Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)
Веб-застосунок для ідентифікації та підбору кімнатно-садових рослин
(тема)

Виконав:
здобувач 4 року навчання,
групи ПЗПІ-21-10

Артем ПЕЧЕНЕВСЬКИЙ
(власне ім'я, прізвище)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник доц. Роксана МЕЛЬНІКОВА
(посада, власне ім'я, прізвище)

Допускається до захисту
Зав. кафедри _____ Кирило СМЕЛЯКОВ
(підпис) (власне імя, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
Кафедра програмної інженерії
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва)
Тип програми освітньо-професійна
Освітня програма програмна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)
«___» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Печеневському Артему Артемовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Веб-застосунок для ідентифікації та підбору кімнатно-садових рослин
Затверджена наказом по університету від 19.05.2025р. № 397Ст
2. Термін подання студентом роботи до екзаменаційної комісії 26.06.2025
3. Вихідні дані до роботи Розробити клієнтську та серверну частини веб-застосунку для ідентифікації та підбору кімнатно-садових рослин, що надає користувачам можливість визначати рослини за зображенням.
4. Перелік питань, що потрібно опрацювати в роботі Вступ, аналіз предметної галузі, формування вимог до програмної системи, архітектура та проектування програмного забезпечення, опис прийнятих програмних рішень, тестування розробленого програмного забезпечення, висновки.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	09.04.2025	*виконано*
2	Створення специфікації ПЗ	14.04.2025	*виконано*
3	Проектування ПЗ	29.04.2025	*виконано*
4	Розробка ПЗ	12.05.2025	*виконано*
5	Тестування ПЗ	20.05.2025	*виконано*
6	Оформлення пояснювальної записки	27.05.2025	*виконано*
7	Підготовка презентації та доповіді	05.06.2025	*виконано*
9	Попередній захист	25.06.2025	*виконано*
9	Нормоконтроль, рецензування	25.06.2025	*виконано*
10	Здача роботи у електронний архів	26.06.2025	*виконано*
11	Допуск до захисту у зав. кафедри	26.06.2025	*виконано*

Дата видачі завдання 9 квітня 2025 р.

Здобувач _____

(підпис)

Керівник роботи _____ доц. Роксана МЕЛЬНІКОВА

(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи бакалавра, 51 сторінок, 7 рисунків, 10 джерел.

РОЗПІЗНАВАННЯ РОСЛИН, ВЕБ-ДОДАТОК, АВТЕНТИФІКАЦІЯ, API, SUPABASE, NODE.JS, PLANT.ID, JAVASCRIPT.

Об’єктом дослідження є веб-застосунок у вигляді веб-сервісу для розпізнавання об’єктів за зображеннями, зокрема рослин, та забезпечення персоналізованої взаємодії користувачів із системою. У сучасному світі все більше людей цікавляться ідентифікацією рослин — для догляду, навчання чи досліджень. Водночас, існуючі сервіси часто вимагають платної підписки або складні у використанні.

Метою роботи є створення веб-застосунку у вигляді простої у користуванні та безпечної вебсистеми для розпізнавання кімнатних і садових рослин за зображенням. Основними задачами стали: інтеграція з API для машинного розпізнавання (Plant.id), реалізація зручного інтерфейсу користувача на фронтенді, створення серверної частини, яка приховує ключі та перевіряє права доступу, а також забезпечення повноцінної авторизації користувачів.

У процесі розробки використано HTML, CSS, JavaScript для фронтенду, Node.js з Express для бекенду та Supabase для бази даних і автентифікації. Для розпізнавання рослин застосовано API Plant.id, а всі запити обробляються захищеним сервером.

У результаті створено вебзастосунок, що дозволяє зареєстрованим користувачам завантажувати фото рослин, отримувати результати, зберігати їх і переглядати історію. Система демонструє можливості сучасної веброзробки з використанням хмарних сервісів і відкритих API.

ABSTRACT

PLANT IDENTIFICATION, WEB APPLICATION, AUTHENTICATION, API, SUPABASE, NODE.JS, PLANT.ID, JAVASCRIPT.

The object of the study is web services for image-based object recognition, particularly plant identification, and providing personalized user interaction with the system. In today's world, more and more people are interested in identifying plants — for care, learning, or research. At the same time, existing services often require paid subscriptions or are too complex to use.

The purpose of this work is to develop a simple and secure web system for identifying indoor and garden plants based on images. The main tasks included: integrating with a machine-learning-based recognition API (Plant.id), implementing a user-friendly frontend interface, building a backend server to hide API keys and verify user access, and ensuring full user authentication.

The project uses HTML, CSS, and JavaScript for the frontend, Node.js with Express for the backend logic, and Supabase as a cloud platform for database and user authentication. For plant identification, the system uses the Plant.id API, which determines the most likely plant species from an image. Authentication is handled through Supabase Auth, and all database operations are routed through a secure backend, separate from the client side.

As a result, a fully functional web application was created that allows registered users to upload plant images, receive identification results, save them to a personal profile, and view their history. The system serves as a demonstration of how open APIs, modern web technologies, and cloud services can be combined to build a useful, accessible, and secure tool.

ЗМІСТ

ВСТУП

Метою цієї кваліфікаційної роботи є створення веб-застосунку, який дозволяє автоматично розпізнавати кімнатні та садові рослини за зображенням, зберігати результати розпізнавання в базі даних і надавати користувачеві персоналізований доступ до власної історії. У розробці проєкту було поєднано сучасні вебтехнології, хмарні сервіси[1; 2; 9] та зовнішні API[3], що дозволило забезпечити зручність використання, захист конфіденційних даних і можливість масштабування в майбутньому.

Веб-застосунок включає механізм автентифікації користувачів із верифікацією електронної пошти[1], можливість завантаження зображень безпосередньо через вебінтерфейс, автоматичне визначення назви рослини за допомогою зовнішнього сервісу Plant.id[3] , а також збереження результатів у базі даних, прив’язаних до конкретного облікового запису. Серверна частина побудована на платформі Node.js[4] з використанням фреймворку Express[6], клієнтський інтерфейс реалізований за допомогою HTML, CSS і JavaScript[7], а роль бази даних і сервісу автентифікації виконує Supabase[1]. Вся взаємодія з Plant.id API відбувається через власний сервер, що дозволяє приховати конфіденційні ключі та контролювати звернення до зовнішнього сервісу.

Система дає змогу користувачам у кілька кліків визначити рослину на фото, автоматично зберегти результат до особистого кабінету та згодом переглядати історію розпізнань. Такий підхід не лише підвищує зручність для кінцевого користувача, а й забезпечує захищену та централізовану обробку даних. Водночас реалізовані механізми безпеки[5] та чітке розділення функцій між фронтендом і бекендом відповідають сучасним вимогам до побудови надійних вебзастосунків.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

Кімнатні та садові рослини традиційно відіграють важливу роль у житті сучасного суспільства. Вирощування квітів та декоративних культур у помешканнях, на присадибних ділянках, балконах і терасах сприяє естетичному оформленню простору, поліпшує мікроклімат та підвищує психологічний комфорт людей._За результатами низки досліджень, наявність живих рослин в інтер'єрі може зменшувати рівень стресу та покращувати загальне самопочуття.

Водночас, щоб вирощувати рослини успішно, потрібно враховувати низку факторів:

— грунтові умови та добрива (тип субстрату, періодичність підживлення);

— освітлення (пряме сонячне світло, розсіяне, затінене приміщення);

— вологість повітря (особливо важливо для тропічних рослин);

— температурний режим (різні види мають оптимальну температуру росту);

— періодичність поливу та спосіб поливу;

— сезонність (деякі рослини особливо вибагливі до циклів зростання та цвітіння).

Через розмаїття видів і сортів, початківцям нелегко визначити, яка рослина підходить саме їм. Також багато проблем виникає з тим, що інформація з догляду за певним видом може бути суперечливою в різних джерелах або неповною. Ба більше, коли людина бачить у когось красиву рослину й хоче придбати собі таку ж, виникає потреба ідентифікувати її: назву, групу, оптимальні умови, можливі труднощі.

Які є сучасні джерела інформації про рослини?

З розвитком інтернет-технологій значна частина інформації про рослини стала доступною в мережі:

— веб-сайти та блоги (наприклад, Plantdecorshop.com, Roslyny.com.ua, Floren.com.ua), де можна знайти короткі описи основних сортів і поради з вирощування.

— мобільні додатки (Greg App, PlantSnap, PictureThis тощо) з функціями розпізнавання рослин за фото.

— соціальні мережі (Instagram, Facebook, TikTok), де любителі рослин діляться досвідом, фотографіями й короткими порадами.

— форуми та групи за інтересами: такі спільноти часто живуть у Facebook, Reddit чи спеціалізованих сайтах, на яких учасники ставлять питання, обмінюються досвідом, викладають свої історії або проблеми.

Попри велику кількість ресурсів, інформація нерідко виявляється:

— недостатньо структурованою (важко знайти конкретну пораду саме для свого сорту).

— взаємно суперечливою: різні автори можуть по-різному тлумачити температуру поливу чи потребу в добривах.

— перевантаженою зайвою рекламою та комерційними пропозиціями.

Далі напишу про популярні рішення та їх недоліки.

Plantdecorshop.com: — це інтернет-магазин кімнатно-садових рослин із вбудованим розділом базових статей про догляд. Користувач може одразу переглянути каталог рослин, підібрати потрібну й оформити покупку в один клік, а також ознайомитися з порадами щодо поливу, освітлення та пересаджування. Перевага — можна одразу купити рослину, отримати першу консультацію. Недолік — відсутній розділ активного спілкування з іншими користувачами та зворотний зв’язок, немає автоматичного визначення видів за фото. Приклад на рисунку 1.1.

Рисунок 1.1 – Головна сторінка plantdecorshop.com (виконано самостійно)

Roslyny.com.ua також фокусується на онлайн-продажі кімнатних та садових рослин, доповнюючи каталог мінімальними текстовими описами кожного виду. Сайт містить фільтрацію за категоріями (квіти, сукуленти, деревовидні), проте його інформаційна база відображає лише загальні рекомендації щодо догляду без урахування індивідуальних умов користувача.Недолік: здебільшого статична інформація, складно знайти рекомендації під конкретні умови користувача (температура, клімат регіону). Приклад на рисунку 1.2.

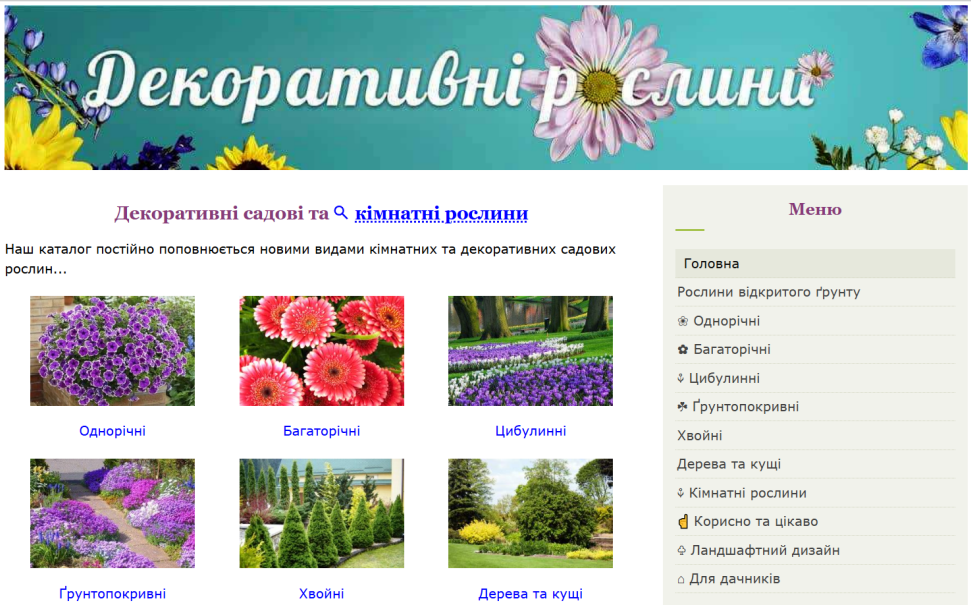

Рисунок 1.2 — Головна сторінка roslyny.com.ua (виконано самостійно)

Floren.com.ua: крамниця та каталог зі стислими характеристиками рослин. Часто оновлюється асортимент, є контакт із продавцем, але відсутня інформація про догляд та зв’язок зі спільнотою. Приклад на рисунку 1.3.

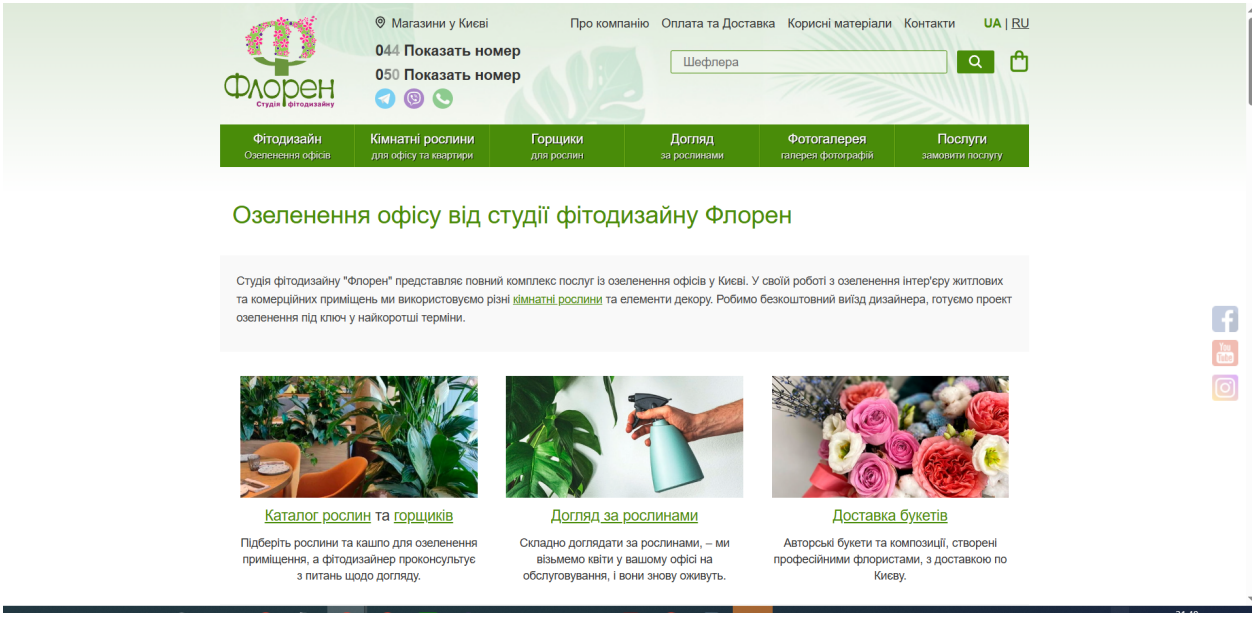

Рисунок 1.3 — Головна сторінка Floren.com.ua (виконано самостійно)

Greg App: це мобільний застосунок, який пропонує систему нагадувань про полив, створення “колекції” власних рослин. Має зручний інтерфейс, але обмежений перелік видів рослин, до того ж відсутні деякі локальні українські сорти. Приклад на рисунку 1.4.

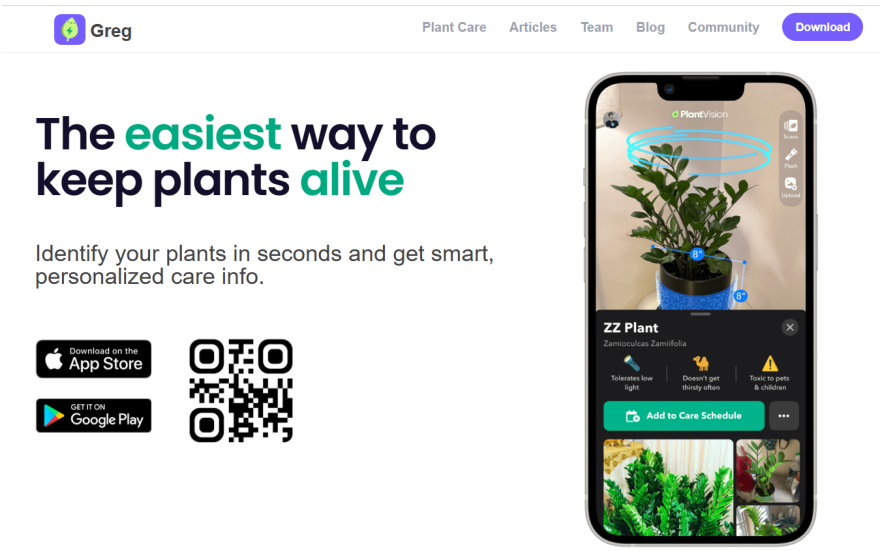

Рисунок 1.4 — Головна сторінка Greg App (виконано самостійно)

Загальні форуми / групи (наприклад, у Facebook): перевага — миттєвий обмін досвідом, іноді можна отримати швидку й корисну пораду. Недолік — інформація хаотична, одна людина пише “поливайте щодня”, інша каже “поливати раз на тиждень”. Нема гарантії кваліфікації того, хто відповідає.

Звідси випливає, що користувачам потрібен комплексний підхід: ідентифікація рослин + рекомендації + динамічний обмін досвідом та проблемами. Саме цей підхід стане концептуальною основою майбутнього веб-застосунку.

Практична цінність і актуальність. У сучасних міських умовах люди прагнуть покращити свої помешкання, наблизитися до природи — ідея “міських джунглів” набуває дедалі більшої популярності. Вирощування рослин стає не просто хобі, а способом життя і розслаблення. Статистика свідчить, що продажі кімнатних рослин зросли в середньому на 30% за останні 5 років (за даними онлайн-маркетплейсів). Отже, збільшується кількість людей, яким потрібна якісна та персоналізована інформація.

Недостатність існуючих рішень (відсутність адекватної персоналізації, розпізнавання на базі великих баз даних, а також локально орієнтованих порад) створює нішеву можливість для нового веб-застосунку. Він може задовольнити потреби різних категорій користувачів: від початківців, які щойно придбали свою першу рослину, до досвідчених садівників, яким потрібні специфічні рекомендації або можливість швидко знаходити колег для обміну рідкісними сортами.

1.2 Виявлення та вирішення проблем

Сучасні користувачі, що цікавляться кімнатними чи садовими рослинами, нерідко стикаються з низкою труднощів, які суттєво ускладнюють процес догляду. Згідно з аналізом популярних тематичних форумів, мобільних

застосунків і онлайн-ресурсів, можна виокремити кілька поширених сценаріїв, у яких звичайний користувач потребує допомоги.

Однією з основних проблем є незнання точної назви рослини. Часто люди купують рослину в магазині або отримують у подарунок, але не мають уявлення про її вид чи сорт. Без цієї інформації складно підібрати належні умови догляду, а отже — забезпечити правильний режим поливу, освітлення, підживлення чи пересадки. У багатьох випадках це призводить до погіршення стану рослини або навіть її загибелі. Додатково, у відкритих джерелах бракує структурованої й повної інформації про догляд. Щоб з’ясувати, як саме утримувати рослину, користувачам доводиться переглядати десятки сайтів, порівнювати поради й орієнтуватися на власну інтуїцію. При цьому часто зустрічаються суперечливі або занадто загальні рекомендації, які не враховують конкретні умови зростання.

Особливо гострою є проблема своєчасного виявлення хвороб або ознак пошкодження. Візуальні симптоми можуть бути нечіткими або схожими між собою, тому користувачі намагаються шукати відповіді у соціальних мережах або на форумах. Однак отримати оперативну, фахову й точну відповідь там вдається далеко не завжди. Через затримки у відповідях або брак досвіду в інших користувачів, шанси вчасно зреагувати знижуються. У більшості випадків людині доводиться діяти навмання або орієнтуватися на сумнівні поради, що не враховують особливості конкретного виду рослини.

Ще одним суттєвим недоліком сучасних інструментів є відсутність централізованої системи нагадувань про важливі дії: полив, підживлення, пересадку. Деякі додатки пропонують такі функції, однак найчастіше вони реалізовані у вигляді стандартних шаблонів, які не враховують індивідуальних потреб конкретної рослини чи реальних умов у приміщенні. Наприклад, при зниженій температурі потреба у поливі змінюється, але більшість додатків не мають таких адаптивних налаштувань.

Також слід відзначити складність пошуку інформації, коли користувач хоче підібрати рослину під власні умови — наприклад, для затіненого підвіконня, холодного балкону або кімнати з низькою вологістю. Такі параметри не завжди враховуються у звичних онлайн-каталогах, а відсутність гнучкої системи фільтрації змушує користувача витрачати час на неефективний пошук і порівняння.

Незважаючи на велику кількість мобільних застосунків і онлайн-сервісів, жоден із них не вирішує всі ці проблеми комплексно. Навіть ті сервіси, які пропонують ідентифікацію рослин за зображенням, не завжди мають високу точність розпізнавання, особливо за умов поганого освітлення, нестандартного ракурсу чи застарілої бази даних. Більше того, багато таких сервісів орієнтовані на іноземні джерела, де відсутні українські назви або локальні види, що ускладнює розуміння й пошук подальшої інформації. Магазини найчастіше обмежуються короткими описами, фокусуючись на продажу, а форуми, навпаки, пропонують надмірну кількість суперечливих коментарів без структури.

Отже, постає актуальна потреба у створенні єдиного вебзастосунку, який би поєднував найнеобхідніші функції в одному інтерфейсі. Така система має включати інструмент розпізнавання рослин за фотографією, який працює через спеціалізоване API, розширену базу знань з описами, фільтрацією та порадами щодо догляду, а також особистий кабінет користувача з історією розпізнавань. Додатково система повинна надавати можливість збереження персональних записів, відображення індивідуального списку рослин і, за потреби, подальшої інтеграції з іншими модулями: системою нагадувань, автоматичного аналізу стану, рекомендацій тощо. Такий підхід дозволить не лише зробити користування простішим і ефективнішим, а й підвищить рівень залученості спільноти та довіру до платформи як до надійного джерела інформації.

У перспективі реалізація подібного рішення сприятиме формуванню повноцінної цифрової екосистеми, яка поєднуватиме інструменти ідентифікації, обміну знаннями, ведення історії та зворотного зв’язку. Це значно покращить

умови догляду за рослинами для широкого кола користувачів — від новачків до досвідчених садівників.

1.3 Постановка задачі

Мета цієї кваліфікаційної роботи — створити веб-застосунок, який допоможе широкому колу користувачів:

— ідентифікувати кімнатні й садові рослини за допомогою фото або текстового опису.

— отримувати персоналізовані рекомендації з догляду (полив, добрива, температурний режим, освітлення).

— взаємодіяти з іншими любителями й експертами через інтегрований форум чи систему запитань і відповідей.

Для досягнення цієї мети визначені такі задачі:

— розробити архітектуру веб-застосунку з урахуванням масштабованості для зберігання великої бази даних.

— реалізувати функціонал розпізнавання видів рослин на основі машинного навчання чи API зовнішніх сервісів (наприклад, Google Vision API).

— забезпечити зручний UI/UX, включно з формою додавання рослини у “власну колекцію” та функціоналом нагадувань.

— імплементувати систему контенту (базу знань, статті, FAQ) із можливістю пошуку за ключовими словами.

— запустити форум (або чат, або дошку обговорення), де користувачі зможуть ставити питання та отримувати відповіді.

— передбачити регулярні оновлення бази даних: можливість адміністрування та модерування контенту.

Трохи інформації про область застосування та очікуваний результат.

Очікується, що створений веб-застосунок буде цікавим для:

— початківців, які хочуть легко знаходити інформацію про рослини.
— досвідчених садівників, яким потрібен інструмент для систематизації власної колекції та спілкування з колегами.
— продавців або дрібних фермерів, які зможуть використовувати форум для маркетингу своїх саджанців і давати експертні консультації.

Кінцевий результат — комплексна онлайн-платформа, здатна полегшити процес вибору та вирощування рослин, зменшуючи кількість поширених помилок (перелив, неправильно обране місце, некоректне внесення добрив тощо) й забезпечуючи спільноту для обміну знаннями.

Основні функціональні вимоги:
— реєстрація та авторизація користувачів (базовий захист, збереження персональних налаштувань);
— інтегрована система розпізнавання (користувач завантажує фото, у відповідь отримує найімовірніший результат);
— особистий кабінет з “колекцією” рослин, де можна зазначати дати поливу, пересадки та інші події.

Обмеження та припущення:

Для першого прототипу можуть використовуватися зовнішні бібліотеки чи хмарні сервіси для розпізнавання зображень, щоб не витрачати надто багато часу на навчання моделей “з нуля”.

База даних із рослинами на етапі прототипу може містити обмежену кількість видів (наприклад, 200 найпопулярніших), але структура має бути розроблена з урахуванням майбутнього розширення.

Форум у початковій версії може бути спрощеним (тільки текст, фото), без продвинутих модераційних інструментів, які можливо додати в майбутньому.

Для зручності розгортання може застосовуватися архітектура “хмарного” типу (наприклад, AWS, Heroku або інші платформи), залежно від технічних рішень.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

2.1 Огляд частин веб-застосунку

Розроблений веб-застосунок побудований на основі трирівневої архітектури, яка забезпечує логічне розділення функціональності між різними компонентами: клієнтською частиною (фронтендом), серверною частиною (бекендом) і зовнішньою інфраструктурою, зокрема хмарними сервісами. Такий підхід дозволяє досягти високої гнучкості, безпеки, легкості масштабування та спрощення супроводу системи в майбутньому.

Уся логіка взаємодії між користувачем і базою знань розділена на окремі шари. Кожен з них виконує власну роль та несе відповідальність лише за свою частину функціональності. Це дозволяє змінювати або доповнювати один з рівнів без необхідності переписувати всю систему. Наприклад, можлива заміна клієнтського інтерфейсу або зовнішнього API без втручання у серверну частину. До складу системи входять наступні ключові компоненти:

— клієнтська частина (фронтенд) — побудована з використанням базових вебтехнологій: HTML, CSS та JavaScript. Вона виконує роль інтерфейсу між користувачем і функціональністю системи. Через веббраузер користувач взаємодіє з елементами інтерфейсу, такими як форми входу, кнопки завантаження зображень, списки розпізнаних рослин тощо. Уся логіка взаємодії з сервером реалізована за допомогою JavaScript-функцій, які надсилають HTTP-запити до відповідних маршрутів бекенду;

— серверна частина (бекенд) — реалізована на основі платформи Node.js із використанням фреймворку Express.js, що дозволяє швидко створювати легкі та продуктивні вебсервери. Сервер відповідає за обробку запитів, перевірку токенів доступу, взаємодію з базою даних Supabase та виклик зовнішнього API для розпізнавання рослин. Усі запити, що містять конфіденційні дані або вимагають авторизації,

проходять через сервер, що дозволяє централізовано контролювати доступ до функціоналу;

— хмарна платформа Supabase — виконує роль як бази даних, так і провайдера автентифікації. Вона надає функціональність реєстрації користувачів, зберігання облікових даних, генерації токенів доступу (у форматі JWT), а також управління таблицями в базі даних PostgreSQL. Supabase дозволяє швидко реалізувати механізми безпеки без написання власної системи реєстрації чи збереження паролів. Крім того, розробник має доступ до вебінтерфейсу Supabase Studio, де можна переглядати таблиці, журнали запитів, керувати обліковими записами тощо;

— зовнішній сервіс Plant.id — надає API для автоматичного розпізнавання рослин за зображенням. Коли користувач завантажує фото, клієнт передає його на сервер, який у свою чергу формує запит до Plant.id API з необхідними параметрами (мовою, модифікаторами, типом відповіді). У результаті система отримує JSON-відповідь із найімовірнішою назвою рослини, описом, посиланням на статтю та іншими даними. Таким чином, вся логіка розпізнавання делегована зовнішньому сервісу, що дозволяє зосередитись на інтеграції та покращенні взаємодії з користувачем.

Поєднання всіх цих компонентів дозволило створити гнучку, безпечну та ефективну систему, яка виконує повний цикл обробки запитів користувача — від автентифікації до збереження результатів розпізнавання.

2.2 Серверна частина веб-застосунку

Серверна частина розробленого веб-застосунку реалізована з використанням мови програмування JavaScript на базі середовища виконання Node.js, що є популярним і стабільним вибором для створення масштабованих

серверних рішень. Основою для побудови маршрутизації та обробки HTTP-запитів слугує фреймворк Express.js. Це легковаговий, але надзвичайно потужний інструмент, який забезпечує високу гнучкість та дозволяє зручно організувати RESTful API, необхідне для взаємодії між клієнтською частиною й сервером.

Основні функції серверної частини включають приймання запитів від користувача через інтерфейс вебзастосунку, верифікацію прав доступу за допомогою Supabase SDK, обробку отриманих зображень та їх подальшу передачу на зовнішній API-сервіс Plant.id. Відповідь, що містить інформацію про розпізнану рослину, зберігається у базі даних Supabase з прив’язкою до конкретного авторизованого користувача.

Особливу увагу під час розробки було приділено питанням безпеки. Усі чутливі дані, включно з API-ключами сторонніх сервісів і параметрами підключення до бази даних, надійно зберігаються у файлі конфігурації `.env`, що не потрапляє до клієнтського коду та не є публічно доступним. Такий підхід дозволяє уникнути витоку конфіденційної інформації й відповідає сучасним стандартам безпечної веброзробки.

Ключовою перевагою такої архітектури є використання серверної проміжної ланки. Завдяки цьому користувачі не взаємодіють безпосередньо із зовнішніми API, зокрема Plant.id, а всі запити проходять через захищений сервер. Це дає змогу централізовано обробляти помилки, керувати доступом, проводити журналювання та в разі потреби — масштабувати або змінювати логіку без необхідності оновлення клієнтської частини. Такий підхід також забезпечує кращу підтримку й уніфікацію всієї системи, відкриваючи можливості для подальшого розширення функціоналу, інтеграції нових сервісів або зміни бізнес-логіки без порушення стабільності роботи вебзастосунку.

2.3 Клієнтська частина веб-застосунку

Фронтенд реалізований з використанням чистого HTML, CSS і JavaScript без фреймворків. Основні функції користувацького інтерфейсу:

— відображення статусу автентифікації та даних користувача;

— вхід і реєстрація користувачів через вебформу;

— завантаження зображення для розпізнавання рослини;

— перегляд історії власних розпізнавань.

Інтерфейс побудований таким чином, щоб бути максимально простим, адаптивним та інтуїтивно зрозумілим. Усі дії здійснюються через події DOM та HTTP-запити до бекенду. Для зберігання токена доступу використовується `localStorage`, що забезпечує збереження сесії між оновленнями сторінки.

Користувач взаємодіє з програмою через сторінки `auth.html` (вхід/реєстрація) та `index.html` (основний функціонал). Залежно від статусу авторизації, елементи інтерфейсу змінюють вигляд або стають недоступними.

3 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 UML проєктування ПЗ

На початковому етапі розробки програмного забезпечення було здійснено моделювання поведінки користувача в межах системи за допомогою UML-діаграми прецедентів (Use Case Diagram). Цей інструмент широко застосовується в інженерії програмного забезпечення для того, щоб у наочному вигляді відобразити функціональні можливості, доступні користувачу, а також показати логіку їх виконання з урахуванням зв’язків і залежностей. Таке представлення дозволяє краще зрозуміти, які саме сценарії передбачені в системі, які обов’язки виконує клієнтська та серверна частини, і які дані обробляються в кожному випадку.

У контексті розробленого вебзастосунку, для спрощення архітектури та підвищення стабільності взаємодії, було прийнято рішення реалізувати лише один тип користувача — зареєстрований користувач. Такий підхід дозволяє уникнути складних рівнів доступу та багатоступеневої логіки авторизації, зосередившись натомість на якісній реалізації основних функцій, що охоплюють увесь необхідний цикл роботи з системою. Після проходження процесу автентифікації, користувач отримує доступ до всього функціоналу без обмежень, що забезпечує простоту використання й логічну завершеність кожного сценарію.

Процес реєстрації або входу до системи відбувається шляхом введення електронної пошти та пароля у відповідну форму на клієнтському інтерфейсі. Ці дані надсилаються на сервер, де за допомогою сервісу Supabase Auth створюється або перевіряється обліковий запис. У разі успішної авторизації, користувач отримує токен доступу, який зберігається у локальному сховищі браузера (`localStorage`) і використовується для здійснення запитів до захищених маршрутів. Це дозволяє уникнути необхідності повторного входу

при кожному оновленні сторінки та забезпечує персоналізовану взаємодію з даними.

Однією з центральних функцій системи є розпізнавання рослин за зображенням. Користувач має можливість завантажити фото рослини безпосередньо зі свого пристрою через вебінтерфейс. Після цього клієнтська частина перетворює зображення у формат Base64 — це дозволяє зручно передати його у тілі HTTP-запиту — і надсилає результат на сервер. Серверна частина, у свою чергу, ініціює звернення до зовнішнього API Plant.id, яке аналізує отримане зображення та повертає найімовірнішу назву рослини з додатковими даними. Увесь цей процес виконується асинхронно, із врахуванням захисту API-ключа на стороні сервера.

Після отримання результату розпізнавання, система автоматично перевіряє, чи користувач автентифікований, і лише в такому випадку результат записується у базу даних Supabase. Кожен запис містить дані про назву рослини, дату запиту, результат ідентифікації, а також прив’язку до конкретного ID користувача. Така реалізація дозволяє формувати особисту історію взаємодії з системою: користувач може в будь-який момент переглянути всі раніше розпізнані рослини, що були збережені саме ним. Це не лише зручно, а й забезпечує цілісність і приватність обробки персональних даних.

Також важливою є реалізація функції виходу із системи, яка передбачає повне очищення токена доступу з локального сховища. Це означає, що після завершення сесії всі захищені дії (такі як розпізнавання або перегляд рослин) стають недоступними, доки користувач знову не пройде автентифікацію. Такий підхід дозволяє гарантувати, що до конфіденційної інформації не матиме доступ стороння особа, навіть якщо пристрій залишиться ввімкненим або відкритим.

Візуалізація усіх перерахованих дій була реалізована за допомогою UML-діаграми прецедентів, яка наведена на рисунку 3.1. На ній чітко позначено основні можливості, які має зареєстрований користувач: автентифікація, розпізнавання рослини, збереження результату, перегляд усіх розпізнань і вихід.

Додатково показано, що збереження результату є вбудованою складовою (include) процесу розпізнавання. Така діаграма дозволила не лише структурувати логіку майбутньої реалізації, а й послужила орієнтиром при написанні клієнтської та серверної частин.

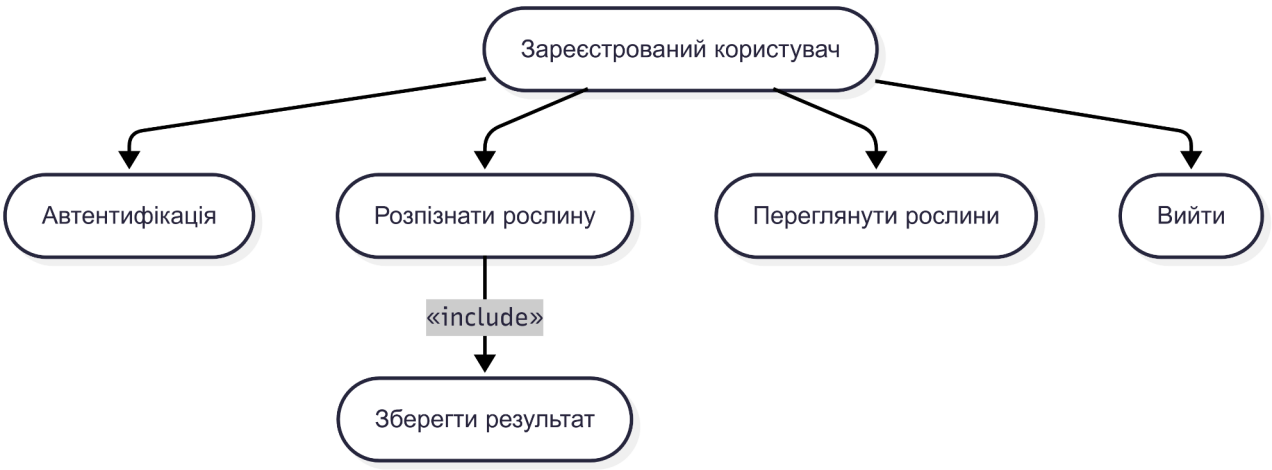

Рисунок 3.1 – Діаграма прецедентів користувача у системі (виконано самостійно)

У підсумку, застосування UML-прецедентів дало змогу зробити підхід до проєктування послідовним і логічним, чітко розмежувати відповідальність між елементами архітектури та уникнути надмірної складності. Водночас такий спосіб моделювання створив передумови для масштабування системи в майбутньому — з можливістю додати нові ролі, функції чи сервіси без суттєвої перебудови існуючої логіки.

3.2 Проєктування архітектури ПЗ

Розроблена система побудована на основі простої, проте ефективної клієнт-серверної архітектури, яка враховує сучасні тенденції у веброзробці та дотримується принципів розділення обов’язків. Головна мета такого архітектурного підходу — забезпечення надійної, гнучкої та масштабованої інфраструктури, що дозволяє кожному з компонентів системи функціонувати

автономно, взаємодіючи з іншими лише через визначені інтерфейси. Це сприяє підвищенню безпеки, полегшує обслуговування та створює умови для незалежного оновлення чи заміни окремих частин без шкоди для загальної роботи системи.

Фронтенд, тобто клієнтська частина, реалізована з використанням стандартних вебтехнологій — HTML для структури, CSS для стилізації та JavaScript для реалізації логіки взаємодії. Увесь клієнтський код є повністю статичним і не потребує складного серверного оточення — він може бути розміщений як на локальному пристрої, так і на будь-якому хостингу, зокрема безкоштовному. Завдяки такому підходу забезпечується висока швидкість завантаження, легкість у розгортанні та можливість гнучкої адаптації інтерфейсу до потреб користувача. Основна логіка взаємодії, включаючи автентифікацію, керування інтерфейсом, обробку подій і підготовку запитів, реалізована безпосередньо у браузері. Після успішного входу користувача до системи, у локальному сховищі браузера (localStorage) зберігається токен, який надалі додається до всіх захищених запитів до сервера, забезпечуючи автентифікований доступ.

Серверна частина, або бекенд, реалізована за допомогою платформи Node.js із використанням Express — легкого та продуктивного фреймворку, що широко застосовується для створення RESTful API. Уся комунікація між клієнтом і сервером побудована на обміні HTTP-запитами, де структура запитів та відповідей відповідає формату JSON. Сервер обробляє запити користувачів, що надходять з клієнтської частини, здійснює реєстрацію та автентифікацію користувачів через механізми Supabase Auth, перевіряє дійсність токенів перед кожною взаємодією з базою даних, обробляє зображення та надсилає їх до зовнішнього API Plant.id. Усі отримані результати розпізнавання зберігаються у Supabase, прив’язуючись до облікового запису користувача. Така централізація логіки обробки даних дозволяє точно контролювати усі дії та гарантує, що жодна критична інформація не потрапить до клієнта без перевірки.

Окрему увагу приділено безпеці. Сервер виступає як проміжний і захисний прошарок між користувачем і зовнішніми сервісами. Він перехоплює всі запити, перевіряє їхню валідність, додає службову інформацію (наприклад, API-ключі) та передає далі лише ті дані, які справді потрібні для обробки. Усі конфіденційні параметри, такі як ключі доступу до API Plant.id або параметри підключення до бази даних, зберігаються у середовищі виконання Node.js — у файлі `.env`, який не включається до публічного репозиторію і недоступний ззовні. Це гарантує, що навіть у разі компрометації клієнтського коду користувачі не зможуть отримати доступ до критичної інформації.

База даних реалізована на основі платформи Supabase — сучасного хмарного рішення, яке надає повноцінний доступ до PostgreSQL через вебінтерфейс і REST-запити. Supabase виконує подвійну функцію: слугує як система автентифікації (Supabase Auth) та одночасно виступає основним сховищем даних. Усі результати розпізнавання рослин, які надходять від Plant.id, зберігаються разом із даними про користувача: його ідентифікатор, дата та час розпізнавання, а також назва рослини, визначена на основі аналізу зображення. Це дозволяє легко організувати персоналізовану історію для кожного користувача та надалі реалізовувати функції фільтрації, сортування або повторного перегляду записів.

Уся система побудована на взаємодії через HTTP, де для авторизованих запитів у заголовку використовується токен доступу (Authorization: Bearer <token>). Серверна частина перевіряє його справжність через SDK Supabase, а у разі відсутності або недійсності — повертає відповідь з помилкою доступу. Це забезпечує високий рівень захисту персональних даних і унеможливлює несанкціонований доступ до чужих ресурсів.

Головною перевагою обраної архітектури є її модульність і масштабованість. У разі необхідності можна незалежно оновлювати будь-який компонент: змінити дизайн клієнтської частини, оновити логіку обробки запитів на сервері або перемістити базу даних на іншу платформу. Наприклад,

клієнтську частину можна без проблем розмістити на GitHub Pages, а сервер — на Render, що дозволяє ефективно використовувати ресурси і розділити навантаження. Усі компоненти взаємодіють лише через стандартизовані запити, що дозволяє легко додавати нові функції або змінювати логіку без повного переписування системи.

Крім того, така архітектура легко інтегрується з додатковими зовнішніми сервісами — наприклад, з іншими API для ідентифікації, аналітики чи зберігання зображень. Це відкриває широкі можливості для подальшого розвитку проєкту. Водночас передача конфіденційних даних відбувається лише через сервер, що мінімізує ризик доступу до критичних параметрів із боку користувача.

У підсумку, обрана архітектура дозволила побудувати надійну, гнучку та безпечну вебсистему, що відповідає сучасним вимогам, легко розгортається, інтегрується з зовнішніми платформами і дає змогу забезпечити зручний і захищений доступ до сервісів для кінцевого користувача. Вона є оптимальною для систем, де важлива обробка приватних даних, розподілене середовище виконання та швидке масштабування.

3.3 Проєктування структури зберігання даних

Для зберігання даних у розробленій програмній системі використовується хмарна платформа Supabase. Цей сервіс надає прямий доступ до повноцінної реляційної бази даних PostgreSQL, а також містить низку вбудованих функцій, таких як підтримка авторизації, керування сесіями користувачів, та зручні інструменти для створення й адміністрування таблиць із гнучкими перевірками доступу. Саме завдяки поєднанню простоти інтеграції, високої надійності та функціональної повноти було прийнято рішення на користь Supabase. Платформа дозволяє не лише зберігати загальнодоступні дані, а й створювати персоналізовані записи, що пов’язані з конкретними користувачами — це

критично важливо для реалізації приватності й персонального досвіду у вебзастосунку.

Центральним елементом структури зберігання є таблиця `saved_plants`, у якій накопичуються результати розпізнавання рослин. Кожен запис у цій таблиці містить набір обов'язкових атрибутів, що забезпечують зберігання повної інформації про виконану операцію. Зокрема, це поле `id`, що є унікальним ідентифікатором запису й генерується автоматично, а також поле `created_at`, у якому фіксується дата й час створення відповідного запису. Додатково, кожен запис містить поле `user_id`, яке дозволяє встановити, який саме користувач ініціював розпізнавання. Це значення отримується за допомогою механізму авторизації Supabase Auth і дозволяє чітко зв'язати дані з конкретним обліковим записом. У полі `plant_name` зберігається назва визначеної рослини, отримана через API-сервіс Plant.id. Також передбачено поле `image_url`, у яке в майбутньому планується додавати посилання на зображення, що було використано при розпізнаванні — наразі це поле залишене порожнім, однак його структура вже закладена для майбутніх оновлень.

Для створення нового запису в системі обов'язковою умовою є проходження процедури автентифікації. Це означає, що жодна взаємодія з базою даних не може бути виконана анонімним або неавторизованим користувачем. Такий підхід дозволяє забезпечити контроль над кожною дією в системі та гарантує, що всі записи створюються виключно конкретними користувачами, які пройшли перевірку особи. Після успішного входу до системи користувач отримує унікальний токен, за допомогою якого відбувається ідентифікація під час усіх подальших запитів. Це дозволяє точно визначати, хто саме надсилає той чи інший запит на створення або отримання інформації.

Усі записи, що створюються в системі, пов'язуються з унікальним ідентифікатором користувача (user_id), який прив'язаний до його облікового запису в Supabase Auth. Таким чином, у базі зберігається не просто набір даних — кожен рядок чітко асоційований із конкретною особою. Це дозволяє

уникнути ситуацій, коли один користувач міг би переглядати або змінювати записи іншого. Дані чітко розмежовані на рівні зберігання, що є важливою передумовою для безпечної роботи із персоналізованим контентом.

Для реалізації такого розмежування доступу використовується вбудований у Supabase механізм Row Level Security (RLS). Ця технологія дає змогу задавати точні правила доступу до кожного окремого рядка таблиці, залежно від умов, визначених у політиках. У випадку з таблицею `saved_plants` встановлено правило `user_id = auth.uid()`. Воно означає, що доступ до запису буде надано лише в тому випадку, якщо ідентифікатор користувача, який виконує запит, збігається з ідентифікатором, записаним у цьому рядку. Це надійний спосіб ізоляції даних, який не вимагає додаткової логіки на клієнті або сервері та зменшує імовірність помилок або неправомірного доступу.

Окрім технічної переваги, така реалізація значно спрощує супровід і розвиток системи. Розробнику не потрібно вручну перевіряти права доступу у кожному запиті — перевірка виконується автоматично на рівні бази даних. Це не лише підвищує ефективність, а й покращує безпеку: навіть якщо клієнт спробує надіслати запит до чужого запису, Supabase самостійно відкине цей запит без передачі будь-якої інформації назад.

З позиції архітектури безпеки, обрана структура реалізації є прикладом централізованого і надійного контролю над усіма критичними діями в системі. Автентифікація, авторизація, перевірка прав доступу і зберігання даних реалізовані не у вигляді окремих функцій, розкиданих по проєкту, а зосереджені у платформі Supabase, яка бере на себе виконання цих завдань. Це дозволяє уникнути дублювання логіки, зменшує ризики помилок і забезпечує єдину точку контролю, яку легко підтримувати та розширювати.

Ще однією перевагою такої архітектури є те, що усі запити до бази даних відбуваються виключно через серверну частину. Клієнт не має прямого доступу до Supabase — усі взаємодії, включаючи створення, читання, редагування або видалення записів, обробляються на рівні бекенду. Це дозволяє приховати всі

чутливі ключі, логіку перевірок і бізнес-правила від кінцевого користувача, що значно зменшує площу потенційної атаки. Навіть у випадку, якщо користувач зможе перехопити запит або спробує змінити параметри, перевірка на сервері й політики RLS не дозволять виконати несанкціоновану дію.

Завдяки такій побудові системи зберігання, розробник отримує надійний і гнучкий інструмент для керування даними. Водночас користувач має впевненість, що його інформація ізольована, захищена й доступна тільки йому. Це особливо важливо у системах, де обробляються персоналізовані або чутливі дані, і де помилка в контролі доступу може призвести до серйозних наслідків.

У підсумку, описана модель збереження даних поєднує простоту реалізації з високим рівнем безпеки, масштабованості та розширюваності. Вона не вимагає складної ручної логіки, повністю сумісна з сучасними вимогами до захисту інформації та ідеально вписується в клієнт-серверну архітектуру проєкту.

Такий підхід не лише надає достатню гнучкість і безпеку на поточному етапі, а й відкриває можливості для майбутнього розвитку. Наприклад, у таблицю `saved_plants` можна буде легко додати нові поля — такі як шлях до зображення, рівень точності розпізнавання або особисті нотатки. Крім того, є потенціал для створення нових таблиць, зокрема для реалізації функцій збереження улюблених рослин, ведення історії запитів або додавання тегів. Побудована структура забезпечує стійку основу для масштабування й адаптації системи відповідно до потреб користувачів, зберігаючи при цьому зрозумілість і простоту реалізації.

3.4 Найцікавіші алгоритми та методи

У процесі розробки програмної системи було реалізовано низку алгоритмів, які не тільки забезпечують функціональність вебзастосунку, а й відповідають вимогам безпеки, надійності та масштабованості. Особливу увагу

було приділено двом ключовим аспектам: реалізації логіки взаємодії з зовнішнім API Plant.id для розпізнавання рослин та захищеному збереженню результатів у базу даних Supabase. Обидва ці компоненти тісно пов’язані з обробкою асинхронних запитів, перевіркою автентичності та дотриманням сучасних практик захисту даних, що є критично важливим для будь-якого вебсервісу з персоніфікованим доступом.

Алгоритм розпізнавання рослини реалізовано на стороні сервера з використанням Node.js та Express.js. Коли користувач надсилає зображення, воно проходить попередню обробку на фронтенді — зображення конвертується у формат Base64. Далі цей закодований масив передається у вигляді JSON-об’єкта до бекенду, де формується запит до API Plant.id. Цей API є потужним інструментом на основі машинного навчання, здатним на основі зображення з високою ймовірністю визначити вид рослини.

Нижче представлено реальний фрагмент коду з проєкту, що відповідає за взаємодію з цим зовнішнім сервісом:

```
const response = await fetch("https://api.plant.id/v2/identify", {
      method: "POST",
      headers: {
            "Content-Type": "application/json",
            "Api-Key": process.env.PLANT_ID_API_KEY
      },
      body: JSON.stringify({
            images: [imageBase64],
            modifiers: ["crops_fast", "similar_images"],
            plant_language: "en",
      plant_details: ["common_names", "url", "wiki_description"]
      })
      });
```

У цьому запиті передаються такі параметри, як список зображень (`images`), мовні налаштування, модифікатори для покращення точності розпізнавання та перелік деталей, які потрібно отримати у відповіді. Важливо відзначити, що ключ до API (`Api-Key`) зберігається у `.env` файлі і не

передається на клієнтську частину, що повністю відповідає сучасним стандартам захисту конфіденційної інформації.

Отримана відповідь від API містить декілька варіантів розпізнавання, з яких система вибирає найбільш ймовірний. Далі, у разі успішної автентифікації, результат записується у базу даних Supabase. Саме перед збереженням результату відбувається обов’язкова перевірка access token, що підтверджує ідентичність користувача. Таким чином, зберігаються лише ті дані, які пов’язані з реальним і дійсним обліковим записом.

Для перевірки токена і отримання ID користувача, який ініціював запит, застосовується вбудована функція з пакета `@supabase/supabase-js`, що виконує звернення до Supabase Auth. Приклад коду:

```
const { data: userData, error: authError } = await
supabase.auth.getUser(access_token);

if (authError || !userData?.user?.id) {
      return res.status(401).json({ error: 'Unauthorized' });
}
```

Цей фрагмент виконує дві критично важливі функції. По-перше, токен перевіряється на справжність і термін дії. По-друге, з відповіді витягується унікальний ідентифікатор користувача (`user.id`), який потім використовується для прив’язки результату до конкретного облікового запису. Якщо токен недійсний або термін його дії вичерпано, система повертає помилку 401 Unauthorized і припиняє подальше виконання запиту.

Завдяки такій логіці, система не лише виконує своє пряме завдання — розпізнавання рослин — а й забезпечує повну відповідність принципам безпеки: жоден користувач не може отримати доступ до чужих даних або виконати запис без дійсної авторизації. Це надзвичайно важливо в контексті роботи з персональними обліковими записами.

Описані вище алгоритми — це не просто технічна реалізація функцій, а демонстрація дотримання найкращих практик сучасної веброзробки. Асинхронна обробка запитів, захищене збереження токенів, централізоване опрацювання даних і чітка верифікація користувачів — усе це формує надійний фундамент, на якому побудовано весь застосунок. Крім того, описані методи є масштабованими: у майбутньому до них можна легко додати нові типи даних, інші API або інтерфейси без потреби кардинально змінювати логіку.

Таким чином, навіть ці два ключові фрагменти — запит до Plant.id та перевірка access token — показують, що система не просто виконує технічну задачу, а робить це ефективно, надійно та відповідно до сучасних стандартів розробки безпечних вебзастосунків.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

4.1 Використання Supabase Auth

У розробленій програмній системі автентифікація користувачів реалізована за допомогою вбудованого сервісу Supabase — Supabase Auth, який забезпечує повноцінний механізм керування обліковими записами. Цей сервіс надає засоби для створення користувачів, їх входу до системи, підтвердження електронної пошти, зберігання сесій, а також перевірки доступу до ресурсів на основі токенів.

Після успішної реєстрації або входу, Supabase формує спеціальний токен доступу (access token), який має формат JWT (JSON Web Token). Цей токен містить зашифровану інформацію про користувача та строк його дії. На клієнтській частині (у браузері) токен зберігається у локальному сховищі (localStorage) й автоматично додається до кожного запиту до серверної частини системи. Такий підхід дозволяє забезпечити зручну та безпечну авторизацію без необхідності повторного входу під час сеансу.

На стороні сервера, створеного на основі Node.js з використанням Express.js, виконується перевірка автентичності кожного запиту шляхом звернення до функції `supabase.auth.getUser(token)` з пакету `@supabase/supabase-js`. Ця функція дозволяє верифікувати отриманий токен і визначити, чи є користувач дійсним, а також отримати його унікальний ідентифікатор (user ID). У разі, якщо токен недійсний або відсутній, система повертає помилку 401 Unauthorized, тим самим запобігаючи несанкціонованому доступу до персональних даних.

Вся система авторизації працює через запити типу `Authorization: Bearer <access_token>`, які автоматично додаються до кожного запиту з клієнтської частини до захищених маршрутів бекенду. Таким чином, користувачі, що не пройшли автентифікацію, не мають доступу до операцій розпізнавання, збереження результатів або перегляду власних записів.

Інтерфейс авторизації користувача реалізований на окремій сторінці `auth.html`. Користувач має змогу ввести електронну пошту та пароль, після чого система виконує відповідний запит до бекенду для входу або реєстрації. Після успішного входу користувач автоматично перенаправляється на головну сторінку додатку, де бачить свій статус авторизації, електронну пошту, а також має можливість вийти з системи. На рисунку 4.1 зображено приклад користувацького інтерфейсу для входу до системи.

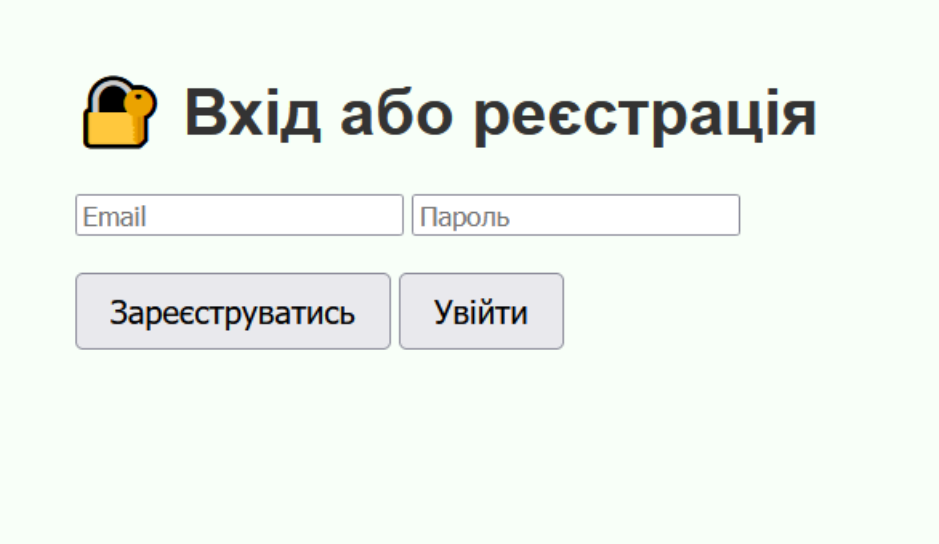

Рисунок 4.1 – Форма авторизації користувача у клієнтському інтерфейсі(виконано самостійно)

Крім цього, Supabase надає веб-інтерфейс для розробника, у якому можна переглядати та адмініструвати зареєстрованих користувачів. У розділі Auth → Users відображається список облікових записів з датами створення, станом верифікації електронної пошти, ID сесій та іншою службовою інформацією. Цей інтерфейс дозволяє видаляти користувачів, вручну активувати акаунти або змінювати налаштування автентифікації.

Застосування Supabase Auth дало змогу уникнути реалізації складної логіки реєстрації, зберігання сесій і шифрування токенів. Усі необхідні механізми захисту вже реалізовані в системі Supabase. Водночас, завдяки

перенесенню всієї взаємодії з базою даних і зовнішніми API на власний сервер, було досягнуто високого рівня безпеки: API-ключі залишаються недоступними на клієнтській стороні, а обробка авторизації повністю контролюється на бекенді.

Таким чином, обраний підхід забезпечує ефективне поєднання зовнішнього хмарного сервісу Supabase з власним сервером і клієнтською частиною, що дозволяє дотримуватись сучасних стандартів побудови безпечних та масштабованих вебсистем.

4.2 Інтеграція з Plant.id API

Однією з найважливіших функціональних можливостей розробленого програмного забезпечення є автоматичне визначення рослин за зображенням, завантаженим користувачем. Для реалізації цього завдання було інтегровано зовнішній вебсервіс Plant.id API — спеціалізовану платформу, що використовує технології штучного інтелекту для класифікації рослин на основі візуального аналізу. Обраний сервіс є одним із найточніших у своїй категорії, що дозволяє досягти високої достовірності результатів без необхідності самостійного розроблення та навчання складних моделей машинного навчання.

Інтеграція з Plant.id реалізована повністю через серверну частину застосунку, створену на основі Node.js. Така реалізація дозволяє централізовано контролювати увесь процес взаємодії з API, зокрема — забезпечити безпеку конфіденційних даних і стандартизувати запити. На стороні користувача все починається із завантаження фотографії рослини у форматі JPEG або PNG. Зображення автоматично конвертується у формат Base64 ще до надсилання на сервер, що дає змогу передавати його у вигляді тексту через HTTP. Водночас система перевіряє, чи користувач є авторизованим: тільки наявність дійсного токена дозволяє звернутися до маршруту, відповідального за розпізнавання. Це запобігає несанкціонованому використанню ресурсоємної функції.

На серверному рівні запит до API Plant.id формується динамічно на основі отриманого зображення. Разом із самим зображенням передаються супровідні параметри, які впливають на якість розпізнавання та характер відповіді. Наприклад, задається бажана мова вихідних даних (у цьому випадку — англійська), типи потрібної додаткової інформації, такі як альтернативні назви, короткий опис або зовнішні посилання на довідкові джерела, а також модифікатори, які допомагають покращити точність аналізу зображення. Таким чином, кожен запит є адаптивним і формально повноцінним, що забезпечує високу ефективність взаємодії з API.

У відповідь сервер отримує структурований JSON-об’єкт, який містить перелік найбільш ймовірних варіантів розпізнаних рослин. Серед них обирається варіант з найвищим коефіцієнтом відповідності, який і вважається основним результатом. Цей результат разом з user ID зберігається до бази даних Supabase. Таким чином, забезпечується збереження інформації про кожне розпізнавання в контексті конкретного користувача, що дає змогу переглядати історію і повертатися до попередніх результатів у будь-який момент.

Ключовою перевагою такого підходу є те, що уся взаємодія з зовнішнім API винесена виключно на бекенд. Це дозволяє не лише приховати API-ключ у середовищі виконання, ізолювавши його від клієнта, а й централізувати контроль над кількістю запитів до стороннього сервісу. За потреби можна впровадити кешування, обмеження частоти запитів або навіть механізм вибору альтернативного сервісу без жодних змін у клієнтській частині. Завдяки цьому розробник отримує повний контроль над процесом, а сама система стає масштабованою, розширюваною та стійкою до змін з боку сторонніх API.

З точки зору кінцевого користувача процес максимально спрощено. Вся складна логіка залишається прихованою: достатньо лише завантажити зображення та натиснути кнопку. Після цього користувач бачить назву рослини, яку вдалося визначити, а також має змогу зберегти результат і переглянути його пізніше. Водночас система автоматично зв’язує результат із обліковим записом,

що унеможливлює втрату інформації або її випадкове змішування з даними інших користувачів. (рис. 4.2)

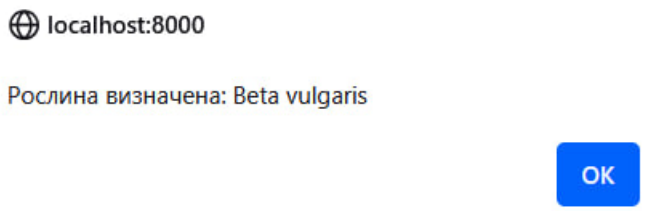

Рисунок 4.2 – Результат визначення рослини (виконано самостійно)

Загалом, інтеграція з Plant.id значно розширила функціональність вебзастосунку, зробивши його практично корисним і зручним у щоденному використанні. Завдяки такому рішенню вдалося реалізувати складну технічну функцію у вигляді простої та надійної взаємодії, яка повністю відповідає вимогам до сучасних захищених вебсервісів, побудованих за клієнт-серверною моделлю.

4.3 Опис архітектури проєкту

Розроблена програмна система побудована на базі клієнт-серверної архітектури, яка є водночас простою у реалізації та ефективною з погляду розподілу обов’язків і масштабування. Основна ідея цієї архітектури полягає в чіткому розмежуванні функціональності між клієнтською та серверною частинами, а також у використанні зовнішніх хмарних сервісів для розширення можливостей і забезпечення гнучкості, безпеки та зручності в розгортанні.

Клієнтська частина застосунку створена з використанням стандартних вебтехнологій — HTML, CSS і JavaScript. Усі сторінки програми працюють безпосередньо у браузері користувача, що виключає необхідність встановлення додаткового програмного забезпечення. Інтерфейс дозволяє зручно взаємодіяти із застосунком: переглядати список збережених рослин, завантажувати

зображення для подальшого аналізу, реєструватися, входити до системи. Передача даних між клієнтом і сервером відбувається через HTTP-запити, а токен доступу до захищених ресурсів зберігається у localStorage браузера. Клієнт реалізований у вигляді окремого статичного сайту, що дозволяє розміщувати його на простих хостингах, зокрема GitHub Pages. Уся логіка, пов’язана з перевіркою автентичності, обробкою даних і розпізнаванням зображень, винесена на сервер.

Серверна частина реалізована з використанням середовища Node.js та вебфреймворку Express, що дозволяє будувати RESTful API з мінімальними витратами часу та ресурсів. Сервер обробляє всі запити, що надходять з клієнтської частини, забезпечуючи автентифікацію користувачів через Supabase Auth, приймає зображення та пересилає їх до стороннього сервісу Plant.id для ідентифікації рослин. Отримані результати зберігаються у базі даних, прив’язуючись до конкретного облікового запису користувача. Особлива увага приділяється захисту конфіденційних даних: усі ключі та токени зберігаються у файлі .env, що не потрапляє на клієнтську сторону. Це дозволяє уникнути витоків і гарантувати захист від зловмисників.

У системі використовується Supabase як хмарна база даних, яка поєднує в собі реляційне сховище на основі PostgreSQL, вбудовану автентифікацію та механізми управління правами доступу. Supabase Auth дозволяє створювати облікові записи з використанням email та пароля, а також автоматично асоціює збережені дані із зареєстрованим користувачем. У таблиці `saved_plants` зберігаються результати розпізнавання разом з унікальним ідентифікатором користувача, що забезпечує персоналізацію та впорядкованість даних. Крім цього, Supabase автоматично обробляє автентифікацію та авторизацію, значно спрощуючи логіку на сервері.

Передача даних у системі реалізована у форматі JSON через HTTP. Клієнт формує запити за допомогою функції `fetch`, додаючи токен авторизації у заголовок. Сервер, у свою чергу, перевіряє цей токен за допомогою Supabase

SDK і лише після підтвердження достовірності виконує запит. Це дозволяє гарантувати безпечну взаємодію без доступу сторонніх осіб до критичних ресурсів.

Загалом, обрана архітектура забезпечує високу гнучкість і безпечність реалізованого застосунку. Вона дозволяє незалежно оновлювати або масштабувати окремі компоненти, ізолює критичні дані від клієнтської частини, а також підтримує інтеграцію зі сторонніми сервісами винятково через серверну логіку. Такий підхід сприяє зменшенню ризиків, спрощує адміністрування та дозволяє розгортати фронтенд і бекенд окремо: клієнтський інтерфейс на GitHub Pages, а серверну частину — на Render. Це дає змогу ефективно керувати ресурсами, оновленнями і забезпечувати стабільну роботу системи.

5 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Ручне тестування

У межах реалізації програмного забезпечення було проведено повноцінне ручне тестування, яке охоплює перевірку всіх основних функціональних компонентів вебзастосунку. Такий підхід дозволив переконатися в коректності логіки, стабільності роботи інтерфейсу, надійності зв’язку між клієнтською та серверною частинами, а також у безпеці збереження й обробки даних. Тестування здійснювалось шляхом імітації типових сценаріїв користування, максимально наближених до реальних умов експлуатації системи.

На початковому етапі було перевірено процес реєстрації нового користувача через відповідну форму у вебінтерфейсі. Введення електронної пошти та паролю автоматично призводило до формування запиту до серверної частини, яка у свою чергу взаємодіяла з Supabase для створення облікового запису. Успішність цього етапу підтверджувалась відображенням повідомлення про реєстрацію, а також появою токена авторизації у локальному сховищі браузера. Після цього виконувалась перевірка переходу на головну сторінку та коректного відображення статусу авторизації, що свідчило про правильну інтеграцію механізму сесій.

У подальшому здійснювалась перевірка процесу входу з уже існуючими обліковими даними, де тестувались як правильні, так і помилкові варіанти. У випадку введення вірного паролю користувач отримував доступ до захищених функцій, і повторна авторизація не вимагалась доти, доки токен залишався в локальному сховищі. При введенні невірних даних система надавала зрозуміле повідомлення про помилку та блокувала доступ до сервісів.

Одним із ключових етапів тестування стала перевірка функції розпізнавання рослин за зображенням. Тестувались зображення у форматах JPEG і PNG, з різною роздільною здатністю та якістю. Після завантаження зображення клієнтська частина перетворювала його у формат Base64, після чого

надсилала на сервер, де вже відбувався запит до зовнішнього API Plant.id. Отриманий результат оброблявся сервером і зберігався у базі Supabase разом із прив’язкою до конкретного користувача, що дозволяло у подальшому відображати персоналізовану історію розпізнань. Особливу увагу було приділено перевірці того, чи доступні ці результати лише відповідному авторизованому обліковому запису, і чи правильно система блокує спроби доступу з боку інших користувачів.

Тестування також охоплювало сценарії виходу із системи. Після натискання кнопки «Вийти» перевірялась коректність очищення токена в локальному сховищі та реакція інтерфейсу при повторному завантаженні сторінки. Було встановлено, що після виходу користувач втрачає доступ до всіх захищених дій, і для поновлення доступу необхідно пройти повторну авторизацію. Такий підхід гарантує базовий рівень безпеки та передбачуваність поведінки застосунку.

Окремий блок ручного тестування був присвячений перевірці реакції системи на потенційно помилкові або нестандартні дії користувача. Було перевірено, як система поводиться при спробі здійснити розпізнавання без автентифікації, надсиланні порожнього зображення або зображення у неправильному форматі, а також при зміні або видаленні токена вручну за допомогою інструментів розробника в браузері. Крім цього, вручну моделювались ситуації, коли користувач намагався надіслати запит до захищеного ресурсу із недійсним, пошкодженим або простроченим токеном. У кожному з цих випадків система демонструвала очікувану поведінку: виводила повідомлення про помилку, блокувала небажаний запит або перенаправляла користувача до сторінки входу. Таким чином, реалізовані механізми обробки помилок показали свою ефективність і сталість.

Завершальний етап ручного тестування полягав у багаторазовому повторенні основних сценаріїв з різними параметрами, включаючи швидкість з’єднання, різні браузери, мобільні та десктопні пристрої. Усі функції

працювали стабільно, не викликаючи зависань, втрат даних чи порушення логіки. Система коректно зчитувала зображення, надсилала їх, отримувала відповідь і зберігала її у базі, що дозволяє стверджувати про її загальну надійність у реальних умовах експлуатації.

У підсумку можна зазначити, що ручне тестування повністю підтвердило працездатність системи, її стійкість до некоректних дій, правильну взаємодію всіх компонентів і відповідність заявленому функціоналу. Система є надійною та безпечною для реального використання, що дає змогу переходити до наступних етапів — аналізу продуктивності та розгортання в публічному середовищі.

ВИСНОВКИ

У межах виконання кваліфікаційної роботи був реалізований веб-застосунок, що дозволяє користувачам автоматично розпізнавати кімнатні та садові рослини за зображенням, зберігати результати в базі даних та переглядати особисту історію взаємодії.

Розробка веб-застосунку здійснювалась з використанням Node.js як серверного середовища виконання та Express як фреймворку для створення RESTful API, що забезпечило простоту та гнучкість побудови серверної логіки. Для зберігання даних та реалізації автентифікації було обрано платформу Supabase, яка поєднує в собі можливості хмарної бази даних PostgreSQL, систему авторизації та вбудовані засоби адміністрування користувачів.

Основний функціонал включає:

— автентифікацію користувачів із верифікацією електронної пошти;

— обробку зображень і їх передачу до зовнішнього API Plant.id для розпізнавання рослин;

— збереження результатів у базі даних із прив’язкою до облікового запису;

— перегляд історії розпізнань через зручний вебінтерфейс.

Весь обмін даними між клієнтом, сервером і зовнішніми сервісами реалізовано через HTTP-запити з передачі токена доступу у заголовках, що забезпечує безпечну взаємодію та обмеження доступу до персональних даних. Також реалізовано розділення обов’язків між клієнтською та серверною частинами з урахуванням сучасних вимог до веброзробки.

Таким чином, створено сучасну, функціональну та безпечну вебсистему, яка може бути основою для подальшого розширення — наприклад, додавання мобільного застосунку, нових алгоритмів класифікації, або системи сповіщень для користувача.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Mozilla Developers Network. JavaScript documentation [Електронний ресурс]. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript (дата звернення: 22.06.2025).

2. Node.js. Documentation [Електронний ресурс]. URL: https://nodejs.org/en/docs (дата звернення: 22.06.2025).

3. Supabase Docs. Supabase [Електронний ресурс]. URL: https://supabase.com/docs (дата звернення: 22.06.2025).

4. Express.js. Documentation [Електронний ресурс]. URL: https://expressjs.com/ (дата звернення: 22.06.2025).

5. Plant.id API. Plant identification documentation [Електронний ресурс]. URL: https://web.plant.id/plant-identification-api/ (дата звернення: 22.06.2025).

6. React Documentation [Електронний ресурс]. URL: https://react.dev/ (дата звернення: 22.06.2025).

7. GitHub Docs. GitHub documentation [Електронний ресурс]. URL: https://docs.github.com/ (дата звернення: 22.06.2025).

8. W3Schools. HTML Tutorial [Електронний ресурс]. URL: https://www.w3schools.com/html/ (дата звернення: 22.06.2025).

9. CSS Tutorial. W3Schools [Електронний ресурс]. URL: https://www.w3schools.com/css/ (дата звернення: 22.06.2025).

10. https://github.com/NurePechenevskyiArtem/2025_B_PI_PZPI-21-10_Pechenevskyi_A_A (дата звернення: 25.06.2025).