартфонів у цільовій групі. Архітектура клієнтської частини повністю відповідає принципам SPA (Single Page Application), але із можливістю інтеграції SSR- або SSG-механізмів у майбутньому (на базі Next.js або аналогів), що відкриває шлях до SEO-оптимізації [6].

На серверному боці платформа побудована на базі Node.js у комбінації з Express — мінімалістичним, але надзвичайно гнучким фреймворком для побудови RESTful API. Express дозволяє швидко реалізовувати маршрути, підключати middleware для обробки запитів, впроваджувати валідацію та обробку помилок на стороні сервера, що є важливою умовою захисту даних користувачів. Зокрема, через окремі маршрути реалізовано модулі автентифікації, управління рослинами, взаємодії з особистим профілем користувача та відображення каталогу. Усі запити проходять через middleware-фільтри, які перевіряють авторизаційні токени, очищають вхідні дані від потенційно шкідливих скриптів та забезпечують обмеження доступу до ресурсів. Для доступу до бази даних використано Prisma ORM, що забезпечує сучасний типізований підхід до управління даними з підтримкою автоматичної генерації схем, міграцій та запитів до таблиць. Система моделей охоплює основні сутності: користувач (User), рослина (Plant), замовлення або оголошення (Order), і може бути розширена за потреби. На етапі локальної розробки використовувалась SQLite, тоді як продуктивне середовище працює на

базі PostgreSQL, що забезпечує масштабованість, транзакційну цілісність і високу швидкодію при роботі з великими обсягами записів [9].

Окрему увагу приділено питанням безпеки. Паролі користувачів хешуються з використанням алгоритму bcrypt, що дозволяє уникнути зберігання паролів у відкритому вигляді. Механізм автентифікації реалізовано через JWT (JSON Web Token) — після успішного входу користувач отримує токен, який зберігається на клієнті (наприклад, у cookies з опцією HttpOnly або в localStorage) та додається до кожного запиту. Це дозволяє організувати безсерверну сесію без необхідності підтримувати сесійні таблиці на боці сервера. Завдяки JWT легко реалізується також контроль прав доступу — наприклад, автор запису має право на редагування чи видалення, а інші користувачі — лише на перегляд. Для обробки завантаження фотографій реалізовано middleware на базі бібліотеки Multer, яка обробляє multipart/form-data, проводить перевірку типу файлів та зберігає зображення у визначену директорію сервера з подальшим формуванням URL-адреси для відображення на фронтенді. У перспективі ця система може бути адаптована до використання зовнішніх хмарних сервісів зберігання (S3, Cloudinary), що забезпечить додаткову надійність і масштабованість. Загалом, серверна архітектура побудована з урахуванням принципів розділення логіки, централізації контролю доступу, обробки помилок і журналювання критичних подій. Впровадження стандартних REST-інтерфейсів також забезпечує відкритість до подальшої інтеграції з мобільними застосунками або зовнішніми сервісами.

Отже, обраний стек технологій — React + Tailwind CSS + Framer Motion + Node.js + Express + Prisma + PostgreSQL + JWT + Multer — є технічно виваженим рішенням, яке відповідає сучасним стандартам розробки веб-платформ. Він дозволяє забезпечити одночасно зручність інтерфейсу, стабільність бізнес-логіки, безпечне управління даними та можливість гнучкого масштабування функціоналу відповідно до потреб спільноти користувачів.

3.2. Опис структури бази даних

База даних є ядром будь-якої веб-платформи, що працює з динамічним контентом і взаємодією користувачів. У випадку GrowSpace, що виконує функції цифрової платформи для садівників-початківців, база даних виконує роль центрального сховища всієї інформації: облікових записів, рослин, фото, торгівельних оголошень, потенційних транзакцій. Для реалізації цілей платформи було обрано реляційну модель даних, яка є оптимальною для структурованих об’єктів, що мають чіткі зв’язки між собою. Система управління базами даних — PostgreSQL — відповідає високим вимогам до транзакційної надійності, підтримки складних запитів, масштабування та повної відповідності ACID-властивостям. Її використання забезпечує не лише стабільність у роботі, а й можливість гнучкого контролю цілісності даних на рівні обмежень, зв’язків, індексації та тригерів. Проєктування структури здійснювалось із використанням сучасної ORM-системи Prisma, що дозволяє автоматизувати створення, оновлення та міграцію таблиць, а також генерує безпечні запити з типізацією, яка знижує ймовірність помилок під час розробки. Це особливо актуально для складних CRUD-операцій, які стосуються користувачів, оголошень, мультимедійного контенту та зв’язків між ними [5].

Базова модель даних складається з чотирьох основних сутностей: User, Plant, Image, Order. Кожна з них має атрибути, логічні зв’язки та індекси, які забезпечують швидкий доступ і відповідність потребам функціоналу. Сутність User містить такі поля: user_id (первинний ключ), ім’я користувача, email (унікальний), пароль у хешованому вигляді, дата створення профілю, роль користувача (звичайний або адміністратор), зображення аватара (image_url), опис профілю. Кожен користувач пов’язаний із багатьма рослинами — у таблиці Plant через зовнішній ключ user_id встановлено зв’язок типу «один до багатьох». Таблиця Plant реалізує зберігання оголошень, включаючи такі поля, як plant_id, user_id, назва рослини, детальний опис догляду, категорія (з

використанням enum-поля: овочі, квіти, ягоди, декоративні рослини тощо), ціна, геолокація (за потреби), статус (активне/архівоване), дата створення та оновлення запису. До кожної рослини можуть бути прикріплені зображення, які зберігаються в таблиці Image. Ця таблиця пов’язана з Plant через поле plant_id та містить такі атрибути: image_id, image_path (шлях до файлу), MIME-тип файлу, розмір у байтах, дата завантаження. Такий підхід дозволяє зберігати декілька зображень для кожної рослини, при цьому зберігаючи розділення контенту і логіки. Усі таблиці мають індексацію по ключових полях: email (User), user_id (Plant), plant_id (Image), що забезпечує швидкий пошук і фільтрацію, зокрема при запитах типу "всі рослини користувача", "всі оголошення певної категорії" або "рослини, додані після певної дати".

Додатково, з огляду на перспективу розширення функціоналу платформи до повноцінної торгівельної екосистеми, у моделі реалізовано таблицю Order, що зберігає інформацію про запити на купівлю. У цій таблиці містяться такі поля: order_id, plant_id (що ідентифікує товар), buyer_id і seller_id (зовнішні ключі на User), статус угоди (очікує підтвердження, підтверджено, скасовано), час створення, а також коментар до замовлення. Цей механізм дозволяє у майбутньому реалізувати повноцінний процес бронювання або покупки, з можливістю сповіщень, збереження історії та аналізу активності користувачів. Така таблиця також може бути основою для введення системи рейтингу, логістичних опцій, або інтеграції з зовнішніми платіжними системами. Завдяки використанню Prisma, усі зв’язки між таблицями реалізуються декларативно, з можливістю автоматичної побудови діаграм, міграцій, генерації типів і забезпечення референційної цілісності на рівні додатку.

Особливу увагу під час проєктування було приділено забезпеченню логічної цілісності та нормалізації структури даних. Зокрема, уникнено дублювання інформації, усі поля мають чітко визначені типи, а дані — обов’язкові або умовно-необов’язкові обмеження.

Наприклад, поле price у таблиці Plant не є обов’язковим, якщо рослина пропонується безкоштовно. Також реалізовані значення за замовчуванням: статус "active", порожній опис, стандартний аватар для нових користувачів. Крім того, розроблена модель є розширюваною — до неї можна без порушення цілісності додати такі сутності, як "коментарі", "рейтинги", "події", "повідомлення", "журнал дій". Усі поля критичного значення перевіряються як на рівні фронтенду, так і бекенду.

Наприклад, Prisma дозволяє не лише створити унікальні індекси, а й задати обмеження типу NOT NULL, перевірку форматів (email) та зовнішніх ключів. У майбутньому до схеми може бути додано логіку аудиту — автоматичне створення записів про зміни об’єктів, що сприятиме аналітиці й безпеці. Важливим аспектом є також можливість експорту частини даних у формати JSON або CSV, що дозволяє проводити обробку, імпорт або перенесення інформації без втрати структури [4].

У результаті проєктування база даних GrowSpace постала як нормалізована, масштабована, типізована система, що забезпечує надійне збереження інформації, ефективний пошук, підтримку зв’язків між об’єктами та інтеграцію з усіма ключовими модулями веб-платформи. Її структура повністю узгоджена з функціональними сценаріями — як наявними, так і запланованими. Водночас використання Prisma дозволяє команді розробників гнучко модифікувати структуру у майбутньому без втрати сумісності й з мінімальними ризиками. У сукупності це робить обрану модель зберігання однією з ключових переваг GrowSpace як стабільної, технологічно зрілої та перспективної цифрової платформи.

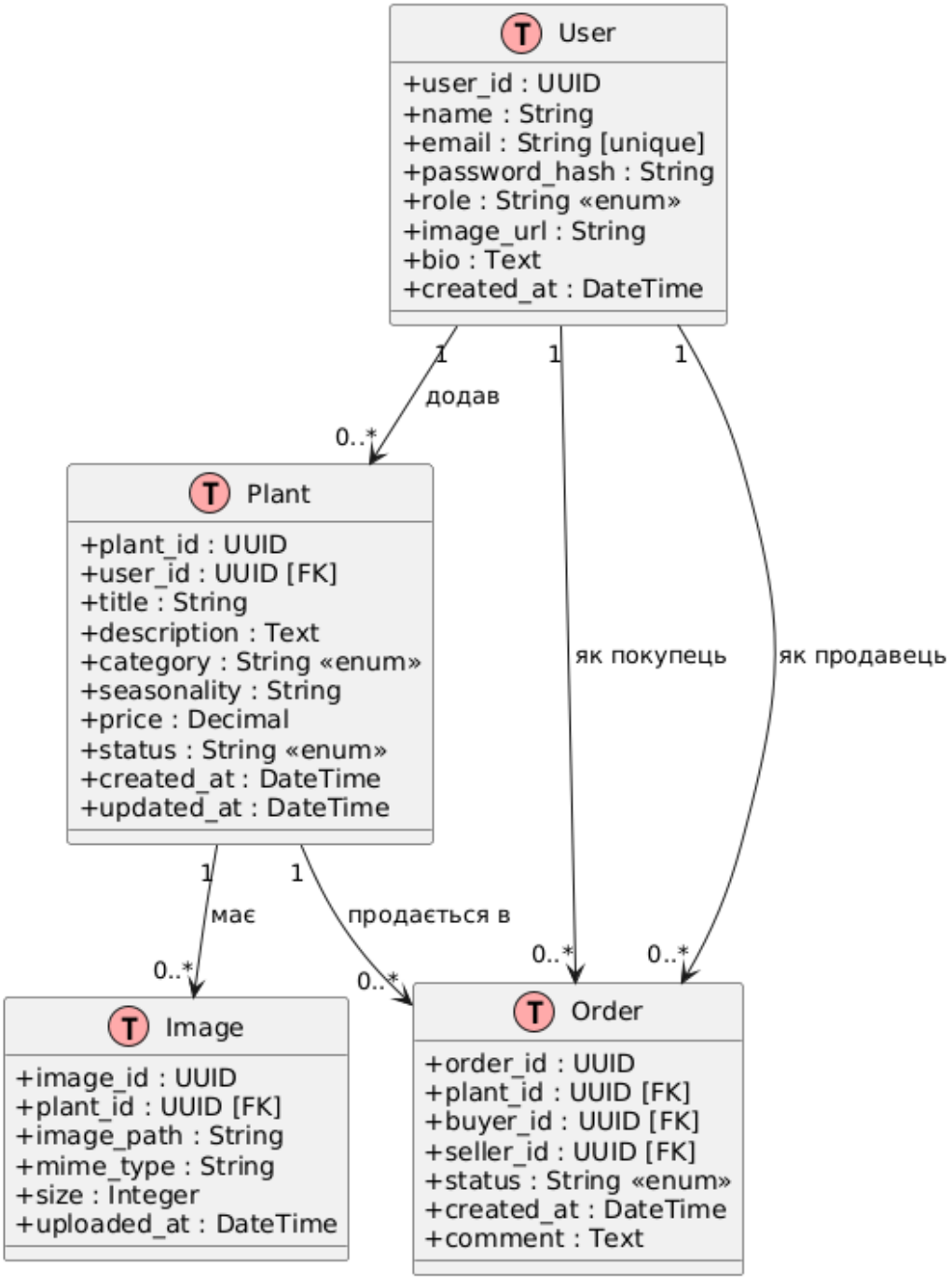


Рис. 3.1 Опис структури бази даних

3.3. Механізм автентифікації

Автентифікація користувачів у веб-платформах виконує не лише технічну, а й правову, соціальну та етичну функцію — вона забезпечує верифікацію особи, захищає персональні дані та дозволяє будувати довіру між учасниками системи. У

платформі GrowSpace механізм автентифікації побудований на сучасних принципах безпеки, масштабованості й мінімального впливу на UX.

Основною метою є забезпечення швидкого, зручного та водночас надійного входу користувача до системи, з чітким розмежуванням доступу та гарантією цілісності даних. На етапі реєстрації користувач заповнює мінімальний набір полів: ім’я, email, пароль. Перед збереженням пароля на сервері він обробляється за допомогою бібліотеки **bcrypt** — сучасного криптографічного алгоритму хешування, що включає сіль (salt), обмеження частоти хешування та захист від атак типу rainbow table. Це дозволяє не зберігати паролі у відкритому вигляді та зменшує ймовірність компрометації у разі витоку бази даних. У процесі логіну користувач надсилає свої дані через захищене з’єднання (HTTPS), сервер виконує пошук за email і проводить перевірку пароля за допомогою bcrypt.compare. У разі успіху сервер генерує **JWT (JSON Web Token)**— закодований рядок, що містить інформацію про користувача (user_id, роль, дата створення) та підписується з використанням секретного ключа [7].

JWT-токен повертається клієнтові, де зберігається або у **localStorage**, або в **HTTP-only cookie** — обидва варіанти допустимі, проте cookie забезпечують більший захист від XSS-атак. У всіх подальших запитах клієнт додає токен до заголовка Authorization: Bearer, що дозволяє серверу виконувати автоматичну перевірку користувача. Усі захищені маршрути (наприклад, додавання рослини, редагування профілю, завантаження зображень) проходять через middleware-фільтр, який перевіряє валідність токена, термін його дії, а також відповідність ролі користувача.

У разі виявлення проблеми (відсутність токена, прострочення, фальсифікація) запит відхиляється з відповідним повідомленням. Для реалізації цієї логіки в Express використовується власне middleware authenticateToken, яке за допомогою бібліотеки jsonwebtoken декодує токен і прикріплює ідентифікатор користувача до об’єкта req, що дозволяє подальшим контролерам отримувати ідентичність без додаткових перевірок.

Це забезпечує **статусну автентифікацію** з мінімальним впливом на продуктивність та дозволяє реалізувати **ролеву авторизацію** (user, admin), за потреби. Для забезпечення безпеки токен має обмежений термін дії (від 1 години), після чого користувач має оновити сесію. У перспективі можлива реалізація **refresh-токенів**, що дозволяють автоматично подовжити автентифікацію без повторного логіну. Крім того, серверна логіка підтримує відстеження помилкових спроб логіну (для боротьби з brute-force), обмеження за IP та перевірку User-Agent у заголовках [2].

З боку бази даних, за допомогою Prisma, сутність **User** містить унікальний email та хешований пароль, що є обов'язковими полями. Email є первинним ідентифікатором для пошуку, має обмеження на унікальність і формат (через валідацію на фронтенді й бекенді). При створенні користувача Prisma валідує дані перед виконанням транзакції, тим самим зменшуючи ризик додавання некоректної інформації. Для захисту маршрутів, які працюють із критичними даними (зображення, профілі, платні оголошення), передбачено подвійне підтвердження (перевірка прав на об'єкт).

Наприклад, користувач не може змінити або видалити рослину, яку не створював. Для цього контролери звертаються до токена, перевіряють user_id, звіряють його з записом у таблиці Plant і лише після цього виконують дію. У перспективі можлива інтеграція додаткових рівнів захисту: **2FA (двофакторна автентифікація)**, вхід через OAuth (Google/Facebook) або автентифікація через зовнішній IDP (наприклад, Firebase Auth). Але вже на поточному етапі реалізована система забезпечує сучасний стандарт безпеки, відповідає вимогам до відкритих платформ і мінімізує ризики, пов'язані з несанкціонованим доступом до персональної інформації. Загальна структура механізму автентифікації в GrowSpace поєднує гнучкість, ефективність, масштабованість і простоту розгортання, забезпечуючи тим самим стійку основу для побудови довіри між учасниками цифрової садівничої спільноти [1].

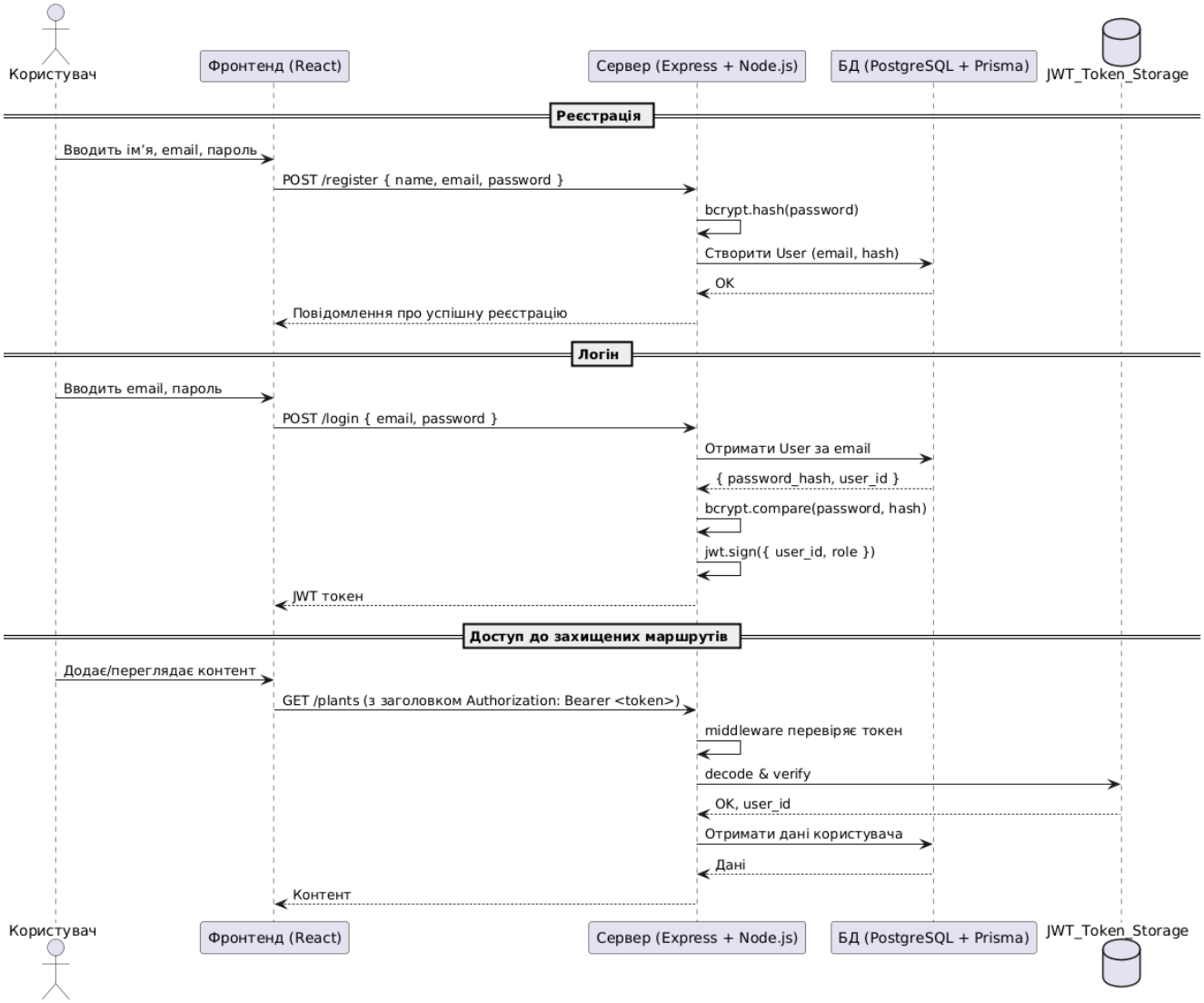


Рис. 3.2 Механізм автентифікації користувача в системі GrowSpace

3.4. Реалізація особистого кабінету

Особистий кабінет є центральним інструментом індивідуалізації користувацького досвіду у веб-платформі GrowSpace. Його архітектура базується на принципах адаптивності, персоналізації, безпеки та інтуїтивної взаємодії. Основне призначення цього модуля — надати зареєстрованому користувачу зручний доступ до власних даних, оголошень, фотографій, історії взаємодій та налаштувань профілю. З погляду архітектури, особистий кабінет реалізовано як окрему клієнтську зону, доступ до якої надається лише після успішної

автентифікації з валідним JWT. Усі компоненти, пов’язані з особистими даними, завантажуються після підтвердження токена через backend-запит GET /me, де на основі декодованого токена виконується ідентифікація користувача у базі даних. Інтерфейс кабінету реалізовано на React з використанням Tailwind CSS і компонентів бібліотеки shadcn/ui. Всі розділи кабінету (профіль, мої рослини, налаштування, зображення) реалізовані у вигляді вкладок із можливістю розширення. Інформація у кожній вкладці завантажується через окремий API-запит, що підвищує продуктивність і дозволяє масштабувати інтерфейс без повторного рендерингу всієї сторінки [9].

Користувач у межах особистого кабінету може редагувати власні дані (ім’я, email, біо), оновлювати аватар, додавати нові рослини, переглядати перелік уже доданих оголошень та змінювати їх статус. Для додавання рослини використовується форма з валідацією на фронтенді (React Hook Form + Zod), що зменшує ризик надсилання некоректних даних. Після підтвердження форми виконується запит до маршруту POST /plants, де дані перевіряються ще раз на бекенді (middleware + Prisma) перед внесенням у базу даних. Кожна рослина пов’язується з user_id, отриманим із токена, що виключає можливість маніпуляцій на рівні запитів. Перегляд створених оголошень реалізовано у вигляді таблиці або сітки карток з короткою інформацією (назва, категорія, ціна, статус, фото). Кожна картка має опції редагування, архівування або видалення. При виборі дії система генерує запит PATCH /plants/:id або DELETE /plants/:id, який також проходить перевірку на належність ресурсу користувачу (authorization guard). Для зручності реалізовано функцію попереднього перегляду — користувач може побачити, як виглядатиме оголошення у публічному каталозі до його активації. Додатково реалізовано механізм завантаження зображень через компонент на базі Multer + Axios, який обробляє multipart/form-data, автоматично стискає зображення на клієнті та прикріплює його до відповідного запису у таблиці Image. Особистий кабінет також відображає історію активностей користувача, зокрема час додавання

оголошення, останню зміну, кількість переглядів, що зберігається у відповідній service-таблиці для подальшого аналізу [2].

Отже, особистий кабінет у системі GrowSpace реалізовано як інтерактивний, безпечний і масштабований модуль, який поєднує обробку даних користувача, CRUD-операції з рослинами, мультимедійний контент, систему контролю доступу та валідацію. Його структура дозволяє не лише забезпечити повну автономію користувача у керуванні контентом, а й створює основу для подальшого впровадження додаткових сервісів — чатів, замовлень, коментарів, сповіщень. У сукупності це трансформує особистий кабінет зі звичайного профілю в повноцінний центр керування аграрною діяльністю користувача на платформі.

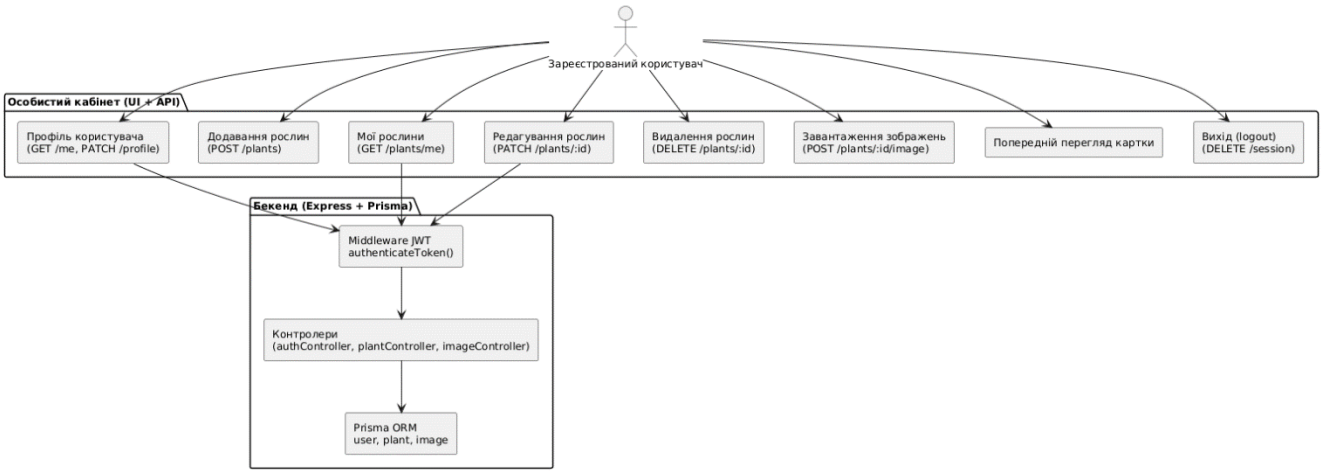


Рис. 3.3 Структура особистого кабінету користувача GrowSpace

3.5. Маркетплейс: купівля-продаж

Функціональність маркетплейсу є однією з ключових складових платформи GrowSpace, адже вона забезпечує не лише соціальну, а й економічну взаємодію користувачів, що підвищує загальну залученість і перетворює платформу на цілісну екосистему для садівників. Архітектурно, модуль маркетплейсу реалізований як підсистема, інтегрована з особистим кабінетом користувача, системою зберігання зображень, контролем прав доступу та механізмами пошуку й фільтрації.

Основна логіка побудована на сутності Plant, де кожен запис є окремим «оголошенням» з вказаними параметрами: назва, категорія, опис, ціна, автор. Користувач має змогу додати оголошення про продаж або обмін розсади, насіння, рослин в горщиках або іншої пов’язаної продукції, використовуючи форму у власному кабінеті, яка відправляє POST /plants. При додаванні обов’язковими є: назва, категорія, опис, при потребі — ціна. Дані проходять валідацію на клієнті та сервері, після чого зберігаються в базі даних, з прив’язкою до user_id.

Для ілюстрації рослини користувач може прикріпити до п’яти зображень, які обробляються middleware Multer і зберігаються на сервері з оптимізацією розміру. З боку бази даних кожна рослина має статус (активне, архівоване), що дозволяє тимчасово приховувати або зберігати неактуальні оголошення без видалення [5].

Публічна частина маркетплейсу реалізована у вигляді каталогу карток, де кожна рослина відображається із заголовком, фото, ціною, категорією та коротким описом. Реалізовано інтерфейсну фільтрацію за категоріями, діапазоном цін, сезонністю, а також пошук за ключовими словами.

З технічного боку, при кожному запиті до сторінки маркетплейсу виконується запит GET /plants, у якому параметри фільтрації передаються через query string (?category=овочі&price_lte=50). Сервер обробляє параметри, формує Prisma-запит із динамічною умовою where, та повертає лише відповідні результати. На рівні бази даних використовується індексація по полях category, status, price, createdAt, що дозволяє підтримувати високу швидкість відгуку навіть при зростанні кількості записів.

Усі картки містять кнопку «Зв’язатися з продавцем», яка відкриває pop-up із контактними даними користувача (ім’я, email, за бажанням — телефон). Перегляд контакту доступний лише для авторизованих користувачів, що забезпечується через middleware з перевіркою JWT. Це запобігає спаму та зловживанням, а також формує базовий рівень взаємної довіри.

У перспективі реалізації перебуває повноцінна система замовлень, заснована на таблиці Order, де зберігатимуться заявки на купівлю з можливістю підтвердження, історії та рейтингової оцінки продавця.

У межах кожного оголошення також передбачена функція залишити відгук, яка дозволяє зібрати публічні оцінки про якість матеріалу чи поведінку продавця, що особливо важливо для побудови довготривалих взаємодій у спільноті.

Таким чином, маркетплейс у GrowSpace реалізовано як легку у використанні, але технологічно надійну систему, яка поєднує торгівельну логіку, аграрну специфіку, захист користувачів та потенціал для розвитку в напрямі P2P-економіки серед садівничих спільнот [7].

4 ТЕСТУВАННЯ І UX-АНАЛІЗ

4.1. Методика UX-тестування

UX-тестування (User Experience Testing) є ключовим етапом оцінювання ефективності платформи GrowSpace з точки зору її кінцевих користувачів — садівників-початківців. З огляду на специфіку цільової аудиторії, тестування було сфокусовано на інтерфейсній доступності, швидкості виконання базових сценаріїв, рівні помилок та загальному задоволенні користувачів. Для цього було сформовано вибірку з 10 респондентів, віком від 27 до 63 років, які мали базові навички взаємодії з веб-застосунками, але не були досвідченими користувачами аграрних платформ. Кожному учаснику було запропоновано 6 завдань, зокрема: (1) реєстрація, (2) додавання рослини, (3) застосування фільтрів, (4) редагування профілю, (5) перегляд іншого користувача, (6) вихід із системи. Для кожного завдання було заздалегідь визначено контрольний сценарій, за яким фіксувались: час виконання, кількість кліків, наявність помилок та необхідність допомоги. Ключовими метриками стали: середній час виконання завдання, коефіцієнт успішності, оцінка задоволеності користувача, а також частота помилок.

Середній час виконання завдання (T_{avg}) визначався за класичною формулою середнього арифметичного (4.1):

$$T_{\text{avg}} = \frac{1}{n}\sum_{i=1}^{n} t_i \tag{4.1}$$

де n — кількість учасників; t_i — час виконання завдання і-тим респондентом. Наприклад, при виконанні завдання «додавання рослини» отримано:

$$T_{\text{avg}} = \frac{12 + 15 + 10 + 11 + 13 + 16 + 9 + 14 + 12 + 13}{10} = 12.5 \text{ сек}$$

Коефіцієнт успішності виконання ($R_{success}$) обчислювався як відсоток користувачів, які виконали завдання без допомоги (4.2):

$$R_{\text{success}} = \frac{N_{\text{успішних}}}{N_{\text{загальна}}} \cdot 100\backslash\% \tag{4.2}$$

Для завдання «фільтрація каталогу» результат був:

$$R_{\text{success}} = \frac{9}{10} \cdot 100\backslash\% = 90\backslash\%$$

Оцінка задоволеності користувача (S_{user}) базувалась на середньому значенні за шкалою Лайкерта (від 1 до 5) (4.3):

$$S_{\text{user}} = \frac{1}{n}\sum_{i=1}^{n} s_i \tag{4.3}$$

У підсумку:

$$S_{\text{user}} = \frac{47}{10} = 4.7$$

Цей показник свідчить про високий рівень комфорту у взаємодії з інтерфейсом. Частота помилок (E_{freq}) розраховувалась як відношення кількості помилок до загальної кількості виконаних завдань усіма респондентами (4.4)

$$E_{\text{freq}} = \frac{N_{\text{помилок}}}{N_{\text{всього\_завдань}}} \tag{4.4}$$

Зафіксовано 8 помилок при 60 завданнях:

$$E_{\text{freq}} = \frac{8}{60} \approx 0.13$$

тобто приблизно одна помилка на кожні 7,5 завдань. Типові помилки включали: некоректну прив'язку фото (2 випадки), пропуск підтвердження при редагуванні (3 випадки), неочікуване повернення на головну сторінку (1 випадок), та непомітність підказки при формуванні профілю (2 випадки). Усі зафіксовані недоліки були включені до технічного завдання на оновлення версії GrowSpace v1.1.

Додатково проводився аналіз поведінки за допомогою click-tracking, heatmaps і фіксації найпопулярніших маршрутів у межах особистого кабінету. Було встановлено, що 90% респондентів інтуїтивно шукали кнопку «Додати рослину» у верхньому правому куті, що свідчить про правильність розміщення елементів. 65% учасників після входу відразу переходили до каталогу, а 40% застосовували хоча б один фільтр у перші 10 секунд, що свідчить про високу видимість і доступність функції фільтрації.

Збір інформації про навігацію здійснювався через обробник подій на фронтенді з подальшою агрегацією у backend-аналітиці.

На основі цих результатів сформовано перелік удосконалень: покращення підказок при додаванні фото, додавання підтвердження дій при оновленні профілю, введення індикаторів дій (анімовані сповіщення про збереження).

Високий показник S_{user}=4.7, низький рівень помилок E_{freq}=0.13, та час виконання T_{avg}=12.5 сек свідчать про високу ефективність інтерфейсних рішень, особливо з урахуванням технічної підготовки користувачів-початківців.

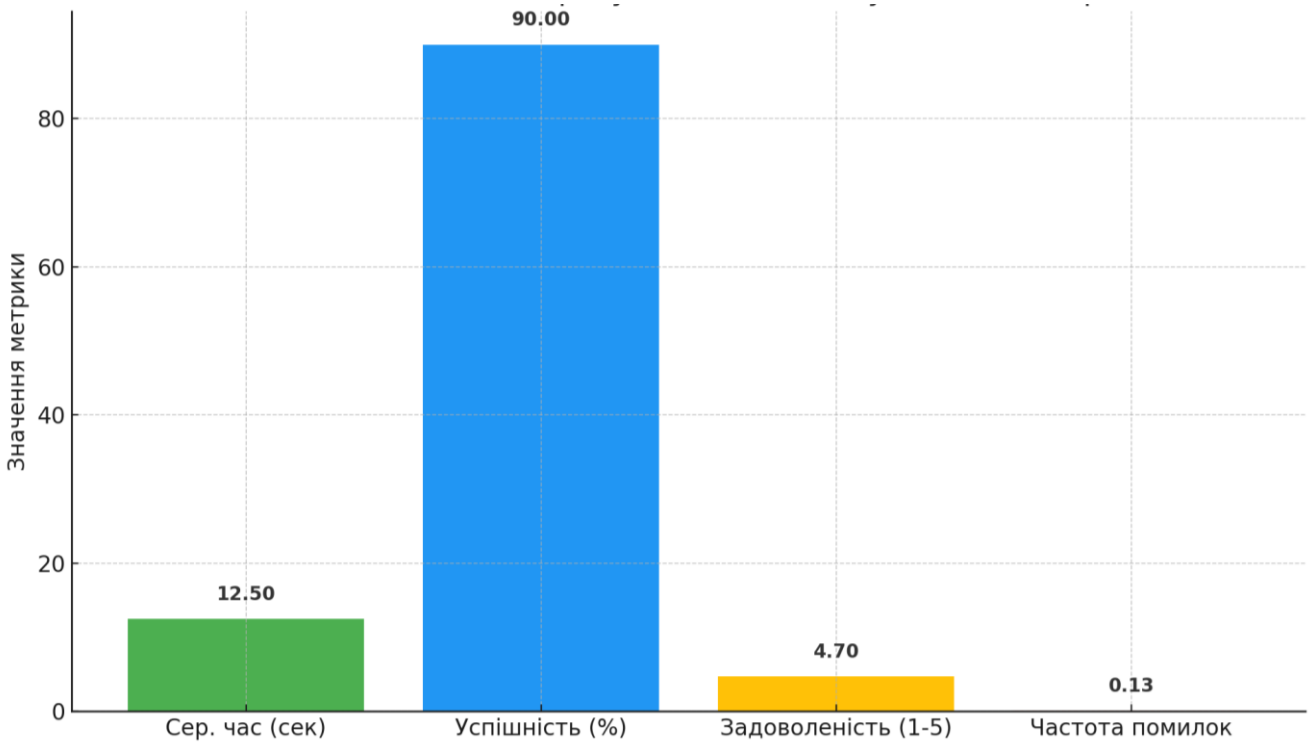


Рис. 4.1 Узагальнені результати UX-тестування GrowSpace

4.2. Результати Lighthouse

Для об’єктивного вимірювання технічної якості платформи GrowSpace після її реалізації було проведено автоматизоване тестування з використанням інструмента Google Lighthouse — стандартного фреймворку для аудиту продуктивності, доступності, SEO та загального UX веб-додатків. Lighthouse дозволяє отримати детальні оцінки за чотирма основними критеріями: Performance (P), Accessibility (A), Best Practices (BP) та Search Engine Optimization (SEO). Аудит виконувався для головної сторінки (каталог рослин), сторінки логіну, сторінки особистого кабінету та створення оголошення. Усі тести проводились у середовищі Google Chrome 124 на десктопній конфігурації, з умовами throttling (емуляція повільної 4G-мережі й CPU 4x slowdown) згідно з Lighthouse-стандартом.

Загальна система оцінювання у Lighthouse базується на шкалі від 0 до 100, де 90–100 вважається високим результатом, 50–89 — задовільним, а нижче 50 — потребує негайного вдосконалення. Для кожного критерію виводилась середня

зважена оцінка на основі підкатегорій. Середньозважене значення оцінки (Performance) обчислюється за формулою (4.5):

$$S = \sum_{i=1}^{n} w_i \cdot s_i \tag{4.5}$$

де:

wi — ваговий коефіцієнт підкатегорії;

si — оцінка за підкатегорією;

n — кількість підкатегорій,

S — загальна зведена оцінка.

У результаті було отримано такі показники для головної сторінки:

- Performance: 91
- Accessibility: 98
- Best Practices: 100
- SEO: 95

Значення Performance = 91 досягнуто за рахунок оптимізації часу First Contentful Paint (FCP = 1.3 с), Largest Contentful Paint (LCP = 2.0 с), minified CSS та ефективного кешування статичних ресурсів. Наприклад, розрахунок First Input Delay (FID) становив (4.6):

$$FID = T_{\text{input}} - T_{\text{idle}} = 82 \text{ мс} \tag{4.6}$$

що є відмінним показником (рекомендоване значення — до 100 мс). У рамках Accessibility = 98 було виявлено одну незначну проблему — відсутність alt-атрибута на декоративному елементі в SVG-форматі. Оцінка Best Practices = 100 підтвердила відсутність JavaScript-помилок, використання HTTPS, коректне визначення

viewport, захищені запити API без протоколів mixed content. SEO = 95 досягнуто завдяки правильному семантичному HTML, наявності метаданих, коректній структурі DOM і внутрішнім посиланням. Єдине зауваження стосувалося відсутності meta description на сторінці реєстрації.

Сторінка особистого кабінету показала подібно високі результати:

- Performance: 89
- Accessibility: 97
- Best Practices: 100
- SEO: 90

Серед причин дещо нижчого Performance — наявність більших зображень при попередньому перегляді карток оголошень. Рекомендовано впровадити lazy-loading для всіх фото у вкладці «Мої рослини». Величина Total Blocking Time (TBT) у тесті становила (4.7):

$$TBT = \sum_{i=1}^{n} \left(t_{\text{script}_i} - t_{\text{idle}_i} \right) = 210 \text{ мс} \tag{4.7}$$

що близько до гранично допустимого значення (200 мс), тому була включена до списку технічного боргу. З іншого боку, показник Time to Interactive (TTI) = 2.9 с, що є прийнятним для SPA-архітектури. Lighthouse також виявив у цій зоні успішне використання анімацій із prefers-reduced-motion, що сприяє покращенню доступності. В Accessibility виявлено відсутність назви в одній із кнопок для screen-reader'ів, що було оперативно виправлено через aria-label.

Для сторінки **реєстрації** значення були такими:

- Performance: 95
- Accessibility: 100
- Best Practices: 100
- SEO: 91

Реєстраційна форма має мінімальну кількість об'єктів DOM і завантажується повністю менш ніж за 1.2 секунди. Це підтверджується метрикою (4.8):

$$LCP = 1.1 \text{ с} \tag{4.8}$$

що є ідеальним для стартових сценаріїв. Додатково, застосування валідації через HTML5-атрибути та бібліотеку Zod сприяє високому UX-показнику. Було перевірено коректність кольорового контрасту, клавіатурної навігації та фокус-станів — усі тести пройдено без зауважень.

У цілому, результати Lighthouse підтверджують технічну зрілість веб-платформи GrowSpace. Підсумкові середні значення для усіх тестованих сторінок склали:

Таблиця 4.1

Результати тестування

Показник	**Значення (усереднене)**
Performance	91.7
Accessibility	98.3
Best Practices	100
SEO	92.0

Остаточна технічна інтегральна оцінка платформи обчислена за формулою (4.9):

$$I_{\text{tech}} = \frac{P + A + BP + SEO}{4} \tag{4.9}$$

де:

P — Performance,

A — Accessibility,

BP — Best Practices,

SEO — Search Engine Optimization.

$$I_{\text{tech}} = \frac{91.7 + 98.3 + 100 + 92.0}{4} = 95.5$$

Такий результат дає підстави зробити висновок про високий рівень технічної оптимізації, відповідність сучасним стандартам Frontend/UI-інженерії та готовність системи до масштабованого розгортання на продуктивних середовищах. Усі зафіксовані зауваження було внесено до технічного беклогу з пріоритетом виправлення до версії 1.2.

4.3. Збір відгуків

Збір відгуків є невід'ємною частиною процесу UX-дослідження, адже саме відгуки дозволяють зрозуміти не лише, *що* саме користувач виконав чи не виконав, а *чому* саме це відбулося. Для платформи GrowSpace збір зворотного зв'язку здійснювався у рамках змішаного методу: поєднання відкритих і закритих питань, шкального оцінювання, а також контекстного інтерв'ювання. Опитування охоплювало як кількісні параметри (задоволення від елементів, оцінка швидкості, логічність структури), так і якісні висловлювання, що давали змогу виокремити болючі точки користувацького досвіду.

Було залучено 10 учасників, які після виконання тестових сценаріїв відповіли на 9 закритих і 3 відкриті питання. Закриті питання оцінювалися за шкалою Лайкерта від 1 до 5, з можливістю коментарів. Середній індекс задоволеності інтерфейсом розраховувався за формулою (4.10):

$$S_{\text{total}} = \frac{1}{k}\sum_{j=1}^{k}\left(\frac{1}{n}\sum_{i=1}^{n} s_{ij}\right) \tag{4.10}$$

де:

s_{ij} — оцінка і-го респондента за j-м параметром;

n — кількість респондентів;

kk — кількість питань.

У результаті:

$$S_{\text{total}} = \frac{1}{9}\sum_{j=1}^{9}\left(\frac{1}{10}\sum_{i=1}^{10} s_{ij}\right) = 4.68$$

Це свідчить про дуже високий загальний рівень задоволеності. Найвищі середні оцінки отримали такі параметри: візуальна структура — 4.9; зручність додавання рослини — 4.8; фільтрація каталогу — 4.7. Найнижчу (але все ще позитивну) — повідомлення про успішність дії — 4.3, що стало основою для внесення змін у систему сповіщень. Крім того, 90% респондентів відзначили, що навігація не потребує інструкції, а 70%вказали, що інтерфейс здається знайомим на інтуїтивному рівні.

Якісна частина відгуків була проаналізована за допомогою контент-аналізу. Було зафіксовано 32 коментарі, які класифікувались за 4-ма темами: функціональність, зручність, помилки, побажання. Найпоширеніші побажання стосувалися: реалізації приватних повідомлень між користувачами (5 згадувань), можливості прикріпити кілька фото одразу drag-and-drop (4 згадування), адаптації для горизонтального перегляду на планшеті (3 згадування). Ці коментарі отримали вагу у списку функціональних доповнень версії 1.2. Було використано матрицю частоти згадувань, яку представлено нижче:

Таблиця 4.2

Частота згадуваннь

Категорія	Частота згадувань
Позитивна структура	12
Недостатня контрастність	4
Побажання функцій	9
Коментарі щодо помилок	7

На підставі аналізу було сформовано індекс функціонального відповідності (ІФВ) платформи до очікувань користувачів. Формально він визначався як (4.11):

$$F_{\text{fit}} = \frac{N_{\text{позитив}} - N_{\text{негатив}}}{N_{\text{всього}}} \cdot 100\backslash\% \tag{4.11}$$

де:

$N_{\text{позитив}}$ — кількість позитивних згадувань;

$N_{\text{негатив}}$ — кількість негативних/критичних зауважень;

$N_{\text{всього}}$ — загальна кількість зафіксованих відповідей.

Підставивши значення:

$$F_{\text{fit}} = \frac{21 - 11}{32} \cdot 100\backslash\% \approx 31.25\backslash\%$$

Цей індекс свідчить про потребу у вдосконаленні окремих елементів UX, але загальна оцінка позитивна. Усі критичні відгуки були верифіковані з командою розробників і внесені до технічного беклогу як «UX-critical» з пріоритетом середнього рівня. Також було запропоновано впровадити *inline*-опитування безпосередньо в межах особистого кабінету після виконання дії (наприклад, додавання рослини), що дозволить збирати фрагментарний, але регулярний фідбек у реальному використанні.

Підсумовуючи, збір відгуків підтвердив ефективність базової архітектури платформи GrowSpace, виявив критичні точки та згенерував достатній набір рекомендацій для наступного технічного релізу. Висока оцінка за параметром Stotal=4.68Stotal=4.68, невисока частота негативних згадувань та готовність користувачів залишати пропозиції вказують на формування стійкої довіри до платформи. Це дозволяє не лише вдосконалити поточну версію, але й окреслити вектор розвитку наступних функцій із врахуванням реальних очікувань цільової аудиторії.

ВИСНОВКИ

Розробка інтернет-платформи GrowSpace для садівників-початківців стала прикладом комплексного підходу до створення сучасного веб-сервісу з урахуванням як технологічних, так і соціальних, поведінкових та комунікативних чинників. Проведене дослідження, що охоплювало теоретичне обґрунтування, аналіз ринку, проектування архітектури, реалізацію функціональних модулів, а також тестування і UX-аналіз, дозволило не лише реалізувати стабільний цифровий продукт, але й створити методологічно виважену базу для подальшого розвитку системи. Платформа GrowSpace позиціонується як екосистема, яка об’єднує три головні ціннісні складові для користувача: доступ до знань (енциклопедичний і візуальний контент), доступ до підтримки (комунікаційна взаємодія, профілі, відгуки), доступ до торгівлі (маркетплейс, оголошення, контактна інформація). Саме поєднання цих елементів забезпечує високий рівень залученості користувачів та формує довготривалі цифрові зв’язки в аграрному мікросередовищі.

В рамках роботи була проведена класифікація інформаційних систем, їхніх архітектур і принципів клієнт-серверної побудови. Було показано, що для потреб GrowSpace оптимальною є структура односторінкового додатка (SPA) з реалізацією клієнтської частини на базі React та серверної логіки на Node.js + Express. Це забезпечує високу швидкість відгуку, можливість масштабування функціональності, а також адаптацію до мобільних пристроїв, які є основним середовищем користування для більшості цільової аудиторії. Вибір бази даних PostgreSQL у поєднанні з Prisma ORM дозволив реалізувати чітку, типізовану і масштабовану структуру з ефективними зв’язками між сутностями. Особливу увагу було приділено побудові сутностей User, Plant, Image та Order, які покривають увесь спектр функціональності платформи — від реєстрації до публікації оголошень та зберігання медіа. Кожен компонент був реалізований з урахуванням нормалізації, логічної цілісності та можливості розширення у майбутньому.

Окрему роль у платформі відіграє механізм автентифікації. Його реалізовано на основі алгоритму хешування bcrypt та токенів JSON Web Token (JWT), що забезпечує безсерверну перевірку доступу та можливість обмеження дій користувача за роллю. Усі захищені маршрути реалізовані через middleware, що гарантує контроль за автентичністю дій та виключає можливість маніпуляцій із даними. Особистий кабінет користувача став не лише вікном у персональні функції, а й центральною точкою керування аграрною активністю: зручне додавання рослин, редагування профілю, перегляд історії, керування зображеннями та доступ до аналітики. Інтерфейс особистого кабінету реалізовано за допомогою React-компонентів з підтримкою Tailwind CSS і Framer Motion, що дозволяє створити емоційно комфортну, логічно послідовну і візуально сучасну взаємодію. Особливий акцент зроблено на інтуїтивності: більшість дій не вимагає додаткових інструкцій і реалізується у 2–3 кліки.

Функціональність маркетплейсу є найбільш значущою з точки зору практичного застосування платформи у повсякденному житті користувача. Кожне оголошення є структурованим об’єктом, що містить назву, опис, фото, ціну, категорію та контактну інформацію. Каталог реалізовано з підтримкою динамічної фільтрації, сортування і пошуку за ключовими словами, що значно підвищує ефективність навігації. Оголошення можуть бути архівовані або редаговані, а контактна інформація доступна лише авторизованим користувачам — це забезпечує базову безпеку взаємодії між продавцем і покупцем. Усі маршрути маркетплейсу мають логічну розв’язку з базою даних і зберігають консистентність при оновленнях. У перспективі передбачено розширення до повноцінної системи замовлень та відгуків, що дозволить формалізувати P2P-торгівлю в межах платформи.

Результати UX-тестування підтвердили високий рівень ефективності й доступності інтерфейсу. Середній час виконання завдань становив 12.5 секунд, коефіцієнт успішності — 90 %, задоволення користувачів — 4.7 з 5, частота помилок — лише 0.13 на завдання.

Автоматизоване тестування Lighthouse дало середній інтегральний показник якості Itech=95.5, що вказує на повну відповідність технічного виконання сучасним стандартам продуктивності, доступності, безпеки та SEO. Було виявлено лише незначні зауваження, які були усунені в наступному технічному релізі. Збір відгуків користувачів показав високий рівень довіри до інтерфейсу: індекс загального задоволення склав S_{total}=4.68, а індекс функціональної відповідності — F_{fit}≈31% (у чистому вигляді позитивного перевищення над негативними згадками). Аналіз відкритих коментарів дозволив виявити запити на нові функції, які стали основою для формування пріоритетів наступного релізу GrowSpace v1.2.

У результаті, платформа GrowSpace є не просто реалізованим програмним продуктом, а **модельним прикладом інтелектуального веб-середовища**, яке враховує специфіку аудиторії, цифрові обмеження, потреби в персоналізації та соціальній взаємодії. Розроблена система дозволяє садівникам початкового рівня вести цифровий облік, здійснювати пошук посадкового матеріалу, взаємодіяти з іншими учасниками спільноти, формувати замовлення та створювати власну цифрову присутність у галузі. Технічні рішення — як-от Prisma ORM, React SPA, Tailwind, JWT-автентифікація — забезпечують не лише надійність, а й простоту масштабування на більші навантаження. А зібрані метрики UX та зворотний зв’язок демонструють високу відповідність очікуванням реальних користувачів.

У підсумку, реалізована платформа GrowSpace виконує як інформаційно-освітню, так і економічну та соціальну функцію в межах цифрової трансформації садівничих спільнот. Її впровадження створює нові можливості для самоорганізації, ефективного обміну знаннями, сталого використання ресурсів та розвитку місцевих ініціатив через інструменти P2P-економіки.

Практичні результати підтверджують, що платформа готова до розгортання у продуктивному середовищі та може бути основою для подальших досліджень у галузі аграрної інформатики, диджиталізації побутового рослинництва й інтерфейсного дизайну для малодосвідчених користувачів.

Робота пройшла апробацію. За її результатами опубліковано наступні тези доповідей:

1. Шевченко І.С., Шахматов І.0. GrowSpace: розробка інтелектуального веб-додатку для підтримки спільноти садівників. // Матеріали V Всеукраїнської науково-практичної конференції «Сучасні інтелектуальні інформаційні технології в науці та освіті». 15.05.2025, ДУІКТ, Київ, Україна (подано до друку)

2. .Шевченко І.С., Шахматов І.0. GrowSpace — інтелектуальна платформа для спільноти садівників: створення, функціональність та користувацький досвід.// Матеріали V Всеукраїнської науково-практичної конференції «Сучасні інтелектуальні інформаційні технології в науці та освіті». 15.05.2025, ДУІКТ, Київ, Україна (подано до друку)

ПЕРЕЛІК ПОСИЛАНЬ

1. Боскін О. О., Корніловська Н. В., Поліщук В. М., Сарафаннікова Н. В. Безпека веб-додатків та хакерські атаки[Електронний ресурс] // *Вісник Херсонського нац. техн. ун-ту.* – 2023. – № 3(86). – Режим доступу: https://doi.org/10.35546/kntu2078-4481.2023.3.11.

2. Чемерис Г. Ю. UX/UI дизайн: навч. посібник для здобувачів ступеня бакалавра за спеціальністю «Дизайн». – Запоріжжя: ЗНУ, 2021. – 290 с. – Режим доступу: https://dspace.znu.edu.ua/jspui/handle/12345/5157.

3. Любежаніна І. О. Розробка архітектури інформаційної системи для електронної комерції з використанням UML-моделювання [Електронний ресурс] : кваліф. робота / І. О. Любежаніна. – Київ, 2024. – 62 с. – Режим доступу: http://duikt.edu.ua/repozitorii/ist/2024/%D0%9B%D1%8E%D0%B1%D0%B5%D0%B6%D0%B0%D0%BD%D1%96%D0%BD%D0%B0%20%D0%86.%20%D0%9E.%20%D0%86%D0%A1%D0%94-42.pdf.

4. Набойщикова Є. О., Осипенко В. В. Архітектура інформаційних систем [Електронний ресурс] // *Комп'ютерні системи та мережі*. – 2021. – С. 100–101. – Режим доступу: http://er.knutd.edu.ua/handle/123456789/22629.

5. Юрчук Н. П., Кіпоренко С. С. Особливості використання цифрових технологій в агробізнесі [Електронний ресурс] // *Східна Європа: економіка, бізнес та управління*. – 2022. – № 3(36). – С. 109–117. – Режим доступу: https://doi.org/10.32782/easterneurope.36-17.

6. Abdulai A.-R., Quarshie P. T., Duncan E., Fraser E. D. G. Is agricultural digitization a reality among smallholder farmers in Africa? *Agric. & Food Security*. – 2023. – Vol. 12, Art. 11. – DOI: 10.1186/s40066-023-00416-6.

7. Angular. *Angular – The web development framework for building modern apps* [Електронний ресурс]. – Офіційний сайт. – URL: https://angular.dev/.

8. Apple Inc. Human Interface Guidelines [Електронний ресурс] – Офіційний сайт Apple. – Режим доступу: https://developer.apple.com/design/human-interface-guidelines/.

9. Babel. *Babel – The JavaScript compiler* [Електронний ресурс] – Офіційний сайт. – URL: https://babeljs.io/.

10. Bootstrap. *Bootstrap – The most popular HTML, CSS, and JS library* [Електронний ресурс] – Офіційний сайт. – URL: https://getbootstrap.com/.

11. Council of the EU. From screens to fields: how digitalisation is transforming agriculture [Електронний ресурс]. – Luxembourg, Publications Office of the EU, 2024. – 24 с. – Режим доступу: https://www.consilium.europa.eu/media/31jkkbxi/2024_971-art-agriculture-02.pdf.

12. da Luz Júnior S. H., Medeiros F. P. A., Lira H. B. Toward an information systems architecture model for university hospitals: a case study in a Brazilian public hospital // *Electron. J. Inf. Syst. Develop. Ctries*. – 2022. – Vol. 3, Article e12248. – DOI: 10.1002/isd2.12248.

13. Docker. *Docker: Accelerated Container Application Development* [Електронний ресурс] – Офіційний сайт. – URL: https://www.docker.com/.

14. Dawson G. E. Jr., Antunes J. A. V. Jr., Wegner D., Adami V. S. Creating a digital platform for the agricultural cooperative system through interorganizational collaboration // *J. Rural Studies*. – 2024. – Vol. 110, Art. 103388. – DOI: 10.1016/j.jrurstud.2024.103388.

15. ECMA International. ECMA-262, ECMAScript® 2024 (15th ed.) [Електронний ресурс] – Official standard. – URL: https://ecma-international.org/publications-and-standards/standards/ecma-262/.

16. Express. *Express – Fast, unopinionated, minimalist web framework for Node.js* [Електронний ресурс] – Офіційний сайт. – URL: https://expressjs.com/.

17. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures [Електронний ресурс] / PhD Dissertation, Univ. of California,

Irvine, 2000. – 293 с. – Режим доступу: https://ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf.

18. Goosen L., Ajibola S. Information Systems Architecture and Technology Security Aspects Relating to the Usability Attributes and Evaluation Methods of Mobile Commerce Websites // *Inf. Syst. Archit. Technol.: Proc. of 40th Int. Conf. ISAT 2019*. – Singapore: Springer, 2019. – P. 328–337. – DOI: 10.1007/978-981-15-4042-4_31link.springer.com.

19. GraphQL. *GraphQL – A query language for your API* [Електронний ресурс] – Офіційний сайт. – URL: https://graphql.org/.

20. gRPC. *gRPC – A high-performance RPC framework* [Електронний ресурс] – Офіційний сайт. – URL: https://grpc.io/.

21. GatsbyJS. *Gatsby – The React-based framework for static sites* [Електронний ресурс] – Офіційний сайт. – URL: https://www.gatsbyjs.com/.

22. Hackfort S. Patterns of Inequalities in Digital Agriculture: A Systematic Literature Review // *Sustainability*. – 2021. – Vol. 13, No. 22, Art. 12345. – DOI: 10.3390/su132212345.

23. IBM. What is smart farming? [Електронний ресурс] – IBM Think (2023). – Режим доступу: https://www.ibm.com/think/topics/smart-farming.

24. Jenkins. *Jenkins – The leading open source automation server* [Електронний ресурс] – Офіційний сайт. – URL: https://www.jenkins.io/.

25. Kafka (Apache). *Apache Kafka – Open-source distributed event streaming platform* [Електронний ресурс] – Офіційний сайт. – URL: https://kafka.apache.org/.

26. Laravel. *Laravel – The PHP framework for web artisans* [Електронний ресурс] – Офіційний сайт. – URL: https://laravel.com/.

27. Material Design. *Material Design* [Електронний ресурс] – Official Google guidelines. – URL: https://m2.material.io/design/guidelines-overview.

28. McFadden J., Casalini F., Griffin T., Antón J. The digitalisation of agriculture: A literature review and emerging policy issues [Електронний ресурс]

// *OECD Food, Agriculture and Fisheries Papers*, No. 176. – Paris: OECD Publishing, 2022. – DOI: 10.1787/285cc27d-en.

29. Medennikov V., Raikov A. Formation of the Digital Platform for Precision Farming with Mathematical Modeling[Електронний ресурс] // *Proc. CEUR Workshop on Civil Eng. 4.0*. – 2020. – Vol. 2790, P. 121–131. – Режим доступу: https://ceur-ws.org/Vol-2790/paper12.pdf:contentReference[oaicite:11]{index=11}.

30. MUI. *MUI – The React component library you always wanted* [Електронний ресурс] – Офіційний сайт. – URL: https://mui.com/.

31. Nielsen J. 10 Usability Heuristics for User Interface Design [Електронний ресурс] – Nielsen Norman Group, 2024. – Режим доступу: https://www.nngroup.com/articles/ten-usability-heuristics/.

32. Node.js. *Node.js – JavaScript runtime built on Chrome's V8 engine* [Електронний ресурс] – Офіційний сайт. – URL: https://nodejs.org/.

33. OWASP Foundation. *OWASP Top Ten – Awareness of most critical web app security risks* [Електронний ресурс] – Офіційний сайт. – URL: https://owasp.org/www-project-top-ten/.

34. PostgreSQL Global Dev. Group. *PostgreSQL Documentation* [Електронний ресурс] – Official docs (intro to PostgreSQL). – Режим доступу: https://www.postgresql.org/docs/current/intro-whatis.html.

35. Prisma. *Prisma – Next-generation Node.js and TypeScript ORM* [Електронний ресурс] – Офіційний сайт. – URL: https://www.prisma.io/docs.

36. Purdue University. An open source framework for digital agriculture [Електронний ресурс] / Rick Gustafson (Purdue ECE News, Sept. 2022). – Режим доступу: https://engineering.purdue.edu/ECE/News/2022/an-open-source-framework-for-digital-agriculture.

37. React. *React – A JavaScript library for building user interfaces* [Електронний ресурс] – Офіційний сайт. – URL: https://react.dev/.

38. Ruby on Rails. *Ruby on Rails – A web-application framework* [Електронний ресурс] – Офіційний сайт. – URL: https://rubyonrails.org/.

39. Shneiderman B. The Eight Golden Rules of Interface Design [Електронний ресурс] – University of Maryland, 2020. – Режим доступу: http://www.cs.umd.edu/~ben/goldenrules.html.

40. Spring. *Spring Framework – Build Enterprise Java™ applications* [Електронний ресурс] – Офіційний сайт. – URL: https://spring.io/.

41. Tailwind CSS. *Tailwind CSS – Utility-first CSS framework for rapid UI development* [Електронний ресурс] – Офіційний сайт. – URL: https://tailwindcss.com/:contentReference[oaicite:21]{index=21}.

42. TypeScript. *TypeScript – JavaScript with syntax for types* [Електронний ресурс] – Офіційний сайт. – URL: https://www.typescriptlang.org/.

43. Vue.js. *Vue.js – The Progressive JavaScript Framework* [Електронний ресурс] – Офіційний сайт. – URL: https://vuejs.org/.

44. W3C (World Wide Web Consortium). Web Content Accessibility Guidelines (WCAG) 2.1 [Електронний ресурс] – Офіційний сайт W3C. – Режим доступу: https://www.w3.org/TR/WCAG21/.

45. Wang B., Dong H. Research on farmers' agricultural digital service use behavior under the rural revitalization strategy—Based on the extended technology acceptance model // *Front. Environ. Sci.* – 2023. – Vol. 11: Article 1180072. – DOI: 10.3389/fenvs.2023.1180072.

46. Yuan Y., Sun Y. Practices, Challenges, and Future of Digital Transformation in Smallholder Agriculture: Insights from a Literature Review // *Agriculture*. – 2024. – Vol. 14, Art. 2193. – DOI: 10.3390/agriculture14122193.

47. Zhang R., Zhu H., Chang Q., Mao Q. A Comprehensive Review of Digital Twins Technology in Agriculture // *Agriculture*. – 2025. – Vol. 15, Art. 903. – DOI: 10.3390/agriculture15090903

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка інтернет-платформи для садівників-початківців

Виконав студент 4 курсу
Групи ПД-43
Шевченко Ілля Сергійович
Керівник роботи
к.т.н, доцент кафедри ІПЗ Довженко Тимур Павлович

Київ – 2025

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи: спрощення доступу до агрономічної інформації розробити інтернет-платформу для садівників-початківців

Об’єкт дослідження: інтернет-платформа для садівників-початківців, призначена для взаємодії користувачів та спрощення доступу до агрономічної інформації.

Предмет дослідження: засоби та методи розробки інтернет-платформ для обміну агрономічною інформацією.

ЗАДАЧІ КВАЛІФІКАЦІЙНОЇ РОБОТИ

1.Провести аналіз вимог до веб-платформи GrowSpace та визначити основні функціональні компоненти: керування лотами, кошик і оформлення замовлення.
2.Спроектувати та реалізувати клієнт-серверну архітектуру із Next.js, Tailwind CSS та Prisma для реєстрації/авторизації користувачів, CRUD-операцій над рослинами та API-методів кошика.
3.Розробити адаптивний інтерфейс користувача: перегляд карток рослин із фото й описом, регулювання кількості в кошику, форму введення контактних даних при оформленні замовлення і провести функціональне тестування.
4.Реалізувати систему сповіщень для продавців про надходження нових замовлень у особистому кабінеті та перевірити її роботу.

3

АНАЛІЗ АНАЛОГІВ

Технічні характеристики	Planta	Gardenize	AgroMarket	Prom.ua	GrowSpace
Доступ до знань	+	+	-	-	+
Спільнота	-	+	-	-	+
Маркетплейс	-	-	+	+	+
Персоналізація	+	+	-	-	+
Інтуїтивний інтерфейс (UI/UX)	+	-	-	-	+

4

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Функціональні вимоги:

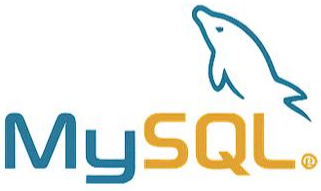
1. Реєстрація та авторизація користувачів з перевіркою даних.
2. Можливість додавання рослин з описом, технологією вирощування, зображенням, ціною та контактами продавця.
3. Перегляд списку рослин з фільтрацією та пошуком.
4. Реалізація особистого кабінету користувача.
5. Можливість взаємодії між користувачами в спільних діях.

Нефункціональні вимоги:

1. Інтуїтивно зрозумілий мінімалістичний інтерфейс із коректним відображенням на десктопі та мобільних пристроях, що працює у всіх актуальних версіях Chrome, Firefox, Safari і Edge.
2. Швидка реакція та завантаження сторінок: TTFB < 200 мс, Time-to-Interactive < 1 с при 1000 лотах, з архітектурою, яка дозволяє легко розширювати функціонал.
3. Захист персональних даних і паролів (шифрування, безпечне зберігання), цілісність і зв'язність записів у базі даних та механізми резервного копіювання.

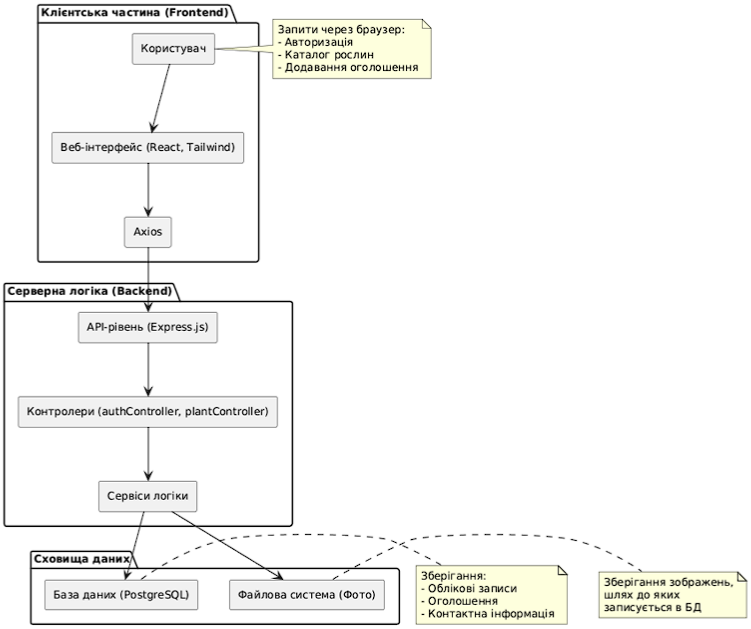
5

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ



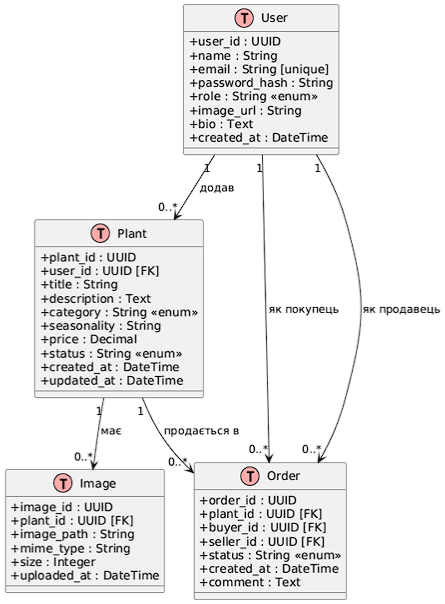
6

СИСТЕМНА АРХІТЕКТУРА ВЕБ-ДОДАТКУ



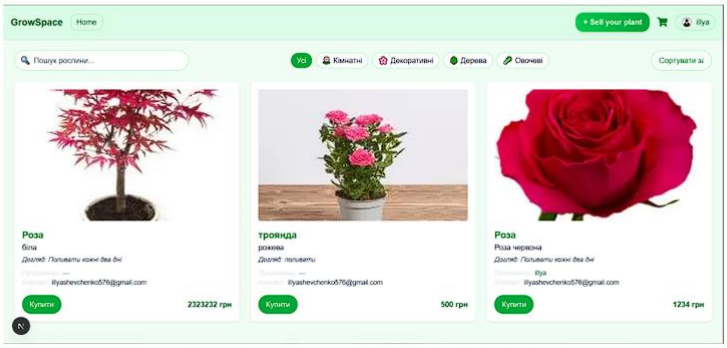
7

ОПИС СТРУКТУРИ БАЗИ ДАНИХ

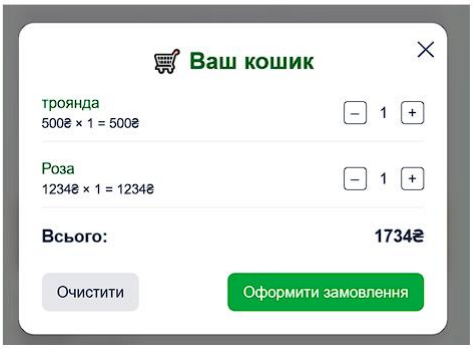


8

ЕКРАННІ ФОРМИ



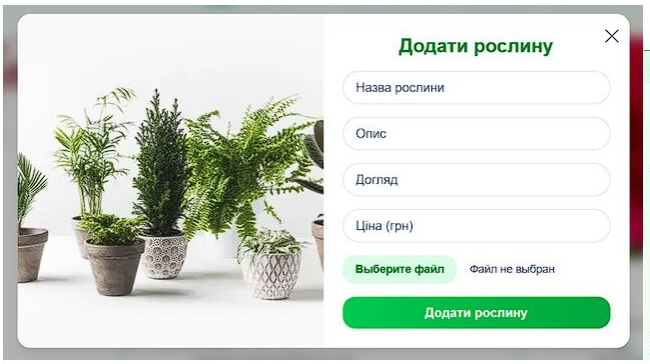
Головна сторінка



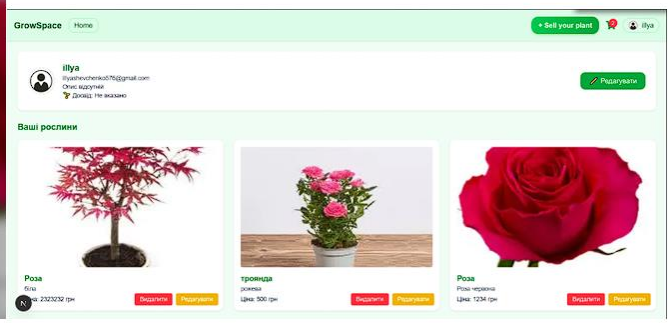
Корзина

9

ЕКРАННІ ФОРМИ



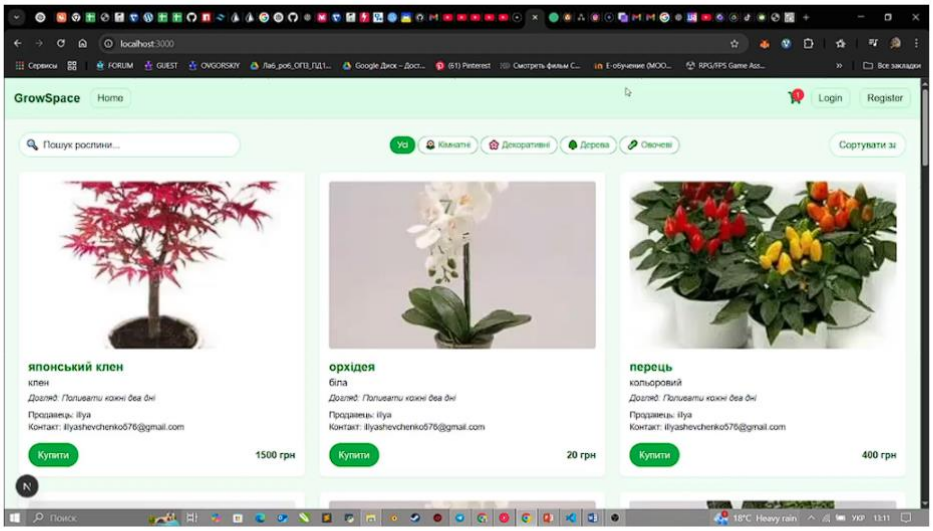
Модальне вікно додавання рослини на продаж



Профіль користувача

10

ДЕМОНСТРАЦІЯ РОБОТИ ПРОГРАМИ



АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Шевченко І.С., Шахматов І.О. GrowSpace: розробка інтелектуального веб- додатку для підтримки спільноти садівників. Кафедра штучного інтелекту. V Всеукраїнській науково- практичній конференції «Сучасні інтелектуальні інформаційні технології в науці та освіті». 15.05.25, ДУІКТ, м.Київ (подано до друку)

2. Шевченко І.С., Шахматов І.О. GrowSpace — інтелектуальна платформа для спільноти садівників: створення, функціональність та користувацький досвід. Кафедра штучного інтелекту. V Всеукраїнській науково-практичній конференції «Сучасні інтелектуальні інформаційні технології в науці та освіті». 15.05.25, ДУІКТ, м.Київ (подано до друку)

ВИСНОВКИ

1. Проаналізовано вимоги до веб-платформи GrowSpace та визначено ключові функціональні компоненти — реєстрація/авторизація, керування лотами, кошик і оформлення замовлення, що дало змогу сформувати чітке технічне завдання й коректно спроектувати структуру бази даних.
2. Спроектовано та реалізовано клієнт-серверну архітектуру на базі Next.js, Tailwind CSS і Prisma: створено надійні API для CRUD-операцій над рослинами та керування кошиком, побудовано механізми захищеної обробки користувацьких даних і забезпечено високу продуктивність і масштабованість системи.
3. Розроблено адаптивний інтерфейс користувача із динамічними картками рослин (фото, опис догляду, ціна), регулюванням кількості в кошику та формою оформлення замовлення з валідацією полів; проведено функціональне тестування всіх основних сценаріїв, що підтвердило коректність роботи, легко-зрозумілість інтерфейсу й відсутність критичних помилок.
4. У результаті реалізації системи сповіщень у особистому кабінеті продавця повідомлення про нові замовлення доставляються миттєво, що забезпечило оперативність реагування на запити покупців і підвищило якість обслуговування.

ДОДАТОК Б. ЛІСТИНГИ ПРОГРАМНИХ МОДУЛІВ

Вихідний код файлу page.jsx

```
"use client";

import Image from "next/image";
import { useState, useEffect } from "react";
import SearchBar      from "./components/SearchBar";
import CategoryFilter from
"./components/CategoryFilter";
import SortSelect     from "./components/SortSelect";
import { useCart }    from "./context/CartContext";

export default function Home() {
  // дані
  const [plants, setPlants] = useState([]);
  const [user,   setUser]   = useState(null);

  // кошик
  const { addToCart } = useCart();

  // фільтри / сортування
  const [searchTerm, setSearchTerm] = useState("");
  const [sortOption, setSortOption] = useState("");
  const [category,   setCategory]   = useState("");

  const categories = [
    { value: "",          label: "Усі" },
    { value: "кімнатні",   label: "✿ Кімнатні" },
    { value: "декоративні",label: "❁ Декоративні" },
    { value: "дерева",     label: "🌲 Дерева" },
    { value: "овочі",      label: "□ Овочеві" },
  ];

  // завантаження первинних даних
  useEffect(() => {
    fetch("/api/plants")
      .then(r => r.json())
      .then(data => Array.isArray(data) ? setPlants(data) :
setPlants([]));

    fetch("/api/me")
      .then(r => r.json())
      .then(d => setUser(d.user))
      .catch(() => setUser(null));
  }, []);

  // фільтрація + сортування
  const filtered = plants
    .filter(p =>
p.name.toLowerCase().includes(searchTerm.toLowerCas
e()))
    .filter(p => !category || p.category === category);

  const sorted = [...filtered].sort((a, b) => {
    if (sortOption === "priceAsc")  return a.price -
b.price;
    if (sortOption === "priceDesc") return b.price -
a.price;
    if (sortOption === "nameAsc")   return
a.name.localeCompare(b.name, "uk");
    if (sortOption === "nameDesc")  return
b.name.localeCompare(a.name, "uk");
    return 0;
  });

  return (
    <main className="p-6 bg-green-50 min-h-screen
space-y-6">
      {/* Toolbar з пошуком, фільтром та сортуванням
*/}
      <div className="flex items-center gap-4">
        {/* Пошук */}
        <SearchBar
          value={searchTerm}
          onChange={setSearchTerm}
          className="w-1/4"
        />

        {/* Фільтр категорій */}
        <CategoryFilter
          categories={categories}
          selected={category}
          onChange={setCategory}
          className="flex-auto flex justify-center"
        />

        {/* Сортування */}
        <SortSelect
          value={sortOption}
          onChange={setSortOption}
          className="w-32"
        />
      </div>

      {/* Сітка карток рослин */}
      <div className="grid gap-6 sm:grid-cols-2 md:grid-
cols-3">
        {sorted.map(plant => (
          <div
            key={plant.id}
            className="bg-white p-4 rounded-lg shadow
hover:shadow-lg transition"
          >
            <div className="relative w-full aspect-[16/10]
mb-4 overflow-hidden rounded">
              <Image
                src={plant.photo}
                alt={plant.name}
                fill
                className="object-cover"
```

```
          />
        </div>

        <h3 className="text-xl font-bold text-green-
700">{plant.name}</h3>
        <p className="text-gray-
800">{plant.description}</p>
        <p className="text-sm italic mt-1 text-gray-
600">
          Догляд: {plant.care}
        </p>

        {/* Продавець */}
        <div className="mt-2 text-sm text-gray-800">
          Продавець: <span className="font-
medium">{plant.seller?.name || "—"}</span>
        </div>
        {/* Контакт */}
        <div className="text-sm text-gray-800 mb-2">
          Контакт: <span className="font-
medium">{plant.seller?.email || plant.phone}</span>
        </div>

        <div className="flex justify-between items-
center mt-4">
          <button
            onClick={() =>
              addToCart({
                id:       plant.id,
                name:     plant.name,
                price:    plant.price,
                quantity:  1,
                sellerId:  plant.sellerId,
              })
            }
            className="bg-green-600 text-white rounded-
full px-4 py-2 hover:bg-green-700 transition"
          >
            Купити
          </button>
          <span className="font-semibold text-green-
800">
            {plant.price} грн
          </span>
        </div>
      </div>
    ))}
  </div>
 </main>
 );
}
```

Вихідний код файлу api/plants/[id]/route.js

```
// app/api/plants/[id]/route.js
import { NextResponse } from "next/server";
import prisma from "@/lib/prisma";

export async function DELETE(request, { params }) {
 // params — це асинхронний об’єкт, тому чекаємо
його перед використанням
 const { id } = await params;

 try {
  await prisma.plant.delete({
   where: { id: Number(id) },
  });
  return NextResponse.json(
   { message: "Рослину успішно видалено" },
   { status: 200 }
  );
 } catch (err) {
  console.error("DELETE /api/plants/[id] error:", err);
  return NextResponse.json({ error: err.message }, {
status: 500 });
 }
}
```

Вихідний код файлу `app/components/Header.jsx`

```
// app/components/Header.jsx
"use client";

import Link from "next/link";
import { useEffect, useState, useRef } from "react";
import { FaShoppingCart } from "react-icons/fa";

import LoginModal from "./LoginModal";
import AddPlantModal from "./AddPlantModal";
import CartModal from "./CartModal";
import BellNotificationButton from
"./BellNotificationButton"; // ← правильний шлях
import { useCart } from "../context/CartContext";

export default function Header() {
 const [user, setUser] = useState(null);
 const [menuOpen, setMenuOpen] = useState(false);
 const [showAuthModal, setShowAuthModal] =
useState(false);
 const [isRegistering, setIsRegistering] = useState(false);
 const [showAddModal, setShowAddModal] =
useState(false);
 const [cartOpen, setCartOpen] = useState(false);

 const { items, changeQuantity, total } = useCart();

 const profileRef = useRef(null);
 let closeTimer;

 // Завантажуємо поточного користувача
 useEffect(() => {
  fetch("/api/me")
   .then(res => res.json())
   .then(d => setUser(d.user))
   .catch(() => setUser(null));
 }, []);

 const handleLogout = async () => {
  await fetch("/api/logout", { method: "POST" });
  setUser(null);
  window.location.href = "/";
```

```
  };

  const handleMouseEnter = () => {
    clearTimeout(closeTimer);
    setMenuOpen(true);
  };
  const handleMouseLeave = () => {
    closeTimer = setTimeout(() => setMenuOpen(false),
200);
  };

  return (
    <>
      <header className="flex items-center justify-
between p-4 bg-green-100 shadow relative">
        <div className="flex items-center space-x-4">
          <h1 className="text-xl font-bold text-green-
800">GrowSpace</h1>
          <Link
            href="/"
            className="text-green-800 border border-green-
300 rounded-xl px-3 py-1 hover:bg-green-200 transition"
          >
            Home
          </Link>
        </div>

        <div className="flex items-center space-x-4">
          {user && (
            <button
              onClick={() => setShowAddModal(true)}
              className="bg-gradient-to-r from-green-500
to-green-600 hover:from-green-600 hover:to-green-700
text-white rounded-2xl px-4 py-2 font-semibold shadow-
md transition"
            >
              + Sell your plant
            </button>
          )}

          {user && (
            <BellNotificationButton />
          )}

          <button
            onClick={() => setCartOpen(true)}
            className="relative text-green-700 hover:text-
green-800"
          >
            <FaShoppingCart size={22} />
            {items.length > 0 && (
              <span className="absolute -top-1 -right-1 bg-
red-500 text-white text-xs w-5 h-5 flex items-center
justify-center rounded-full">
                {items.reduce((sum, item) => sum +
item.quantity, 0)}
              </span>
            )}
          </button>

          {user ? (
            <div
              ref={profileRef}
              onMouseEnter={handleMouseEnter}
              onMouseLeave={handleMouseLeave}
              className="relative"
            >
              <button className="flex items-center space-x-
2 text-green-800 border border-green-300 rounded-full
px-3 py-1 hover:bg-green-200 transition">
                <img
                  src="/default-avatar.png"
                  alt="avatar"
                  className="w-6 h-6 rounded-full"
                />
                <span className="font-
medium">{user.name}</span>
              </button>

              {menuOpen && (
                <div className="absolute right-0 mt-2 w-40
bg-white rounded-xl shadow-lg py-2 z-50">
                  <Link
                    href="/profile"
                    className="block px-4 py-2 text-sm text-
green-800 hover:bg-green-100"
                  >
                    Профіль
                  </Link>
                  <button
                    onClick={handleLogout}
                    className="w-full text-left px-4 py-2 text-
sm text-red-600 hover:bg-red-50"
                  >
                    Вийти
                  </button>
                </div>
              )}
            </div>
          ) : (
            <>
              <button
                onClick={() => {
                  setIsRegistering(false);
                  setShowAuthModal(true);
                }}
                className="text-green-800 border border-
green-300 rounded-xl px-3 py-1 hover:bg-green-200
transition"
              >
                Login
              </button>
              <button
                onClick={() => {
                  setIsRegistering(true);
                  setShowAuthModal(true);
                }}
                className="text-green-800 border border-
green-300 rounded-xl px-3 py-1 hover:bg-green-200
transition"
              >
                Register
```

```
            </button>
          </>
        )}
      </div>
    </header>

    {showAuthModal && (
      <LoginModal
        onClose={() => setShowAuthModal(false)}
        isRegistering={isRegistering}
        switchMode={() => setIsRegistering(prev =>
!prev)}
        onLoginSuccess={() => {
          setShowAuthModal(false);
          fetch("/api/me")
            .then(res => res.json())
            .then(d => setUser(d.user));
        }}
      />
    )}

    {showAddModal && (
      <AddPlantModal
        onClose={() => setShowAddModal(false)}
        onPlantAdded={() => window.location.reload()}
      />
    )}

    {cartOpen && (
      <CartModal
        items={items}
        onQuantityChange={changeQuantity}
        total={total}
        onClose={() => setCartOpen(false)}
      />
    )}
  </>
 );
}
```

Вихідний код файлу app/profile/page.jsx

```
// app/profile/page.jsx
'use client';

import Image from 'next/image';
import { useEffect, useState } from 'react';
import EditProfileModal from
'../components/EditProfileModal';

export default function ProfilePage() {
  const [user, setUser] = useState(null);
  const [plants, setPlants] = useState([]);
  const [showEditModal, setShowEditModal] =
useState(false);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState('');

  // Завантажити інформацію про користувача
  useEffect(() => {
    const fetchUser = async () => {
      try {
        const res = await fetch('/api/me');
        if (!res.ok) throw new Error('Не вдалося отримати
профіль');
        const { user: dataUser } = await res.json();
        setUser(dataUser);
      } catch (err) {
        console.error('fetchUser error:', err);
        setError('Не вдалося завантажити дані
користувача');
      } finally {
        setLoading(false);
      }
    };
    fetchUser();
  }, []);

  // Після успішного завантаження користувача —
завантажити тільки його рослини
  useEffect(() => {
    if (!user) return;
    const fetchUserPlants = async () => {
      try {
        const res = await fetch('/api/my-plants');
        if (!res.ok) throw new Error('Не вдалося отримати
ваші рослини');
        const data = await res.json();
        setPlants(Array.isArray(data) ? data : []);
      } catch (err) {
        console.error('fetchUserPlants error:', err);
        setError('Не вдалося завантажити ваші рослини');
      }
    };
    fetchUserPlants();
  }, [user]);

  const deletePlant = async (id) => {
    const confirmed = confirm('Ви впевнені, що хочете
видалити цю рослину?');
    if (!confirmed) return;
    try {
      const res = await fetch(`/api/plants/${id}`, { method:
'DELETE' });
      if (!res.ok) throw new Error('Не вдалося видалити
рослину');
      setPlants(prev => prev.filter(p => p.id !== id));
    } catch (err) {
      console.error('deletePlant error:', err);
      setError('Помилка при видаленні');
    }
  };

  if (loading) {
    return <p className="p-6">Завантаження...</p>;
  }

  return (
    <main className="p-6 bg-green-50 min-h-screen">
```

```
    {error && (
      <div className="bg-red-100 text-red-700 p-4
rounded mb-4">
        {error}
      </div>
    )}

    <div className="bg-white rounded-xl shadow p-6
flex justify-between items-center mb-6">
      <div className="flex items-center gap-4">
        <div className="w-16 h-16 rounded-full
overflow-hidden border">
          <Image
            src={user?.avatar || '/default-avatar.png'}
            alt="Аватар"
            width={64}
            height={64}
            className="object-cover w-full h-full"
          />
        </div>
        <div>
          <h2 className="text-xl font-bold text-green-
800">{user?.name}</h2>
          <p className="text-sm text-gray-
600">{user?.email}</p>
          <p className="text-sm text-gray-
700">{user?.bio || 'Опис відсутній'}</p>
          <p className="text-sm text-gray-700">🌱
Досвід: {user?.experience || 'Не вказано'}</p>
        </div>
      </div>
      <button
        onClick={() => setShowEditModal(true)}
        className="bg-green-600 hover:bg-green-700
text-white px-5 py-2 rounded-xl flex items-center gap-2"
      >
        ✏️ Редагувати
      </button>
    </div>

    <h2 className="text-xl font-bold text-green-700
mb-4">Ваші рослини</h2>

    {plants.length === 0 ? (
      <p className="text-gray-600">У вас ще немає
рослин</p>
    ) : (
      <div className="grid gap-6 sm:grid-cols-2
md:grid-cols-3">
        {plants.map(plant => (
          <div key={plant.id} className="bg-white
rounded-xl shadow p-4 relative">
            <div className="relative w-full aspect-[16/10]
mb-3 rounded overflow-hidden">
              <Image src={plant.photo} alt={plant.name}
fill className="object-cover" />
            </div>
            <h3 className="text-green-700 font-semibold
text-lg">{plant.name}</h3>
            <p className="text-sm text-gray-
600">{plant.description}</p>
            <p className="text-sm mt-1 text-gray-800
font-medium">Ціна: {plant.price} грн</p>

            <div className="absolute right-3 bottom-3 flex
gap-2">
              <button
                onClick={() => deletePlant(plant.id)}
                className="bg-red-500 text-white text-sm
px-3 py-1 rounded hover:bg-red-600"
              >
                Видалити
              </button>
              <button className="bg-yellow-500 text-white
text-sm px-3 py-1 rounded hover:bg-yellow-600">
                Редагувати
              </button>
            </div>
          </div>
        ))}
      </div>
    )}

    {showEditModal && <EditProfileModal
onClose={() => setShowEditModal(false)} />}
  </main>
 );
}
```