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-сервіс для планування розміщення декоративних рослин на земельній ділянці «Flexible landscape design»
(тема)

Виконав:
Студент 4 курсу, групи ПЗПІ-20-6
Маценко Д. О.
(прізвище, ініціали)
Спеціальність 121 – Інженерія програмного Забезпечення
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник доц. кафедри ПІ Кравець Н.С.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З.В. Дудар
(підпис) (прізвище, ініціали)

Харків 2024

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)
«____» _____ 20 ____ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Маценку Дмитру Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи: Веб-сервіс для планування розміщення декоративних рослин на земельній ділянці «Flexible landscape design»

затверджена наказом по університету від 20 травня 2024 р. № 471Ст

2. Термін подання студентом роботи до екзаменаційної комісії 07 червня 2024 р.

3. Вихідні дані до роботи *В програмній системі передбачити: веб-сайт, на якому передбачити можливість реєстрації користувача та подальшого створення даних про послуги і проекти у сфері ландшафтного дизайну. Використовувати технологію JS.*

4. Перелік питань, що потрібно опрацювати в роботі: *вступ, аналіз предметної області, формування вимог до програмної системи, архітектура та проектування програмного продукту, висновки, перелік джерел посилань.*

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін	Відмітки про виконання
1	Аналіз предметної галузі	03.04.2024	Виконано
2	Створення специфікації ПЗ	03.04.2024 – 16.04.2024	Виконано
3	Проєктування та розробка ПЗ	10.04.2024 – 17.05.2024	Виконано
4	Тестування та дослідна експлуатація ПЗ	17.04.2024 – 26.05.2024	Виконано
5	Написання пояснювальної записки	20.04.2024 – 01.06.2024	Виконано
6	Перевірка пояснювальної записки	04.06.2024	Виконано
7	Оцінка роботи рецензентом	05.06.2024	Виконано
8	Здача роботи у електронний архів	06.06.2024	Виконано
9	Попередній захист кваліфікаційної роботи	17.06.2024	Виконано
10	Захист кваліфікаційної роботи	20.06.2024	Виконано

Дата видачі завдання «3» квітня 2024 р.

Студент _____ Маценко Д. О.
(підпис)

Керівник роботи _____ доц. кафедри ПІ Кравець Н.С.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до кваліфікаційної роботи бакалавра: 90 стор., 16 рис., 2 табл., 13 джерел.

АДАПТИВНИЙ ДИЗАЙН, ВЕБ-СЕРВІС, КОНТАКТНА ФОРМА, ЛАНДШАФТНИЙ ДИЗАЙН, ПРЕЗЕНТАЦІЯ ПОСЛУГ, РОЗРОБКА САЙТУ

Об’єктом розробки є веб-сервіс для презентації та надання послуг у сфері ландшафтного дизайну.

Метою роботи є створення веб-сервісу, який допоможе компаніям, що займаються ландшафтним дизайном, презентувати свої послуги, демонструвати виконані проекти, а також забезпечити зручний спосіб взаємодії з потенційними клієнтами.

Для вирішення поставлених завдань використовувалися HTML, CSS, JavaScript, Node.js, Express, MongoDB, Mongoose, Multer та Session.

У результаті розробки було створено веб-сервіс "Flexible landscape design", який складається з декількох основних розділів. Головна сторінка містить навігаційне меню, девіз компанії та кнопки для швидкого доступу до основних розділів сайту. Розділ послуг детально описує послуги, що надає компанія, з використанням зображень та текстових описів. Розділ виконаних робіт представляє галерею зображень прикладів виконаних проектів. Розділ контактів включає інформацію про компанію, посилання на соціальні мережі та контактну форму для зв’язку з компанією.

Веб-сервіс забезпечує адаптивний дизайн, що дозволяє коректно відображатися на різних пристроях, включаючи настільні комп’ютери, планшети та мобільні телефони. Розроблений веб-сервіс сприяє ефективному представленню компанії на ринку послуг з ландшафтного дизайну, забезпечуючи зручний інтерфейс для користувачів та можливість легкого зв’язку з компанією.

ADAPTIVE DESIGN, WEB SERVICE, CONTACT FORM, LANDSCAPE DESIGN, PRESENTATION OF SERVICES, WEBSITE DEVELOPMENT

The object of development is a web service for presentation and provision of services in the field of landscape design.

The purpose of the work is to create a web service that will help landscape design companies present their services, demonstrate completed projects, and provide a convenient way to interact with potential clients.

HTML, CSS, JavaScript, Node.js, Express, MongoDB, Mongoose, Multer and Session were used to solve the tasks.

As a result of the development, the web service "Flexible landscape design" was created, which consists of several main sections. The main page contains a navigation menu, the company motto and buttons for quick access to the main sections of the site. The services section describes in detail the services provided by the company using images and text descriptions. The completed works section presents a gallery of images of examples of completed projects. The contact section includes information about the company, links to social networks and a contact form for contacting the company.

The web service provides a responsive design that allows it to display correctly on various devices, including desktop computers, tablets and mobile phones. The developed web service contributes to the effective presentation of the company on the market of landscape design services, providing a convenient interface for users and the possibility of easy communication with the company.

Я, Маценко Дмитро Олександрович, студент гр. ПЗПІ-20-6, здобувач вищої освіти на першому (бакалаврському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційн робота на тему «Веб-сервіс для планування розміщення декоративних рослин на земельній ділянці «Flexible landscape design»», що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Усі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови до допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ВСТУП

У сучасному світі, де екологічні та естетичні аспекти мають значну вагу, ландшафтний дизайн стає все більш популярним та затребуваним. Ландшафтний дизайн охоплює планування, дизайн та озеленення відкритих просторів, таких як парки, сади, прибудинкові території тощо. Веб-сервіс "Flexible landscape design" представляє собою платформу, яка покликана спростити процес надання послуг у сфері ландшафтного дизайну, зробивши його більш доступним для широкого кола користувачів. Даний сервіс допомагає компаніям, що спеціалізуються на ландшафтному дизайні, презентувати свої послуги, демонструвати портфоліо виконаних робіт, а також забезпечувати зручну комунікацію з потенційними клієнтами.

Об’єктом дослідження є веб-сервіс "Flexible landscape design", який розробляється для компаній, що надають послуги ландшафтного дизайну. Соціальна значимість цього проекту полягає у сприянні розвитку культурного та естетичного середовища через озеленення та благоустрій міських та заміських територій. Технічна значимість проекту обумовлена потребою в інтеграції сучасних технологій для оптимізації процесів надання послуг, підвищення якості сервісу та забезпечення зручності для кінцевих користувачів. Веб-сервіс повинен забезпечити надійну платформу для презентації послуг, надання вичерпної інформації про компанії та їхні проекти, а також для ефективної комунікації між компаніями та клієнтами.

Метою даного дослідження є розробка веб-сервісу, який забезпечить компаніям, що займаються ландшафтним дизайном, ефективний інструмент для презентації своїх послуг та взаємодії з клієнтами. Основними завданнями, які необхідно вирішити в процесі реалізації проекту, є: створення адаптивного дизайну, який забезпечить коректне відображення на різних пристроях; розробка інтуїтивно зрозумілого інтерфейсу для зручної навігації користувачів; інтеграція контактної форми для зворотного зв'язку; забезпечення безпеки та надійності системи; а також оптимізація швидкості завантаження сайту. У результаті кваліфікаційної роботи повинна бути реалізована комплексна веб-платформа

"Flexible landscape design", яка надає компаніям-ландшафтним дизайнерам ефективний інструмент для презентації своїх послуг і взаємодії з клієнтами. Система повинна забезпечувати декілька ключових функціональних компонентів, кожен з яких є важливим для досягнення основних цілей проекту.

Розробка веб-сервісу "Flexible landscape design" є вкрай актуальною на сучасному етапі. Зростаючий попит на послуги ландшафтного дизайну обумовлений бажанням людей створювати привабливі та функціональні відкриті простори, що сприяють покращенню якості життя. Сучасні технології дозволяють значно спростити процес взаємодії між компаніями та клієнтами, підвищуючи при цьому ефективність надання послуг. В умовах швидкого розвитку цифрових технологій та зростання інтернет-аудиторії, наявність якісного веб-сервісу стає важливою складовою успіху будь-якої компанії. Веб-сервіс "Flexible landscape design" сприяє не лише покращенню комунікації між компаніями та клієнтами, але й підвищенню прозорості та доступності інформації про послуги ландшафтного дизайну, що є важливим фактором у сучасних умовах ринку.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Огляд існуючих досліджень і розробок

У сфері ландшафтного дизайну існує велика кількість різноманітних веб-сервісів та програмних продуктів, які спрямовані на покращення якості та ефективності надання послуг. Веб-сервіси з ландшафтного дизайну допомагають компаніям і приватним особам створювати, планувати та візуалізувати свої проекти, забезпечуючи зручний доступ до інформації та інструментів, необхідних для успішної реалізації ландшафтних рішень[1].

Одним з найпопулярніших і добре відомих сервісів є "Houzz" (див. рис. 1.1).

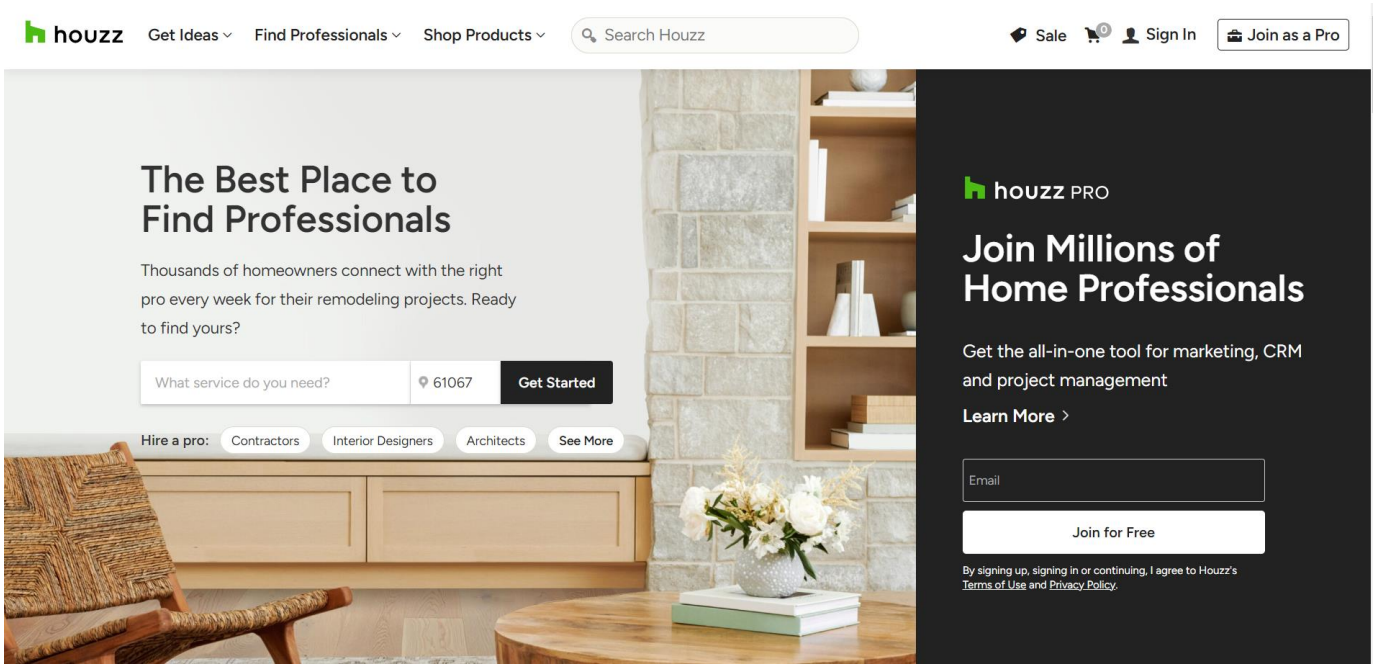

Рисунок 1.1 – Houzz (за даними [1])

Ця платформа надає користувачам доступ до великої бібліотеки ідей для домашнього та ландшафтного дизайну. Houzz пропонує інструменти для створення візуалізацій, що дозволяють користувачам побачити, як виглядатиме їхній простір після завершення проекту. Крім того, Houzz має розділ з відгуками та рейтингами професіоналів, що допомагає користувачам знайти та обрати найкращих виконавців для своїх проектів. Платформа також підтримує функцію інтерактивного спілкування між клієнтами та дизайнерами, що сприяє

ефективному обміну ідеями та забезпечує високу якість обслуговування.

Іншим прикладом є сервіс "SketchUp" (див. рис. 1.2), який пропонує потужні інструменти для тривимірного моделювання ландшафтних проектів. SketchUp використовується як професіоналами, так і аматорами для створення детальних моделей та планів територій[2].

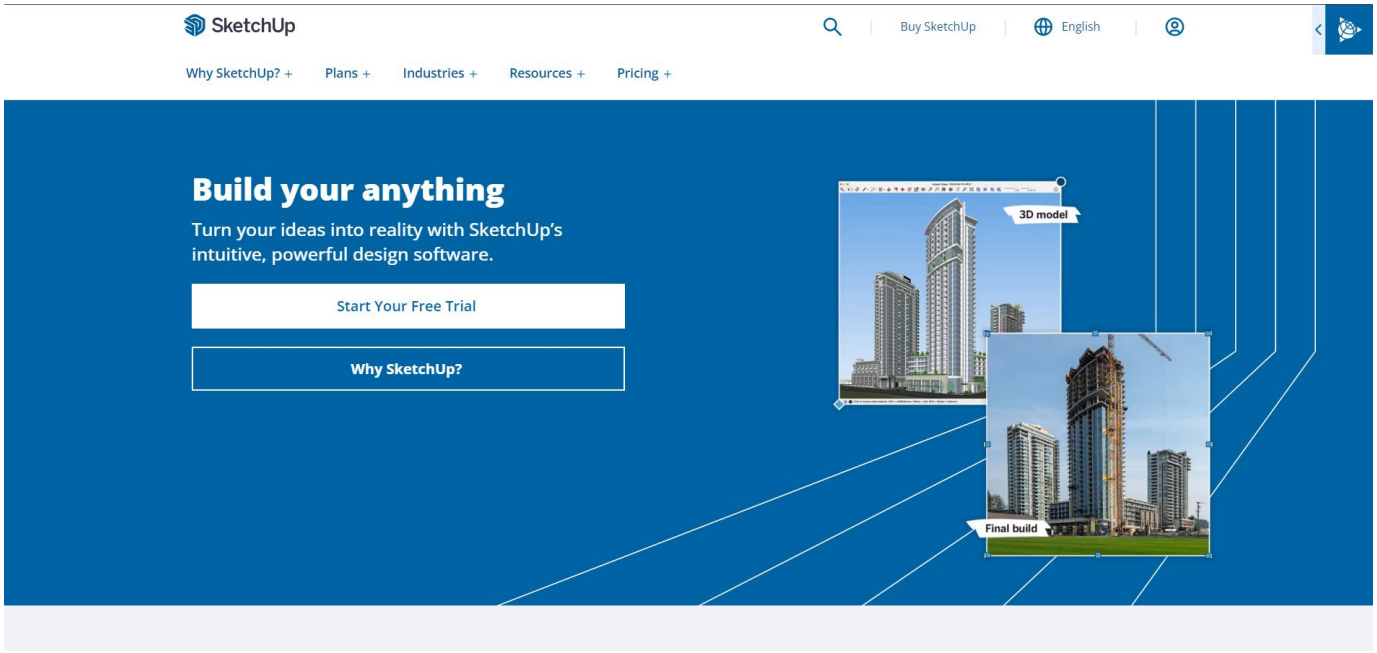

Рисунок 1.2 – SketchUp (за даними [2])

Платформа дозволяє інтегрувати різні елементи ландшафту, такі як рослини, водойми, архітектурні споруди, що робить процес проектування більш реалістичним та точним. Крім того, SketchUp має широкий спектр навчальних матеріалів та спільноту користувачів, що надає можливість отримувати консультації та обмінюватися досвідом.

Також варто згадати веб-сервіс "Pro Landscape" (див. рис. 1.3), який спеціалізується на професійному ландшафтному дизайні. Pro Landscape пропонує інструменти для створення планів та візуалізацій, а також має розширені можливості для управління проектами та ведення клієнтських баз даних. Цей сервіс надає доступ до бібліотек зображень рослин, матеріалів та об'єктів, що допомагає дизайнерам створювати високоякісні та деталізовані проекти[3].

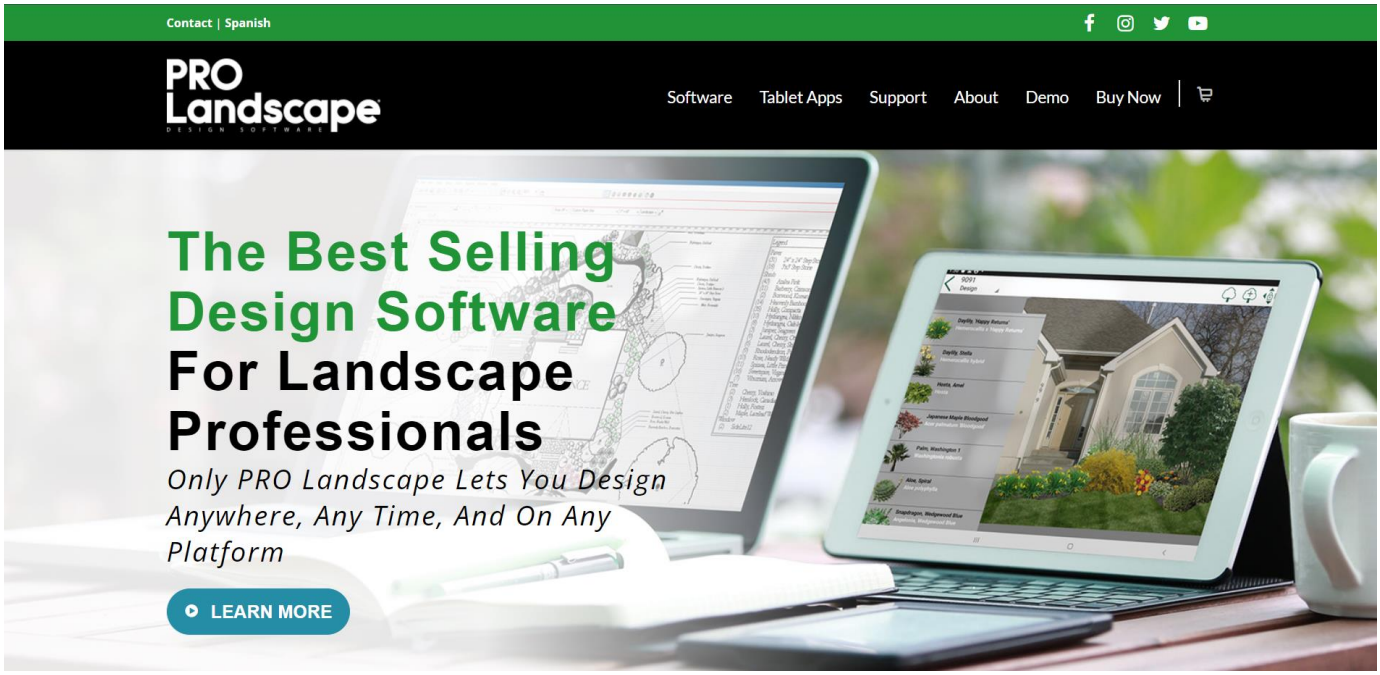

Рисунок 1.3 – Pro Landscape (за даними [3])

Крім того, Pro Landscape підтримує мобільні додатки, що дозволяє дизайнерам працювати над проектами безпосередньо на об'єкті, використовуючи планшети або смартфони.

Крім зазначених вище веб-сервісів, варто згадати такі інструменти, як "Garden Planner" та "Realtime Landscaping Architect". "Garden Planner" – це простий у використанні інструмент, який дозволяє користувачам створювати детальні плани садів та ландшафтних проектів. Відмінною рисою цього сервісу є можливість додавати до проекту не лише рослини та дерева, але й різноманітні садові меблі, доріжки, огорожі та інші елементи ландшафтного дизайну. "Garden Planner" також має вбудовану бібліотеку рослин, яка включає докладну інформацію про кожну рослину, що допомагає користувачам правильно планувати посадки та догляд за садом.

"Realtime Landscaping Architect" є професійним інструментом для створення тривимірних моделей ландшафтних проектів. Цей програмний продукт дозволяє дизайнерам створювати реалістичні візуалізації, використовуючи різноманітні елементи ландшафту, включаючи водойми, тераси, альтанки та інші архітектурні споруди. "Realtime Landscaping Architect" підтримує функцію імпорту реальних

зображень, що дозволяє дизайнерам створювати проекти на основі реальних фотографій території. Крім того, цей інструмент має розширені можливості для створення презентацій та анімацій, що допомагає дизайнерам ефективно представляти свої проекти клієнтам.

Існує також багато інших рішень, таких як "Landscaping Network" та "iScape", які забезпечують користувачів різноманітними інструментами та ресурсами для ландшафтного дизайну. Всі ці платформи мають свої унікальні особливості та переваги, але основною спільною рисою є їхнє прагнення спростити процес проектування та реалізації ландшафтних рішень, зробивши його доступним для широкого кола користувачів.

Аналізуючи існуючі рішення, можна зробити висновок, що розробка веб-сервісу "Flexible landscape design" є актуальною та затребуваною. Відмінною рисою даного сервісу буде інтеграція всіх найкращих практик та інноваційних підходів, які використовуються в сучасних платформах, з метою створення більш гнучкого та зручного інструменту для ландшафтного дизайну.

"Flexible landscape design" буде орієнтований на забезпечення високої якості обслуговування, зручності користування та максимальної адаптивності до потреб як професіоналів, так і аматорів у сфері ландшафтного дизайну.

1.2 Визначення потреб користувачів

Одним із ключових аспектів розробки веб-сервісу "Flexible landscape design" є глибоке розуміння потреб і очікувань потенційних користувачів. Виявлення цих потреб дозволить створити систему, яка максимально відповідає вимогам цільової аудиторії, забезпечуючи ефективність та зручність у використанні.

Перш за все, користувачі веб-сервісу "Flexible landscape design" можуть бути розділені на кілька основних категорій. Серед них – професійні ландшафтні дизайнери, компанії, що займаються ландшафтними роботами, та кінцеві споживачі, які шукають якісні послуги з озеленення та благоустрою своїх територій. Кожна з цих груп має свої специфічні потреби та очікування від системи.

Професійні ландшафтні дизайнери, які використовуватимуть сервіс, потребують зручного та функціонального інструменту для створення та презентації своїх проектів. Вони очікують від системи можливості легкої інтеграції різноманітних елементів дизайну, таких як рослини, архітектурні споруди, водні об'єкти та освітлення. Крім того, важливою є наявність функції тривимірного моделювання, яка дозволить створювати детальні та реалістичні візуалізації проектів. Дизайнери також очікують на можливість швидкого редагування та коригування своїх проектів, що забезпечить гнучкість та адаптивність у роботі з клієнтами.

Компанії, що займаються ландшафтними роботами, потребують платформи, яка дозволить їм ефективно презентувати свої послуги та успішно конкурувати на ринку. Вони очікують, що веб-сервіс надасть можливість створювати привабливі портфоліо з прикладами виконаних робіт, які можуть бути легко переглянуті потенційними клієнтами. Крім того, важливою є інтеграція функції відгуків та рейтингів, що дозволить компаніям демонструвати свою репутацію та залучати нових клієнтів. Також компанії зацікавлені у наявності інструментів для управління проектами та ведення клієнтських баз даних, що сприятиме організації та ефективності їхньої роботи.

Кінцеві споживачі, які шукають послуги з ландшафтного дизайну, очікують від веб-сервісу зручності та доступності інформації. Вони хочуть мати можливість легко знайти та ознайомитися з пропозиціями різних компаній, переглянути приклади виконаних робіт та отримати детальну інформацію про кожну послугу. Важливою є також наявність функції інтерактивного спілкування з дизайнерами та компаніями, яка дозволить їм отримати консультації, задати питання та замовити послуги безпосередньо через платформу. Крім того, кінцеві споживачі очікують, що веб-сервіс буде мати адаптивний дизайн, який забезпечить зручний доступ з будь-якого пристрою, включаючи мобільні телефони та планшети.

Враховуючи ці потреби та очікування, розробка веб-сервісу "Flexible landscape design" повинна бути спрямована на створення інтуїтивно зрозумілого та функціонального інтерфейсу, який забезпечить високу якість обслуговування та

задоволення користувачів. Особлива увага повинна бути приділена забезпеченню надійності та безпеки системи, що є важливим фактором для збереження довіри користувачів. Таким чином, глибоке розуміння та врахування потреб кожної категорії користувачів дозволить створити ефективну та затребувану платформу, яка сприятиме розвитку та успіху компаній у сфері ландшафтного дизайну.

1.3 Виявлення прогалин у наявних рішеннях

При аналізі існуючих рішень у сфері ландшафтного дизайну виявляються певні прогалини та слабкі місця, які потребують удосконалення. Незважаючи на те, що сучасні веб-сервіси пропонують широкий спектр функціональних можливостей[4], багато з них не задовольняють усіх потреб користувачів повною мірою. Розробка веб-сервісу "Flexible landscape design" спрямована на подолання цих недоліків та створення більш досконалого інструменту для ландшафтного дизайну.

Однією з основних проблем, з якою стикаються користувачі існуючих систем, є недостатня гнучкість інструментів для проектування[5]. Багато платформ обмежують дизайнерів у виборі елементів та їхній комбінації, що може призводити до створення стандартних та передбачуваних проектів. Крім того, деякі системи не дозволяють інтегрувати специфічні деталі та індивідуальні побажання клієнтів, що знижує якість та індивідуальність кінцевих рішень. Веб-сервіс "Flexible landscape design" має на меті надати дизайнерам максимальну свободу у створенні та редагуванні проектів, забезпечуючи інтуїтивно зрозумілий інтерфейс та потужні інструменти для налаштування.

Іншою суттєвою прогалиною є недостатня інтерактивність та взаємодія між користувачами. Багато існуючих платформ не забезпечують ефективних засобів комунікації між дизайнерами та клієнтами, що може призводити до непорозумінь та затримок у виконанні проектів. Веб-сервіс "Flexible landscape design" планується створити інтерактивні функції для обміну повідомленнями, проведення відеоконференцій та спільного редагування проектів у режимі реального часу. Це сприятиме більш тісній співпраці між сторонами та забезпечить своєчасне внесення

змін та коригувань.

Також варто зазначити, що багато веб-сервісів не надають достатньої кількості навчальних ресурсів та підтримки для користувачів, особливо для новачків. Відсутність докладної документації, навчальних відео та підтримки з боку спільноти може стати серйозною перепоною для тих, хто лише починає працювати у сфері ландшафтного дизайну. Веб-сервіс "Flexible landscape design" передбачає створення обширної бази знань, яка включатиме детальні керівництва, навчальні матеріали та форуми для обміну досвідом. Це допоможе користувачам швидко освоїти функціональні можливості системи та підвищити свою кваліфікацію.

Ще одним важливим аспектом є питання безпеки та конфіденційності даних. Деякі існуючі системи не забезпечують належного захисту персональних даних користувачів та конфіденційної інформації про проекти[6]. Це може стати серйозною проблемою, особливо для великих компаній, які працюють над комерційними проектами. Веб-сервіс "Flexible landscape design" має включати сучасні засоби захисту даних, такі як шифрування, багатофакторна аутентифікація та регулярні оновлення системи безпеки. Це забезпечить надійний захист інформації та підвищить довіру користувачів до платформи.

Нарешті, багато існуючих рішень не надають можливості інтеграції з іншими системами та додатками, що може обмежувати їхню функціональність. Веб-сервіс "Flexible landscape design" буде розроблений з урахуванням можливості інтеграції з іншими популярними програмними продуктами та сервісами, що дозволить користувачам використовувати різноманітні інструменти у своїй роботі та забезпечить більш високу ефективність процесів.

Таким чином, веб-сервіс "Flexible landscape design" спрямований на подолання основних прогалин існуючих рішень, забезпечуючи високу гнучкість, інтерактивність, підтримку користувачів, безпеку та інтеграційні можливості. Це дозволить створити досконалий інструмент для ландшафтного дизайну, який відповідатиме сучасним вимогам та потребам користувачів.

1.4 Перспективи розвитку веб-сервісів у сфері ландшафтного дизайну

Розвиток технологій у сфері ландшафтного дизайну не стоїть на місці, і з кожним роком з'являються нові інструменти та рішення, які роблять процес проектування більш ефективним та зручним. Одним із ключових напрямів розвитку є інтеграція штучного інтелекту (ШІ) та машинного навчання (МН) у веб-сервіси для ландшафтного дизайну. Використання ШІ дозволяє автоматизувати рутинні завдання, такі як розміщення рослин та елементів дизайну, що значно скорочує час на створення проектів. Крім того, МН може аналізувати попередні проекти та надавати рекомендації щодо оптимізації дизайну, враховуючи кліматичні умови та особливості території.

Іншим перспективним напрямом є розвиток доповненої реальності (AR) та віртуальної реальності (VR). Веб-сервіси, що підтримують AR та VR, дозволяють користувачам створювати тривимірні моделі своїх проектів та переглядати їх у реальному середовищі. Це дозволяє дизайнерам та клієнтам отримати більш точне уявлення про те, як виглядатиме кінцевий результат, та вносити необхідні корективи на ранніх етапах проектування. Використання AR та VR також сприяє підвищенню залученості клієнтів у процес створення дизайну, що покращує їхнє задоволення від співпраці.

Не менш важливим є розвиток мобільних додатків для ландшафтного дизайну, наприклад iScape (див. рис. 1.4). Сучасні мобільні додатки дозволяють дизайнерам працювати над проектами безпосередньо на об'єкті, використовуючи планшети або смартфони. Це забезпечує більшу гнучкість та оперативність у роботі, дозволяючи швидко вносити зміни та узгоджувати їх з клієнтами. Крім того, мобільні додатки часто мають інтеграцію з хмарними сервісами, що забезпечує надійне збереження даних та можливість спільної роботи над проектами з різних пристроїв.

Також перспективним напрямом є розвиток екологічних та стійких рішень у ландшафтному дизайні. Веб-сервіси можуть надавати інструменти для оцінки екологічного впливу проектів, пропонуючи альтернативні рішення, які сприяють збереженню природних ресурсів та покращенню екологічного стану території. Це

може включати використання місцевих рослин, впровадження систем збору дощової води, встановлення сонячних панелей та інших екологічних технологій. Врахування екологічних аспектів у процесі проектування стає все більш важливим, оскільки все більше клієнтів віддають перевагу стійким та екологічно дружнім рішенням.

Рисунок 1.4 – iScape mobile (за даними [3])

Таким чином, розвиток веб-сервісів у сфері ландшафтного дизайну має великий потенціал та спрямований на підвищення ефективності, зручності та екологічності процесу проектування. Інтеграція новітніх технологій, таких як ШІ, AR/VR та мобільні додатки, а також увага до екологічних аспектів дозволить створити більш досконалі та затребувані інструменти для ландшафтного дизайну.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

2.1 Функціональні вимоги

Функціональні вимоги до веб-сервісу "Flexible landscape design" визначають основні функції, які повинна виконувати система для задоволення потреб користувачів. Враховуючи потреби різних груп користувачів, система повинна забезпечувати широкий спектр функцій, що дозволяють ефективно презентувати послуги, взаємодіяти з клієнтами та підтримувати високу якість наданих послуг[7].

Перш за все, веб-сервіс "Flexible landscape design" повинен мати зручну та функціональну головну сторінку, яка містить навігаційне меню для швидкого доступу до основних розділів сайту. Навігаційне меню повинно включати посилання на розділи "Home", "Services", "Work" та "Contact". Користувачі повинні мати можливість легко переходити між розділами сайту, забезпечуючи зручність та інтуїтивність користування.

Основним розділом веб-сервісу є розділ "Services", де повинні бути детально описані всі послуги, що надає компанія. Кожна послуга повинна мати власну картку, яка включає зображення, заголовок та текстовий опис. Ці картки повинні надавати вичерпну інформацію про кожну послугу, допомагаючи користувачам зрозуміти її суть та переваги. Крім того, повинна бути можливість для компаній додавати нові послуги та редагувати існуючі, що забезпечить актуальність інформації.

Розділ "Previous Work" повинен містити галерею зображень, що демонструють приклади виконаних робіт. Галерея повинна бути зручною для перегляду та включати функцію збільшення зображень для детального огляду. Це дозволить потенційним клієнтам оцінити якість виконаних робіт та прийняти обґрунтоване рішення щодо вибору компанії.

Розділ "Contact" є важливим елементом веб-сервісу, оскільки він забезпечує зворотний зв'язок між компанією та клієнтами. У цьому розділі повинна бути представлена детальна контактна інформація, включаючи номер телефону, електронну адресу та посилання на соціальні мережі. Також необхідно інтегрувати контактну форму, яка дозволить користувачам швидко і легко зв’язатися з

компанією для отримання консультацій або замовлення послуг. Форма повинна включати поля для введення імені, електронної адреси, номера телефону та повідомлення.

Розділ "Custom Design" у веб-сервісі "Flexible landscape design" є однією з ключових функцій, що надає користувачам можливість створювати та редагувати власні ландшафтні проекти. Цей розділ забезпечує високу інтерактивність, дозволяючи користувачам візуально проектувати ландшафти, додаючи та переміщуючи рослини, елементи декору та інші об'єкти на канвасі. Основною метою цього розділу є забезпечення зручності та гнучкості у процесі проектування, що сприяє кращому плануванню та реалізації ідей користувачів.

Крім того, веб-сервіс повинен підтримувати функцію реєстрації та авторизації користувачів. Зареєстровані користувачі матимуть доступ до додаткових функцій, таких як створення та редагування профілю, збереження обраних послуг та проектів, а також можливість залишати відгуки про послуги компанії. Функція реєстрації повинна бути простою та зручною, включаючи можливість реєстрації через соціальні мережі.

Для забезпечення зручності використання веб-сервісу, необхідно передбачити функцію локалізації, що дозволить користувачам обирати мову інтерфейсу. Це забезпечить доступність сервісу для широкої аудиторії, включаючи користувачів з різних країн[8].

Важливим аспектом є забезпечення безпеки та конфіденційності даних користувачів. Веб-сервіс повинен включати механізми захисту даних, такі як шифрування інформації, багатофакторна аутентифікація та регулярні оновлення системи безпеки. Це забезпечить надійний захист персональних даних та підвищить довіру користувачів до платформи.

Таким чином, основними функціональними вимогами до веб-сервісу "Flexible landscape design" є:

- локалізація (укр/eng);
- реєстрація та авторизація користувачів;
- створення та редагування профілів користувачів;

- перегляд та редагування послуг компанії;
- інтерактивна галерея виконаних робіт;
- детальна контактна інформація та інтегрована контактна форма;
- підтримка відгуків та рейтингів послуг;
- забезпечення безпеки та конфіденційності даних;
- можливість редагування власного дизайну на канвасі шляхом перетягування моделей рослин та елементів декору по полю.

Реалізація цих функцій дозволить створити сучасний, зручний та надійний веб-сервіс, який задовольнить потреби як професійних ландшафтних дизайнерів та компаній, так і кінцевих споживачів, забезпечуючи високу якість наданих послуг.

2.2 Нефункціональні вимоги

Нефункціональні вимоги визначають параметри, які не пов'язані безпосередньо з функціональністю системи, але є критичними для забезпечення її якості, надійності та зручності використання. Для веб-сервісу "Flexible landscape design" важливо врахувати такі аспекти, як надійність, ефективність, зручність користування, масштабованість, безпека та підтримка[9].

Більш конкретні вимоги наведено у таблиці 2.1

Таблиця 2.1 – Чисельні значення вимог

Вимога	Пояснення	Числове значення
Час безвідмовної роботи (Uptime)	Безперебійне функціонування системи	Не менше 99.9% на рік
Частота резервного копіювання даних	Регулярне збереження даних	Щоденно
Час відновлення після збоїв	Швидке відновлення роботи після непередбачених ситуацій	Не більше 1 години

Продовження таблиці 2.1

Моніторинг системи	Постійне відстеження стану системи з автоматичними сповіщеннями	24/7 з автоматичними сповіщеннями
Час завантаження сторінки	Швидке завантаження веб-сторінок	Не більше 3 секунд
Час відгуку сервера на запит	Швидка обробка запитів користувачів	Не більше 200 мілісекунд
Кількість одночасних користувачів	Підтримка великої кількості одночасних з’єднань	До 1000 одночасних з’єднань
Використання технологій оптимізації	Підвищення ефективності за допомогою кешування, стиснення даних	Кешування, стиснення даних, використання CDN
Інтуїтивно зрозумілий інтерфейс	Зручний та логічно структурований інтерфейс	Проведення юзабіліті-тестування
Адаптивний дизайн	Коректне відображення на різних пристроях	Адаптація для настільних комп'ютерів, планшетів, мобільних телефонів
Доступність для людей з обмеженими можливостями	Використання семантичного HTML, підтримка технологій екранного читання	Дотримання стандартів доступності
Здатність витримувати навантаження	Можливість додавання нових функцій та розширення існуючих	Використання модульної архітектури

Кінець таблиці 2.1

Захист персональних даних	Захист від несанкціонованого доступу та забезпечення конфіденційності	Аутентифікація, авторизація, шифрування, захист від SQL-ін'єкцій та XSS-атак
Стабільна робота та швидке вирішення проблем	Забезпечення постійної підтримки користувачів	Технічна підтримка, оновлення, навчання

Веб-сервіс "Flexible landscape design" повинен забезпечувати високу надійність, що означає безперебійне функціонування та доступність системи для користувачів у будь-який час. Для досягнення цього необхідно реалізувати механізми резервного копіювання та відновлення даних, які дозволять швидко відновити роботу системи у разі виникнення збоїв чи непередбачених ситуацій. Також важливо забезпечити моніторинг системи для виявлення та усунення потенційних проблем до їхнього виникнення.

Ефективність роботи веб-сервісу полягає у швидкому завантаженні сторінок, обробці запитів та відгуку на дії користувачів[10]. Веб-сервіс повинен бути оптимізованим для мінімізації часу завантаження, навіть при великому обсязі даних та великій кількості одночасних користувачів. Для цього необхідно використовувати сучасні технології та методи оптимізації, такі як кешування, стиснення даних та використання CDN (Content Delivery Network).

Зручність користування є критичним параметром для забезпечення позитивного досвіду користувачів. Інтерфейс веб-сервісу повинен бути інтуїтивно зрозумілим, зручним та логічно структурованим. Для досягнення цього важливо провести юзабіліті-тестування з залученням представників цільової аудиторії та врахувати їхні відгуки та побажання. Крім того, дизайн інтерфейсу повинен бути адаптивним, що забезпечить коректне відображення та функціонування на різних пристроях, включаючи настільні комп'ютери, планшети та мобільні телефони.

Веб-сервіс "Flexible landscape design" повинен бути масштабованим, тобто

здатним витримувати збільшення навантаження при зростанні кількості користувачів та обсягу даних[11]. Це передбачає можливість додавання нових функцій та розширення існуючих без суттєвого впливу на продуктивність системи. Для забезпечення масштабованості необхідно використовувати модульну архітектуру та гнучкі рішення для управління ресурсами.

Безпека є одним з ключових аспектів будь-якої веб-системи. Веб-сервіс "Flexible landscape design" повинен забезпечувати надійний захист персональних даних користувачів та конфіденційної інформації про проекти. Це включає реалізацію механізмів аутентифікації та авторизації користувачів, шифрування даних, захист від SQL-ін'єкцій та XSS-атак, а також регулярні оновлення системи безпеки. Крім того, важливо проводити регулярні аудити безпеки для виявлення та усунення потенційних вразливостей.

Веб-сервіс повинен мати чітко визначену систему підтримки та обслуговування для забезпечення стабільної роботи та швидкого вирішення проблем користувачів. Це включає надання технічної підтримки, оновлення та вдосконалення системи, а також регулярне проведення навчань та інструктажів для користувачів. Важливо забезпечити доступність документації, керівництв та інших ресурсів, які допоможуть користувачам ефективно використовувати веб-сервіс.

Веб-сервіс "Flexible landscape design" повинен бути доступним для максимально широкого кола користувачів, включаючи людей з обмеженими можливостями. Це передбачає дотримання принципів доступності, таких як використання семантичного HTML, підтримка технологій екранного читання, забезпечення контрастності тексту та фону, а також наявність альтернативних текстів для зображень[12].

Врахування цих нефункціональних вимог дозволить створити надійний, ефективний та зручний веб-сервіс "Flexible landscape design", який задовольнить потреби користувачів та забезпечить високу якість наданих послуг.

2.3 Вимоги до інтерфейсу користувача

Інтерфейс користувача веб-сервісу "Flexible landscape design" має бути

інтуїтивно зрозумілим, естетично привабливим та функціональним. Це означає, що кожен елемент інтерфейсу повинен мати чітке призначення, бути легко доступним та сприяти ефективній взаємодії користувача з системою. Інтерфейс повинен бути адаптивним, забезпечуючи коректне відображення на різних пристроях, включаючи настільні комп’ютери, планшети та мобільні телефони[13]. Крім того, важливо дотримуватися принципів доступності для користувачів з обмеженими можливостями.

Головна сторінка веб-сервісу "Flexible landscape design" повинна включати чітке та логічне навігаційне меню, розташоване у верхній частині сторінки. Це меню повинно містити наступні розділи: "Home", "Services", "Work" та "Contact". Кожне посилання в меню повинно бути чітко виділеним і легко натискатися. Крім того, меню повинно залишатися фіксованим при прокручуванні сторінки, забезпечуючи швидкий доступ до будь-якої частини сайту в будь-який момент.

У верхній частині головної сторінки повинно бути розміщено логотип компанії та девіз "Your Lawn, Our Passion, Your Satisfaction". Поруч з девізом повинні бути кнопки для швидкого переходу до розділів "Services" та "Previous Work". Ці кнопки повинні бути достатньо великими та добре видимими, щоб користувачі могли легко знайти та натиснути їх.

Розділ "Services" повинен містити детальний опис усіх послуг, що надаються компанією. Кожна послуга повинна бути представлена у вигляді картки, яка включає зображення, заголовок та текстовий опис. Зображення повинні бути високоякісними та релевантними до описуваних послуг, що допоможе користувачам краще зрозуміти суть кожної послуги. Текстовий опис повинен бути чітким та інформативним, виділяючи ключові аспекти та переваги кожної послуги.

Картки послуг повинні бути розташовані в логічному порядку, наприклад, за категоріями або типами послуг. Це допоможе користувачам швидко знайти необхідну інформацію. Крім того, важливо передбачити можливість розширення цього розділу для додавання нових послуг або редагування існуючих.

Розділ "Previous Work" повинен містити галерею зображень, що демонструють виконані проекти компанії. Галерея повинна бути зручною для

перегляду та включати функцію збільшення зображень для детального огляду. Кожне зображення повинно мати підпис, що описує проект, який воно представляє. Це допоможе користувачам краще оцінити якість виконаних робіт та прийняти обґрунтоване рішення щодо вибору компанії.

Розділ "Contact" є важливим для забезпечення зворотного зв’язку між компанією та клієнтами. Він повинен включати детальну контактну інформацію, таку як номер телефону, електронну адресу та посилання на соціальні мережі. Крім того, у цьому розділі повинна бути інтегрована контактна форма, що дозволить користувачам швидко і легко зв’язатися з компанією. Форма повинна включати поля для введення імені, електронної адреси, номера телефону та повідомлення. Всі поля повинні бути обов’язковими для заповнення, а кнопка відправки повинна бути чітко видимою та легко натискатися.

Центральним елементом інтерфейсу є великий інтерактивний канвас, де користувачі можуть візуально розміщувати об'єкти. Канвас повинен мати високу роздільну здатність, щоб забезпечити чіткість зображень та детальність проекту. З лівого боку або зверху канвасу розташована панель інструментів, яка містить список доступних для додавання об'єктів. Об'єкти поділені на категорії, такі як "Plants" і "Houses", що полегшує їх пошук та використання.

Користувачі можуть вибирати об'єкти з панелі інструментів і перетягувати їх на канвас. Це забезпечує інтерактивність і дозволяє користувачам легко розміщувати елементи на проекті. Об'єкти, розміщені на канвасі, можуть бути переміщені, змінені за розміром або видалені. Це дає користувачам можливість точно налаштувати розташування кожного елемента відповідно до їхнього бачення.

Для зареєстрованих користувачів повинен бути доступний розділ профілю, де вони можуть переглядати та редагувати свою особисту інформацію, зберігати обрані послуги та проекти, а також залишати відгуки про послуги компанії. Інтерфейс профілю повинен бути простим та інтуїтивно зрозумілим, забезпечуючи легкий доступ до всіх необхідних функцій.

Веб-сервіс "Flexible landscape design" повинен мати адаптивний дизайн, що дозволяє коректно відображатися на різних пристроях. Всі елементи інтерфейсу

повинні автоматично підлаштовуватися під розмір екрану, забезпечуючи зручність використання на мобільних телефонах, планшетах та настільних комп’ютерах.

Також важливо забезпечити доступність для користувачів з обмеженими можливостями. Це включає дотримання стандартів доступності, таких як використання семантичного HTML для структурування контенту, забезпечення достатнього контрасту між текстом і фоном, а також надання альтернативних текстів для зображень. Крім того, всі інтерактивні елементи, такі як кнопки та форми, повинні підтримувати навігацію з клавіатури.

Таким чином, веб-сервіс "Flexible landscape design" повинен забезпечувати високий рівень інтуїтивності та доступності інтерфейсу користувача, що сприятиме позитивному досвіду використання та задоволенню потреб усіх категорій користувачів.

2.4 Вимоги до бекенд частини

Бекенд частина веб-сервісу "Flexible landscape design" виконує критичну роль у забезпеченні функціональності, безпеки та надійності системи. Вона відповідає за обробку даних, управління користувачами, забезпечення безпеки та взаємодію з базою даних.

Бекенд частина повинна бути побудована на основі сучасних архітектурних підходів, таких як мікросервісна або серверлесс архітектура, для забезпечення масштабованості та гнучкості. Використання таких технологій, як Node.js, Express.js, plain js забезпечить високу продуктивність та ефективність обробки запитів.

Система повинна підтримувати функції реєстрації та авторизації користувачів. Це включає:

- реєстрація: користувачі повинні мати можливість створювати облікові записи, заповнюючи форму з особистими даними (ім'я, електронна пошта, пароль, тощо). Після реєстрації користувач отримує електронний лист для підтвердження адреси електронної пошти;
- управління профілем: користувачі повинні мати можливість редагувати

свою особисту інформацію, змінювати пароль та переглядати історію своїх дій на платформі.

Бекенд частина повинна забезпечувати функціональність для управління контентом веб-сервісу, включаючи послуги, галереї робіт та контактну інформацію. Це включає:

- CRUD операції для послуг: адміністратори повинні мати можливість створювати, читати, оновлювати та видаляти інформацію про послуги;
- галерея робіт: адміністратори повинні мати можливість додавати нові зображення в галерею, видаляти або змінювати існуючі;
- контактна інформація: адміністратори повинні мати можливість оновлювати контактну інформацію компанії.

Система повинна забезпечувати обробку запитів від користувачів через контактну форму. Це включає:

- валідація даних: дані, що вводяться користувачами в контактну форму, повинні проходити перевірку на коректність та повноту;
- збереження повідомлень: повідомлення від користувачів повинні зберігатися в базі даних для подальшої обробки;
- сповіщення адміністрації: адміністратори повинні отримувати сповіщення про нові повідомлення електронною поштою або через інші канали.

Безпека є одним з найважливіших аспектів бекенд частини. Система повинна включати наступні механізми безпеки:

- шифрування даних: всі конфіденційні дані, такі як паролі, повинні зберігатися у зашифрованому вигляді;
- багатофакторна аутентифікація: для підвищення рівня безпеки користувачів повинна підтримуватися багатофакторна аутентифікація;
- захист від атак: система повинна бути захищена від типових веб-атак, таких як XSS, CSRF. Це може бути досягнуто за допомогою використання фреймворків з вбудованими засобами безпеки та регулярних аудитів безпеки.

Бекенд частина повинна ефективно взаємодіяти з базою даних, забезпечуючи

швидкий доступ до даних та їх збереження. Вимоги до бази даних включають:

- NoSQL бази даних: використання таких баз даних, як MongoDB;
- резервне копіювання: регулярне резервне копіювання даних для запобігання втратам інформації;
- масштабованість: база даних повинна бути здатною обробляти великі обсяги даних та високі навантаження.

Бекенд частина повинна надавати API для взаємодії з фронтендом. Це включає RESTful API, що забезпечує доступ до всіх необхідних даних та функцій. Важливо забезпечити:

- доступність: API повинне бути доступним та швидким у відповідях;
- документація: всі кінцеві точки API повинні бути детально задокументовані для зручності розробників фронтенду;
- аутентифікація та авторизація: доступ до API повинен бути захищений, з використанням токенів доступу та інших механізмів аутентифікації.

Для забезпечення стабільної роботи та швидкого вирішення проблем необхідно реалізувати механізми логування та моніторингу:

- логування подій: запис всіх важливих подій та помилок для подальшого аналізу;
- моніторинг продуктивності: відстеження продуктивності системи, використання ресурсів та інших показників;
- сповіщення про збої: автоматичне сповіщення адміністрації про критичні збої та помилки.

Щоб забезпечити стійкість та ефективну роботу веб-сервісу "Flexible landscape design", необхідно розглянути такі аспекти:

- масштабованість: бекенд частина повинна бути здатною масштабуватися горизонтально для відповіді на зростаюче навантаження;
- тестування: необхідно ретельно тестувати бекенд частину для виявлення та виправлення помилок перед їх виявленням користувачами;
- моніторинг та аналіз даних: важливо встановити механізми моніторингу, щоб відстежувати продуктивність системи, виявляти проблеми та

приймати швидкі заходи щодо їх вирішення. Аналіз даних також може надати корисну інформацію щодо використання платформи та потреб користувачів;

- безпека: потрібно враховувати всі аспекти безпеки, включаючи захист від SQL-ін'єкцій, витоків даних та інших загроз. Проведення регулярних аудитів безпеки та впровадження відповідних заходів захисту допоможе запобігти можливим атакам;
- резервне копіювання та відновлення: необхідно регулярно робити резервні копії даних та мати механізми швидкого відновлення в разі втрати або пошкодження даних.

Щоб забезпечити надійну та стабільну роботу веб-сервісу "Flexible landscape design", важливо врахувати такі аспекти:

- кешування: впровадження кешування зменшення навантаження на базу даних і прискорення обробки запитів;
- контроль версій: використання Git для управління кодом та відстеження змін;
- документування: документування коду та створення коментарів для спрощення підтримки та розвитку системи;
- тестування безпеки: регулярне проведення тестів на проникнення для виявлення вразливостей;

Таким чином, бекенд частина веб-сервісу "Flexible landscape design" повинна забезпечувати високу продуктивність, надійність, безпеку та зручність в управлінні даними. Це дозволить створити ефективну та надійну платформу для надання послуг у сфері ландшафтного дизайну.

3 АРХІТЕКТУРА ТА ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 UML проектування системи

UML (Unified Modeling Language) проектування є важливим етапом у розробці програмного забезпечення, оскільки воно дозволяє візуалізувати структуру системи, її компоненти та взаємодії між ними. Для веб-сервісу "Flexible landscape design" використовуються кілька видів діаграм UML, включаючи діаграму варіантів використання, діаграму потоків даних та діаграму класів. Кожен з цих видів діаграм допомагає зрозуміти різні аспекти системи та їх взаємозв'язки.

3.1.1 Діаграма варіантів використання

Діаграма варіантів використання (use case diagram) показує взаємодію між користувачами (акторами) та системою, а також функціональні можливості, які надає система. Ця діаграма є основою для розуміння вимог до системи та визначення її функціональних можливостей.

У веб-сервісі "Flexible landscape design" основними акторами є користувачі (адміністратори, дизайнери, клієнти) та система. Діаграма варіантів використання включає такі основні сценарії:

- реєстрація користувача;
- вхід до системи;
- перегляд послуг;
- додавання/редагування послуг (адміністратори);
- перегляд проектів;
- додавання/редагування проектів (адміністратори);
- надсилання повідомлень через контактну форму;
- управління профілем користувача.

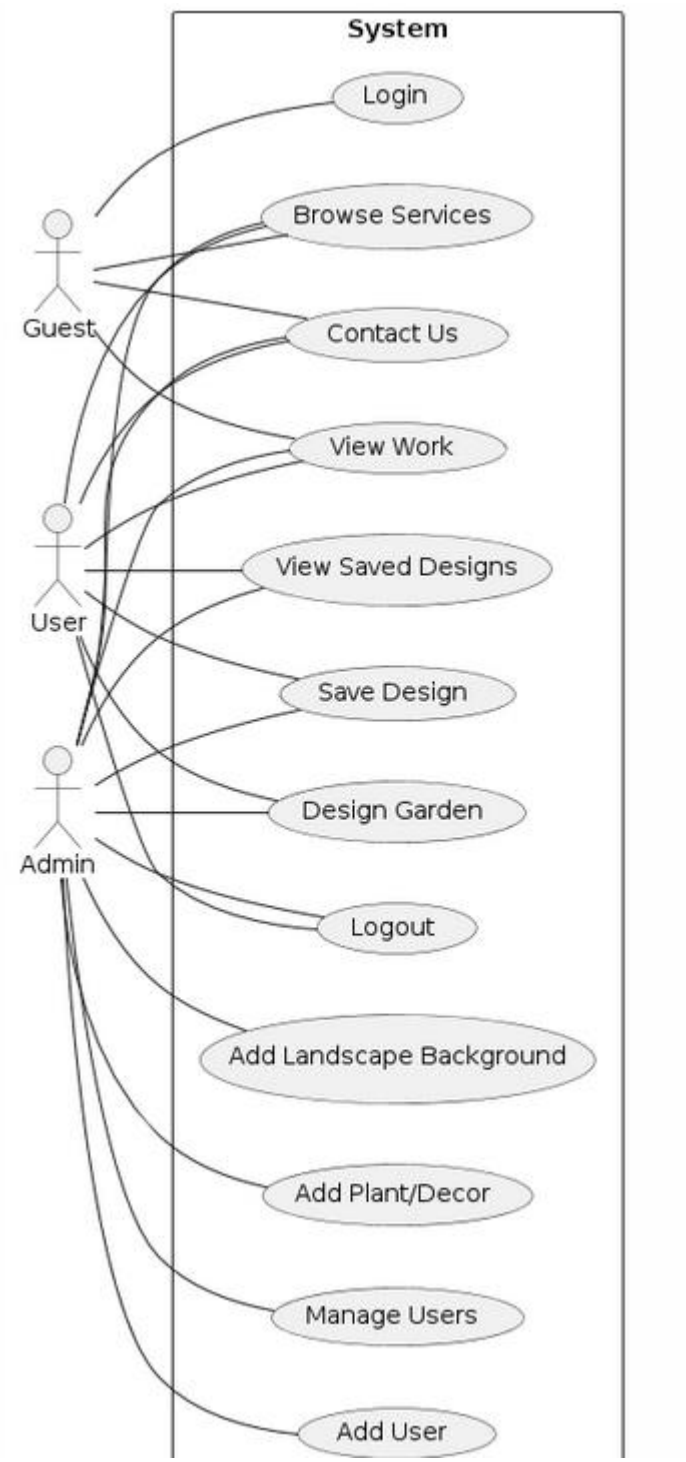

Рисунок 3.1 – Діаграма варіантів використання (рисунок виконано самостійно)

3.1.2 Діаграма потоків даних

Діаграма потоків даних (data flow diagram) ілюструє, як дані переміщуються всередині системи, які процеси обробляють ці дані та де вони зберігаються. Ця діаграма допомагає зрозуміти, як інформація передається між різними частинами

системи.

Для веб-сервісу "Flexible landscape design" діаграма потоків даних може включати наступні елементи:

– введення даних користувачем через форми реєстрації або авторизації
– обробка даних на сервері, включаючи валідацію та хешування паролів
– збереження даних у базі даних
– запити користувачів на перегляд послуг та проектів
– обробка запитів на сервері та отримання даних з бази даних
– відправка даних користувачам для відображення на веб-сторінках
– обробка контактних повідомлень та зберігання їх у базі даних

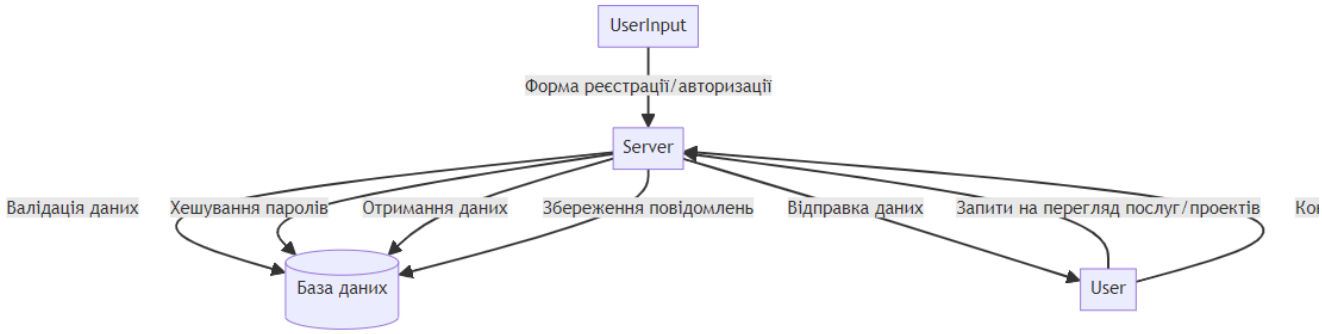

Рисунок 3.2 – Діаграма потоків даних (рисунок виконано самостійно)

3.1.3 Діаграма класів

Діаграма класів (class diagram) представляє структуру системи на рівні класів та їх взаємозв'язків. Вона показує атрибути класів, методи та асоціації між класами, що дозволяє зрозуміти логічну структуру програми.

Для веб-сервісу "Flexible landscape design" діаграма класів включає основні класи, такі як:

User: представляє користувача системи і включає атрибути, такі як id, name, email, password, phone_number, role, created_at, updated_at.

Service: представляє послугу і включає атрибути, такі як id, title, description, image_url, created_at, updated_at.

Project: представляє проект і включає атрибути, такі як id, title, description, image_url, created_at, updated_at.

ContactMessage: представляє контактне повідомлення і включає атрибути, такі як id, name, email, phone_number, message, created_at.

Взаємозв'язки між класами включають асоціації, наприклад, один користувач може мати кілька проектів, але кожен проект належить лише одному користувачу.

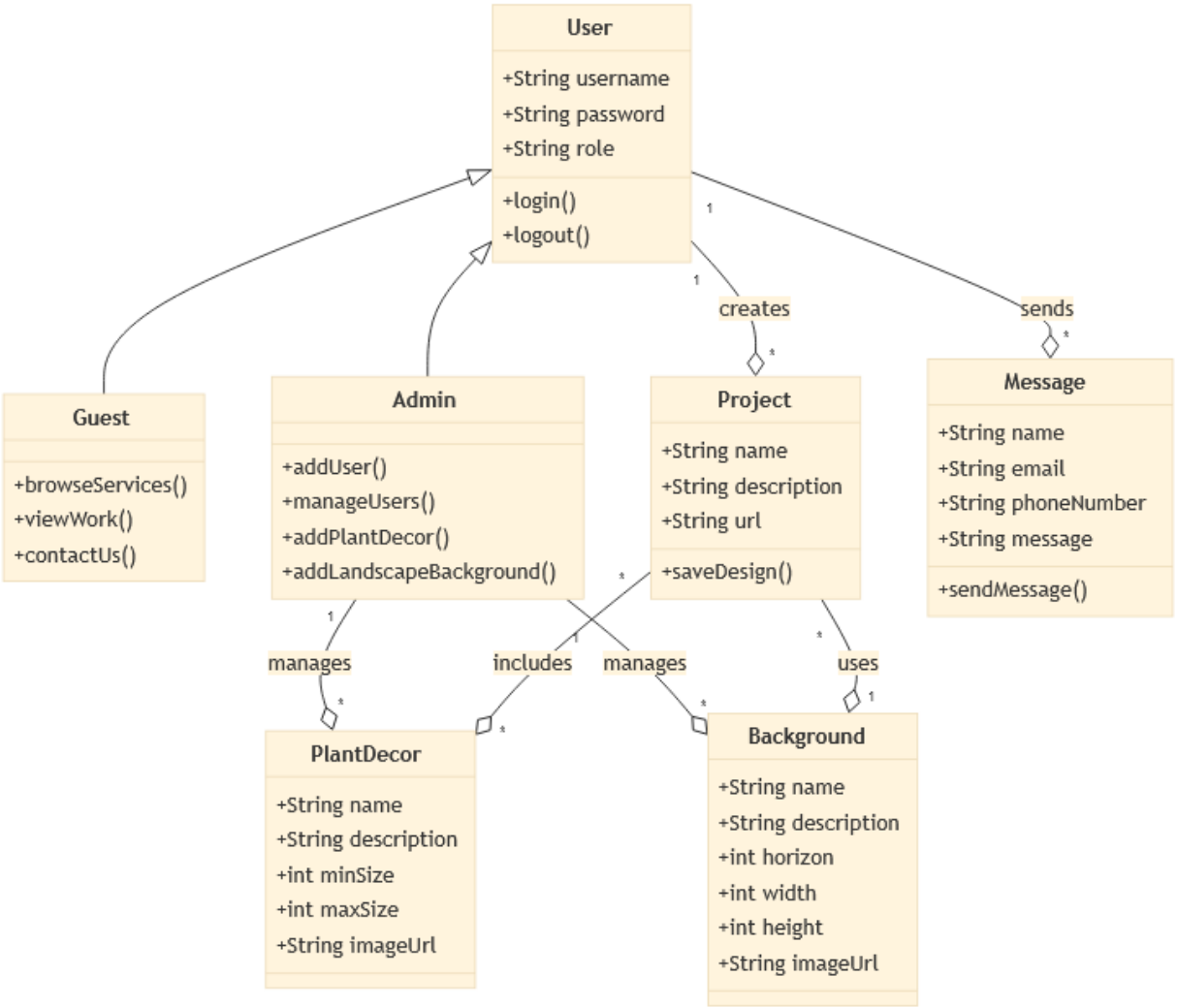

Рисунок 3.3 – Діаграма класів (рисунок виконано самостійно)

Діаграма класів наочно показує, як різні об'єкти в системі взаємодіють між собою та як вони пов'язані. Клас User має асоціацію з класом Project, що вказує на те, що кожен проект належить конкретному користувачеві. Клас Service не має прямих зв'язків з іншими класами в даній діаграмі, оскільки він самостійно

представляє послуги, що надаються компанією. Клас ContactMessage також не має прямих зв'язків з іншими класами, але він є важливим для обробки запитів користувачів через контактну форму.

Використання UML діаграм у процесі розробки системи "Flexible landscape design" дозволяє детально спланувати архітектуру програми, зрозуміти взаємозв'язки між її компонентами та забезпечити послідовну реалізацію всіх необхідних функцій.

3.1.4 Діаграма послідовності дій

Діаграма послідовності дій (sequence diagram) є важливим інструментом UML, який використовується для візуалізації взаємодії між об'єктами в системі в певному порядку. Ця діаграма допомагає зрозуміти, як різні частини системи взаємодіють одна з одною під час виконання конкретної задачі або сценарію. Для веб-сервісу "Flexible landscape design" діаграма послідовності дій може бути розроблена для ключових процесів, таких як реєстрація користувача, додавання нового проекту та надсилання контактного повідомлення.

Діаграма послідовності дій для реєстрації користувача показує, як користувач взаємодіє з системою під час створення нового облікового запису.

Користувач: Вводить дані у форму реєстрації (ім'я, електронна адреса, пароль тощо) і натискає кнопку "Зареєструватися".

Форма реєстрації: Відправляє введені дані на сервер.

Сервер: Отримує дані, перевіряє їх на коректність (валідація), хешує пароль та зберігає дані в базу даних.

База даних: Зберігає інформацію про нового користувача.

Сервер: Відправляє підтвердження успішної реєстрації на фронтенд.

Форма реєстрації: Показує користувачу повідомлення про успішну реєстрацію.

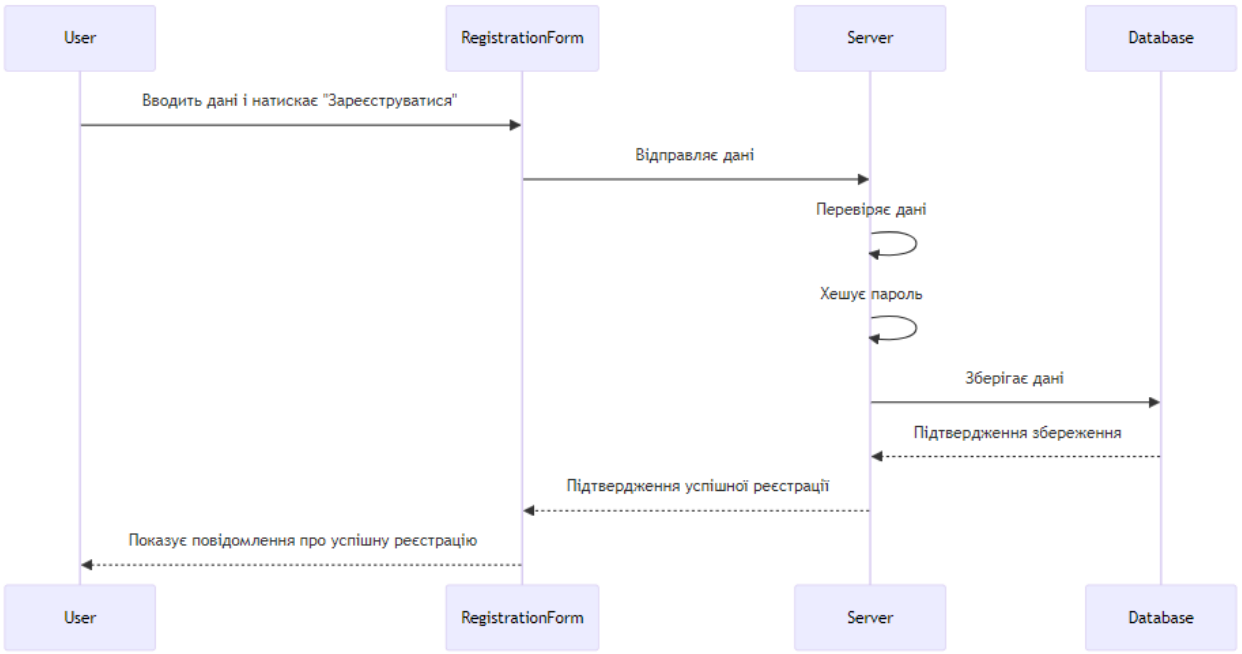

Рисунок 3.4 – Діаграма послідовності дій для реєстрації користувача (рисунок виконано самостійно)

Діаграма послідовності дій для додавання нового проекту демонструє, як адміністратор взаємодіє з системою під час створення нового проекту.

Адміністратор: Входить у систему та переходить до форми додавання проекту.

Форма додавання проекту: Адміністратор вводить дані про проект (назва, опис, завантажує зображення) і натискає кнопку "Додати проект".

Форма: Відправляє дані на сервер.

Сервер: Отримує дані, перевіряє їх на коректність, зберігає зображення у сховище та записує інформацію про проект у базу даних.

База даних: Зберігає інформацію про новий проект.

Сервер: Відправляє підтвердження успішного додавання проекту на фронтенд.

Форма додавання проекту: Показує адміністратору повідомлення про успішне додавання проекту.

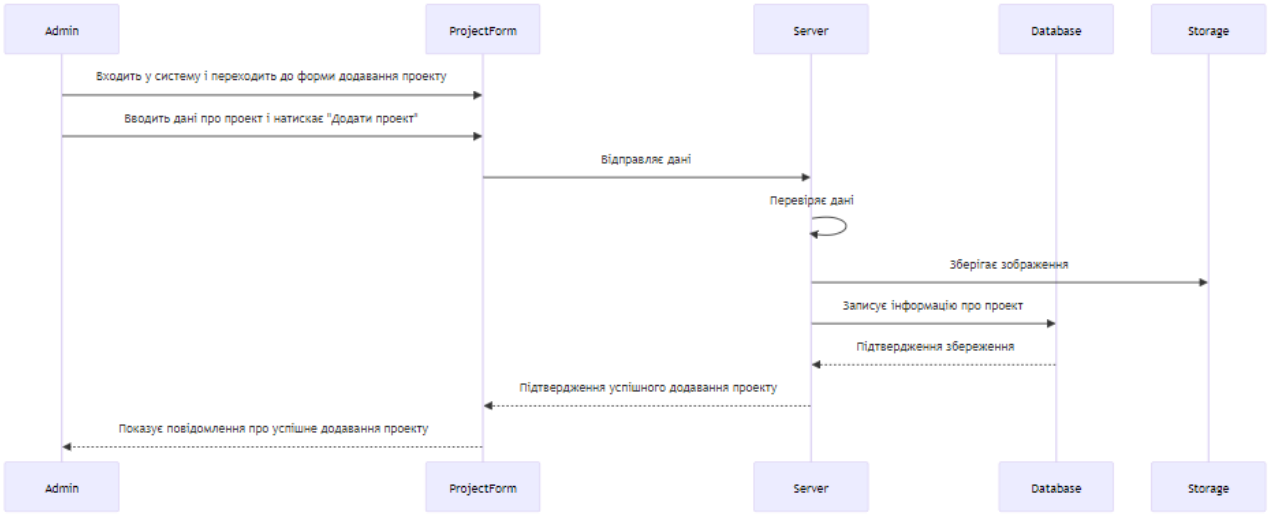

Рисунок 3.5 – Діаграма послідовності дій для додавання нового проекту (рисунок виконано самостійно)

Діаграма послідовності дій для надсилання контактного повідомлення ілюструє, як користувач взаємодіє з системою для надсилання повідомлення через контактну форму.

Користувач: Відкриває контактну форму, заповнює поля (ім'я, електронна адреса, номер телефону, повідомлення) і натискає кнопку "Надіслати".

Контактна форма: Відправляє введені дані на сервер.

Сервер: Отримує дані, перевіряє їх на коректність та зберігає повідомлення в базі даних.

База даних: Зберігає інформацію про нове повідомлення.

Сервер: Відправляє підтвердження успішного надсилання повідомлення на фронтенд.

Контактна форма: Показує користувачу повідомлення про успішне надсилання.

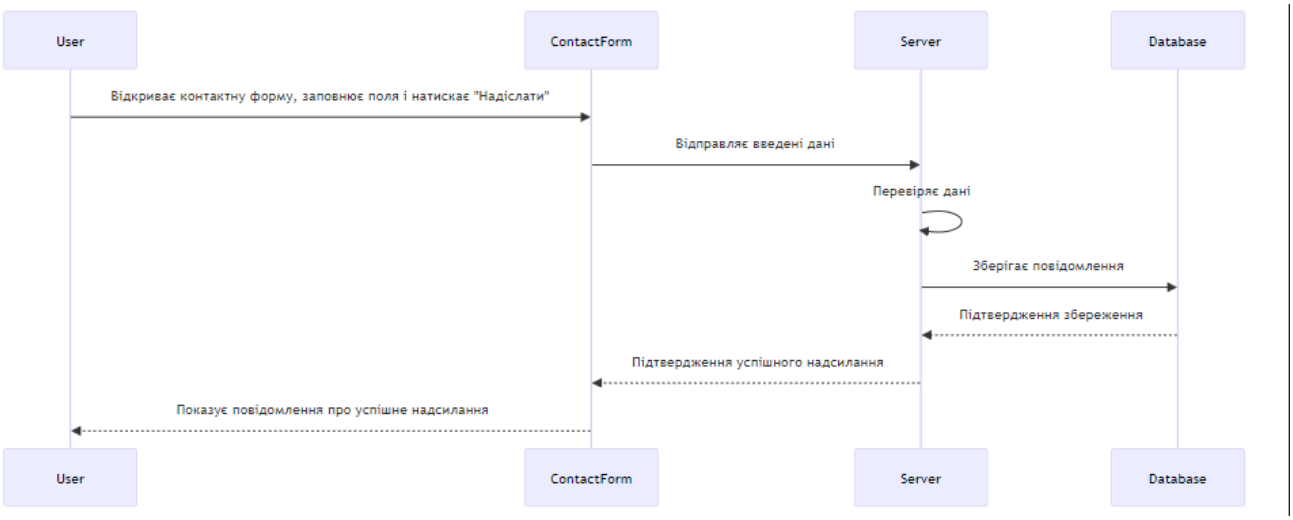

Рисунок 3.6 – Діаграма послідовності дій для надсилання контактного повідомлення (рисунок виконано самостійно)

Ці діаграми послідовності дій наочно показують, як різні компоненти веб-сервісу "Flexible landscape design" взаємодіють між собою під час виконання ключових операцій. Вони допомагають розробникам зрозуміти логіку та порядок дій, необхідних для реалізації функціональності системи, що сприяє кращій координації та ефективному розподілу завдань у процесі розробки.

3.2 Проєктування архітектури ПЗ

Розробка модулів програмного забезпечення є критичним етапом у створенні веб-сервісу "Flexible landscape design". Модульний підхід дозволяє розділити систему на окремі, незалежні компоненти, які легко підтримувати, розширювати та тестувати. У цьому розділі детально розглядається процес створення і взаємодії основних модулів системи, необхідних для реалізації MVP (Minimum Viable Product).

3.2.1 Модуль управління користувачами

Модуль управління користувачами є одним з ключових компонентів системи, відповідальним за реєстрацію, аутентифікацію та управління профілями користувачів.

Процес створення цього модуля включає кілька етапів. Спочатку

розробляється база даних для зберігання інформації про користувачів. Для зберігання інформації про користувачів створюється таблиця "Users" з наступними полями:

- id: унікальний ідентифікатор користувача (типу UUID);
- name: ім'я користувача (типу VARCHAR);
- email: електронна адреса користувача (типу VARCHAR), унікальне поле;
- password: зашифрований пароль (типу VARCHAR);
- phone_number: номер телефону (типу VARCHAR);
- role: роль користувача (типу ENUM, можливі значення: "admin", "designer", "client");
- created_at: дата та час створення облікового запису (типу TIMESTAMP);
- updated_at: дата та час останнього оновлення облікового запису (типу TIMESTAMP).

Після цього реалізується API для управління користувачами, який включає кінцеві точки для реєстрації, входу, виходу та оновлення профілю користувача.

Взаємодія між клієнтською частиною (фронтендом) та серверною частиною (бекендом) здійснюється через API. При реєстрації або вході користувач надсилає свої дані на сервер, де вони перевіряються та обробляються. У разі успішної аутентифікації користувач отримує токен доступу, який використовується для подальших запитів до захищених ресурсів.

3.2.2 Модуль управління послугами

Модуль управління послугами відповідає за створення, редагування, видалення та перегляд інформації про послуги, що надаються компанією.

Процес створення цього модуля починається з розробки структури бази даних для зберігання інформації про послуги. Кожна послуга має містити такі поля, як ідентифікатор, назва, опис та URL зображення. Далі реалізується API для управління послугами, який включає кінцеві точки для створення, оновлення, видалення та отримання списку послуг.

Фронтенд взаємодіє з цим модулем через API. Адміністратори можуть

додавати нові послуги або редагувати існуючі через відповідний інтерфейс на сайті, надсилаючи запити на сервер. Користувачі, у свою чергу, можуть переглядати список доступних послуг та отримувати детальну інформацію про кожну з них.

3.2.3 Модуль управління проектами

Модуль управління проектами є важливим компонентом веб-сервісу "Flexible landscape design", оскільки він дозволяє компанії демонструвати виконані роботи, підвищуючи таким чином довіру та залучаючи нових клієнтів. Цей модуль забезпечує функціональність для додавання, редагування, видалення та перегляду проектів у вигляді галереї зображень.

Основою модуля управління проектами є база даних, в якій зберігається інформація про кожен проект. Таблиця "Projects" повинна включати наступні поля:

- id: унікальний ідентифікатор проекту (типу UUID);
- title: назва проекту (типу VARCHAR);
- description: опис проекту (типу TEXT);
- image_url: URL зображення проекту (типу VARCHAR);
- created_at: дата та час створення проекту (типу TIMESTAMP);
- updated_at: дата та час останнього оновлення проекту (типу TIMESTAMP).

Процес додавання нового проекту включає декілька кроків. Спочатку адміністратор заповнює форму на сайті, де вводить назву, опис проекту та завантажує зображення. Після цього дані з форми відправляються на сервер через POST-запит. Сервер зберігає отриману інформацію в базу даних, генерує унікальний ідентифікатор для проекту та зберігає зображення на сервері або в хмарному сховищі. Посилання на зображення (URL) зберігається в базі даних разом з іншими даними проекту.

Редагування проектів дозволяє адміністраторам оновлювати інформацію про вже існуючі проекти. Це може включати зміну назви, опису або зображення проекту. Для цього адміністратор відкриває форму редагування, де всі поля попередньо заповнені поточними значеннями з бази даних. Після внесення змін і відправки форми PUT-запит надсилається на сервер, де оновлені дані

перевіряються на коректність та зберігаються в базі даних. Якщо було завантажено нове зображення, старе зображення може бути видалене із сервера або хмарного сховища.

Веб-сервіс підтримує систему реєстрації, яка дозволяє користувачам створювати облікові записи та отримувати доступ до додаткових функцій платформи. При реєстрації користувачі можуть вибирати одну з кількох ролей: адміністратор, дизайнер або кінцевий користувач. Кожна роль має свої привілеї та обмеження, що забезпечує контроль доступу до різних функцій системи.

Адміністратори мають повний доступ до управління контентом та користувачами. Дизайнери можуть створювати та редагувати проекти, додавати нові рослини та елементи до бібліотеки. Кінцеві користувачі можуть переглядати проекти, зберігати їх та залишати відгуки. Така структура забезпечує чітке розмежування прав та обов’язків між різними категоріями користувачів.

Видалення проектів є важливою функцією для підтримання актуальності галереї робіт. Адміністратор може видалити проект, натиснувши відповідну кнопку у інтерфейсі. Це надсилає DELETE-запит на сервер, де проект видаляється з бази даних, а пов'язане з ним зображення видаляється із сховища. Перед видаленням може бути реалізовано підтвердження дії для запобігання випадковим видаленням.

Основною функцією модуля управління проектами є надання можливості користувачам переглядати виконані роботи компанії. Проекти відображаються у вигляді галереї зображень на сторінці "Previous Work". Кожне зображення повинно мати підпис із назвою проекту та коротким описом. Користувачі можуть натискати на зображення для перегляду повнорозмірної версії та детального опису проекту. Це забезпечує користувачам зручний та візуально привабливий спосіб ознайомитися з портфоліо компанії.

Модуль управління проектами інтегрується з фронтендом через RESTful API. Для кожної операції (додавання, редагування, видалення та перегляд проектів) є відповідні кінцеві точки:

– додавання проекту: POST-запит на кінцеву точку /api/projects з даними форми;

- редагування проекту: PUT-запит на кінцеву точку /api/projects/{id} з оновленими даними;
- видалення проекту: DELETE-запит на кінцеву точку /api/projects/{id};
- отримання проектів: GET-запит на кінцеву точку /api/projects для отримання списку всіх проектів.

Забезпечення безпеки модуля управління проектами є критично важливим. Всі операції з проектами повинні бути доступні лише авторизованим користувачам з відповідними правами (наприклад, адміністраторам). Для цього всі запити до API повинні містити токен аутентифікації, який перевіряється на сервері. Крім того, сервер повинен перевіряти права користувача на виконання відповідної дії. Це запобігає несанкціонованому доступу та маніпуляціям з даними проектів.

Для відстеження активності та виявлення потенційних проблем необхідно реалізувати логування всіх операцій з проектами. Це включає запис інформації про додавання, редагування та видалення проектів, а також інформації про користувачів, які виконували ці операції. Логування допомагає виявляти помилки та аналізувати дії користувачів для подальшого покращення системи.

Таким чином, модуль управління проектами є важливим компонентом веб-сервісу "Flexible landscape design", який забезпечує зручне управління та перегляд проектів, підвищуючи довіру клієнтів до компанії. Реалізація цього модуля включає розробку структури бази даних, API для взаємодії з фронтендом, забезпечення безпеки та логування операцій.

3.2.4 Модуль зворотного зв'язку

Модуль зворотного зв'язку забезпечує можливість користувачам зв'язатися з компанією через контактну форму.

Розробка цього модуля включає створення бази даних для зберігання повідомлень від користувачів, що містить такі поля, як ідентифікатор повідомлення, ім'я, електронна адреса, номер телефону та текст повідомлення. Після цього реалізується API для обробки контактних форм, який приймає дані від користувачів, перевіряє їх на коректність та зберігає в базі даних.

Фронтенд взаємодіє з цим модулем через контактну форму, розміщену на сайті. Користувачі заповнюють форму, вводячи свої дані та повідомлення, після чого форма надсилається на сервер для обробки. Адміністратори отримують сповіщення про нові повідомлення та можуть переглядати їх через адмін-панель.

3.2.5 Модуль безпеки

Модуль безпеки відповідає за захист даних користувачів та забезпечення безпечної взаємодії з системою.

Розробка цього модуля включає реалізацію механізмів шифрування для зберігання паролів, використання токенів для аутентифікації користувачів та впровадження багатофакторної аутентифікації для додаткового захисту. Крім того, необхідно забезпечити захист від типових веб-атак, таких як SQL-ін'єкції, XSS та CSRF.

Взаємодія між модулями здійснюється через захищені API. Всі запити від користувачів проходять через модуль безпеки, де перевіряються на валідність та авторизацію перед обробкою.

3.2.6 Шаблони проектування

При розробці веб-сервісу "Flexible landscape design" було використано кілька шаблонів проектування, які допомагають забезпечити структурованість, зручність та ефективність коду. Шаблони проектування, або патерни проектування, представляють собою перевірені рішення для типових задач у розробці програмного забезпечення. У наведеному коді реалізовані декілька таких шаблонів, серед яких виділяються "Модуль", "Спостерігач", "Фабрика" та "Команда".

Шаблон "Модуль" (Module) використовується для організації коду в незалежні, ізольовані блоки, що мають власний контекст. У наведеному коді функція drawCanvas та інші пов'язані з нею функції, такі як drawBackground, представляють модуль для відображення графіки на полотні. Цей модуль інкапсулює логіку малювання та взаємодії з полотном, дозволяючи зберігати інші частини коду окремо від специфічних деталей реалізації графічного інтерфейсу.

Шаблон "Спостерігач" (Observer) дозволяє об'єктам (спостерігачам) відслідковувати стан іншого об'єкта (суб'єкта) та автоматично отримувати сповіщення про зміни його стану. У цьому коді роль спостерігачів виконують обробники подій (event listeners), які підписуються на різні події, такі як dragstart, dragover, drop, mousedown, mousemove та mouseup. Коли відбувається відповідна подія, викликається відповідний обробник, який обробляє зміну стану системи, наприклад, переміщення або додавання рослин на полотно.

Шаблон "Фабрика" (Factory) використовується для створення об'єктів без вказівки конкретних класів об'єктів, які будуть створені. У коді використовується фабричний метод для створення нових елементів рослин та будинків у списках. Наприклад, при додаванні нових зображень рослин та будинків до списків plantList та houseList, створюються нові елементи img за допомогою циклів forEach, які додають ці елементи до DOM без необхідності створювати кожен елемент вручну.

Шаблон "Команда" (Command) дозволяє інкапсулювати запити як об'єкти, що дозволяє параметризувати клієнтів з різними запитами, ставити запити в чергу або логгувати їх. У наведеному коді реалізація шаблону "Команда" можна побачити у вигляді обробників подій, які виконують конкретні дії, такі як переміщення рослин або їх малювання на полотні. Кожен обробник подій виконує певну команду, залежно від того, яка подія була викликана (наприклад, dragstart, drop, mousedown, mousemove, mouseup).

Шаблон "Одинак" (Singleton) може бути використаний для забезпечення того, щоб клас мав тільки один екземпляр, і надавав до нього глобальну точку доступу. Хоча явного використання "Одинака" в коді не представлено, можна припустити його доцільність у контексті зберігання глобального стану, такого як список рослин, що знаходяться на полотні. В такому випадку, об'єкт plants може бути реалізований як одинак, щоб гарантувати єдиний доступний список для всіх частин програми.

3.3 Проєктування структури зберігання даних

Проєктування бази даних є важливим етапом у розробці веб-сервісу "Flexible

landscape design". Правильно спроєктована база даних забезпечує ефективне зберігання, обробку та доступ до даних, що є критичним для функціонування системи. У цьому розділі детально розглядається процес проєктування бази даних для даного веб-сервісу.

Першим кроком у проєктуванні бази даних є вибір типу бази даних. Існує два основних типи баз даних: реляційні (SQL) та нереляційні (NoSQL). Для веб-сервісу "Flexible landscape design" доцільно використовувати нереляційну базу даних, таку як MongoDB, оскільки вони забезпечують зручні засоби для управління даними, підтримують складні запити та забезпечують високу цілісність даних.

Наступним кроком є визначення основних сутностей, які будуть зберігатися в базі даних, та їх атрибутів. Для веб-сервісу "Flexible landscape design" можна виділити наступні основні сутності.

Користувачі (Users):

- id: унікальний ідентифікатор користувача;
- name: ім'я користувача;
- email: електронна адреса користувача;
- password: зашифрований пароль;
- phone_number: номер телефону;
- role: роль користувача (адміністратор, дизайнер, клієнт);
- created_at: дата та час створення облікового запису;
- updated_at: дата та час останнього оновлення облікового запису.

Послуги (Services):

- id: унікальний ідентифікатор послуги;
- title: назва послуги;
- description: опис послуги;
- image_url: URL зображення послуги;
- created_at: дата та час створення послуги;
- updated_at: дата та час останнього оновлення послуги.

Проекти (Projects):

– id: унікальний ідентифікатор проекту;
– title: назва проекту;
– description: опис проекту;
– image_url: URL зображення проекту;
– created_at: дата та час створення проекту;
– updated_at: дата та час останнього оновлення проекту.

Контактні повідомлення (ContactMessages):

– id: унікальний ідентифікатор повідомлення;
– name: ім'я відправника;
– email: електронна адреса відправника;
– phone_number: номер телефону відправника;
– message: текст повідомлення;
– created_at: дата та час створення повідомлення.

Для забезпечення цілісності даних та ефективного управління інформацією важливо правильно визначити зв'язки між сутностями. У базі даних веб-сервісу "Flexible landscape design" можна виділити наступні зв'язки.

Зв'язок між користувачами та проектами:

– один користувач (дизайнер) може мати кілька проектів;
– один проект може належати лише одному користувачу;
– це зв'язок типу "один-до-багатьох".

Зв'язок між користувачами та контактними повідомленнями:

– один користувач (адміністратор) може отримувати кілька повідомлень;
– одне повідомлення може належати лише одному користувачу;
– це зв'язок типу "один-до-багатьох".

Створімо ER діаграму даної структури бази даних (див. рис. 3.7).

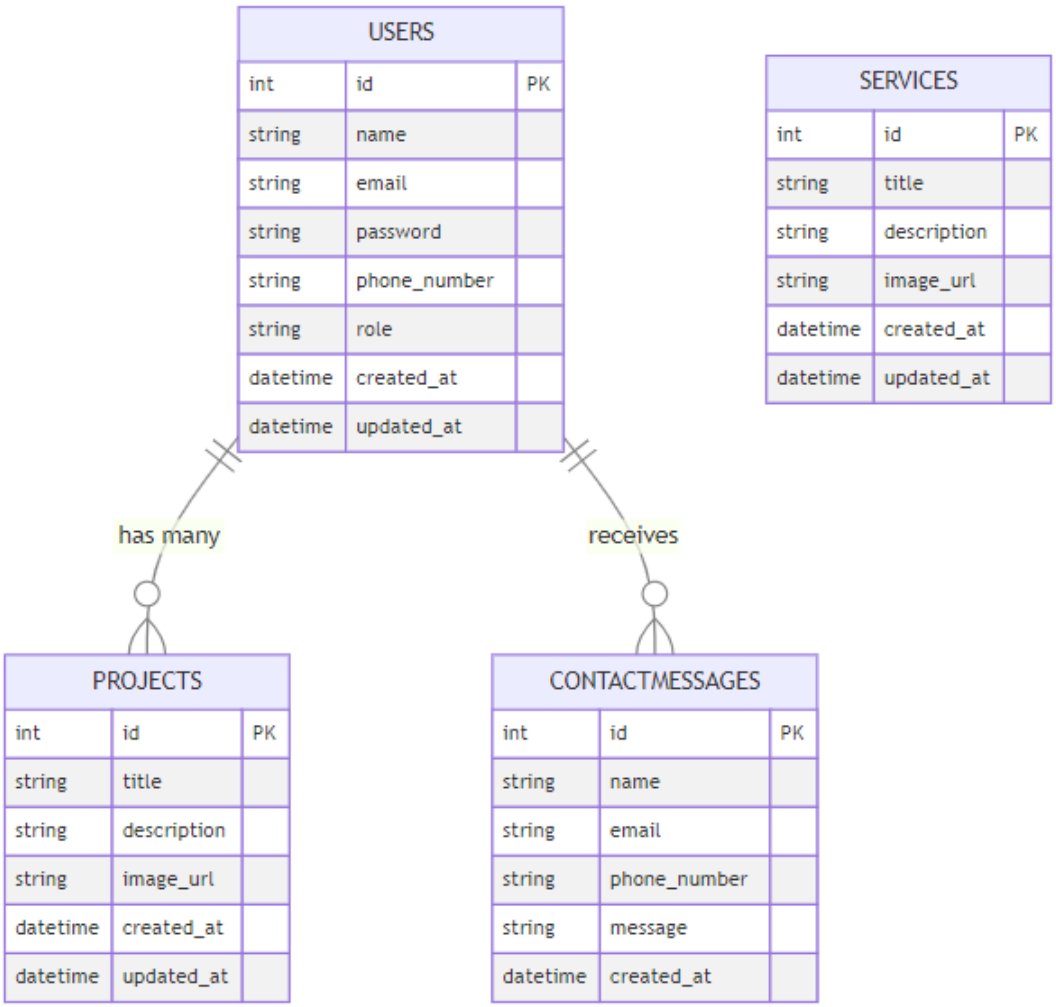

Рисунок 3.7 – ER діаграма бази даних (рисунок виконано самостійно)

Для підвищення продуктивності бази даних важливо використовувати індекси для часто використовуваних запитів. Індекси дозволяють швидше знаходити та сортувати дані. Необхідно створити індекси для полів, які часто використовуються в умовах пошуку, таких як "email" у таблиці користувачів або "title" у таблиці послуг.

Крім того, важливо оптимізувати запити для зменшення навантаження на базу даних. Це включає використання ефективних запитів, уникнення надмірних джойн-запитів та використання кешування для зменшення кількості запитів до бази даних.

Для запобігання втраті даних необхідно реалізувати механізми регулярного резервного копіювання бази даних. Це може бути автоматизована система, яка

створює резервні копії бази даних у визначені інтервали часу. Також важливо передбачити процедуру відновлення даних з резервних копій у разі виникнення збоїв або втрати даних.

Таким чином, проєктування бази даних для веб-сервісу "Flexible landscape design" включає вибір типу бази даних, визначення сутностей та їх атрибутів, визначення зв'язків між сутностями, нормалізацію даних, використання індексів, оптимізацію запитів, реалізацію механізмів резервного копіювання та відновлення, а також забезпечення безпеки даних. Правильно спроєктована база даних є основою для ефективного функціонування веб-сервісу та забезпечує надійне зберігання та обробку даних.

3.4 Приклади найцікавіших алгоритмів та методів

У цьому розділі розглянемо один із найбільш цікавих алгоритмів, реалізованих у веб-сервісі "Flexible landscape design". Ми детально проаналізуємо алгоритм, який дозволяє користувачам створювати інтерактивні візуалізації ландшафтних проектів, перетягуючи зображення рослин та будинків на полотно (canvas). Цей алгоритм ілюструє, як можна використовувати JavaScript та HTML5 для створення динамічного та інтерактивного інтерфейсу.

Алгоритм перетягування та масштабування елементів на полотні включає кілька ключових компонентів: завантаження зображень, обробку подій перетягування, розміщення елементів на полотні та їх масштабування в залежності від позиції на полотні. Розглянемо цей процес детально.

Першим кроком є ініціалізація HTML-елементів та завантаження зображень рослин і будинків, які будуть використовуватися для створення ландшафтних проектів. Зображення завантажуються в спеціальні списки та робляться доступними для перетягування.

Наступним кроком є обробка подій перетягування. Ці події включають початок перетягування зображення, його переміщення над полотном та завершення перетягування з розміщенням зображення на полотні.

Коли користувач починає перетягування зображення, створюється об'єкт

`currentDraggedPlant`, який зберігає інформацію про зображення та його початкову позицію.

Коли зображення перетягується над полотном, подія `dragover` запобігає стандартній обробці браузером, щоб дозволити подію `drop`:

При завершенні перетягування і розміщенні зображення на полотні, об'єкт `currentDraggedPlant` отримує координати для розміщення зображення в місці, де відбулося завершення перетягування.

Крім простого перетягування, алгоритм також дозволяє користувачам переміщати зображення по полотну і змінювати їх розмір залежно від вертикальної позиції.

Коли користувач натискає на зображення на полотні (mousedown), об'єкт currentDraggedPlant оновлюється з урахуванням координат миші, щоб забезпечити точне переміщення. Зміщення миші відносно зображення зберігається в dragOffsetX та dragOffsetY, щоб під час переміщення зображення зберігало свою початкову позицію відносно курсора миші.

Під час переміщення зображення (mousemove) координати зображення оновлюються в реальному часі, враховуючи зміщення курсора миші. Розмір зображення змінюється пропорційно вертикальній позиції на полотні, що додає ефект перспективи – зображення ближче до нижньої частини полотна виглядають більшими.

3.5 Створення UI / UX

Проєктування інтерфейсу користувача (UI) є одним з найважливіших етапів розробки веб-сервісу "Flexible landscape design". Основною метою цього етапу є створення інтуїтивно зрозумілого, естетично привабливого та функціонального інтерфейсу, який забезпечить зручність користувачів при взаємодії з системою. У цьому розділі розглянемо деталі проєктування інтерфейсу, базуючись на наданому макеті сайту.

Головна сторінка веб-сервісу "Flexible landscape design" (див. рис. 3.8) має включати чітку навігаційну структуру, що допоможе користувачам легко

орієнтуватися на сайті. У верхній частині сторінки розташовується навігаційне меню, яке містить посилання на основні розділи сайту: "Home", "Services", "Work" та "Contact". Логотип компанії також розташований у верхньому лівому куті і виконує функцію повернення на головну сторінку при натисканні.

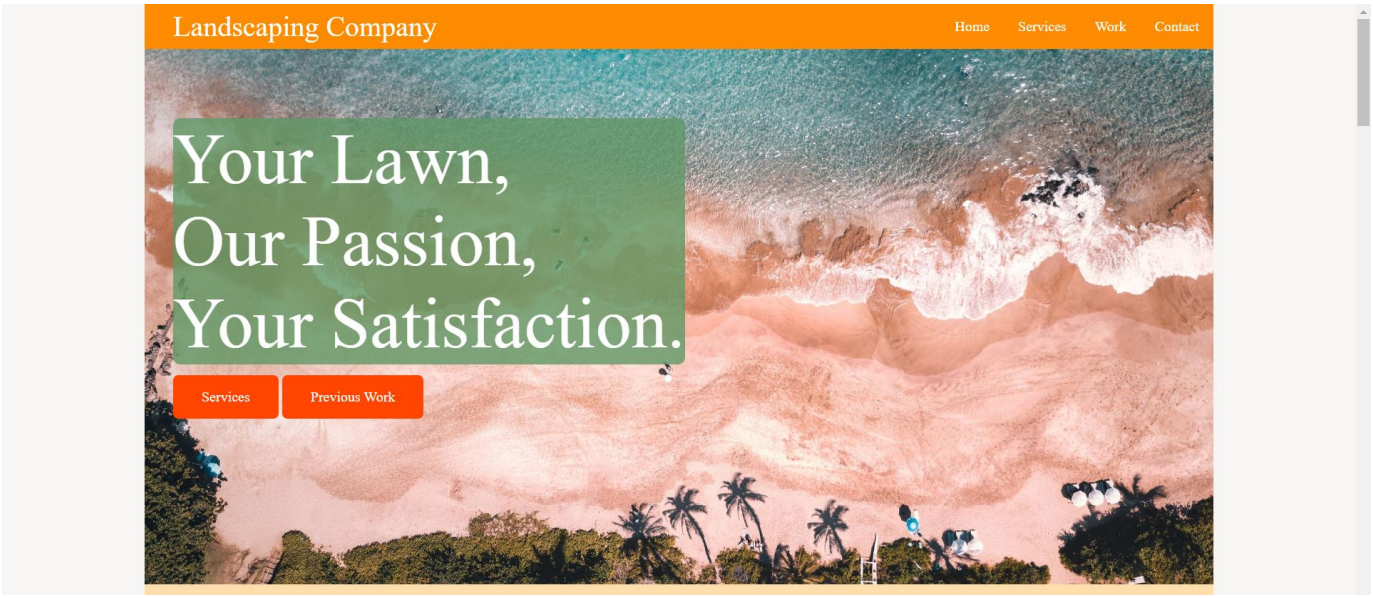

Рисунок 3.8 – Головна сторінка (рисунок виконано самостійно)

Навігаційне меню повинно бути фіксованим при прокручуванні сторінки, щоб користувачі мали постійний доступ до всіх розділів сайту. Це забезпечить зручність навігації, особливо на довгих сторінках з великою кількістю контенту.

Після навігаційного меню на головній сторінці розташована секція з девізом компанії "Your Lawn, Our Passion, Your Satisfaction". Ця секція повинна бути візуально привабливою, використовувати великий шрифт для девізу та фон у вигляді високоякісного зображення, що підкреслює сферу діяльності компанії.

Під девізом розташовані дві кнопки: "Services" та "Previous Work". Ці кнопки повинні бути чітко видимими та мати достатній розмір, щоб користувачі могли легко їх знайти та натиснути. Кнопки повинні використовувати контрастні кольори для покращення видимості.

Розділ "Services" (див. рис. 3.9) представляє послуги, що надаються компанією. Кожна послуга оформлена у вигляді картки, яка включає зображення, заголовок та текстовий опис. Зображення повинні бути високої якості та релевантні

до описуваних послуг, що допоможе користувачам краще зрозуміти суть кожної послуги.

Рисунок 3.9 – Розділ "Services" (рисунок виконано самостійно)

Заголовок послуги повинен бути виділений великим шрифтом, а текстовий опис – середнього розміру, зручно читабельним. Важливо використовувати однаковий стиль для всіх карток, щоб забезпечити консистентність інтерфейсу. Картки послуг повинні бути розташовані у вигляді сітки, що забезпечить зручний перегляд та доступ до кожної послуги.

Розділ "Previous Work" (див. рис. 3.10) демонструє приклади виконаних робіт компанії у вигляді галереї зображень.

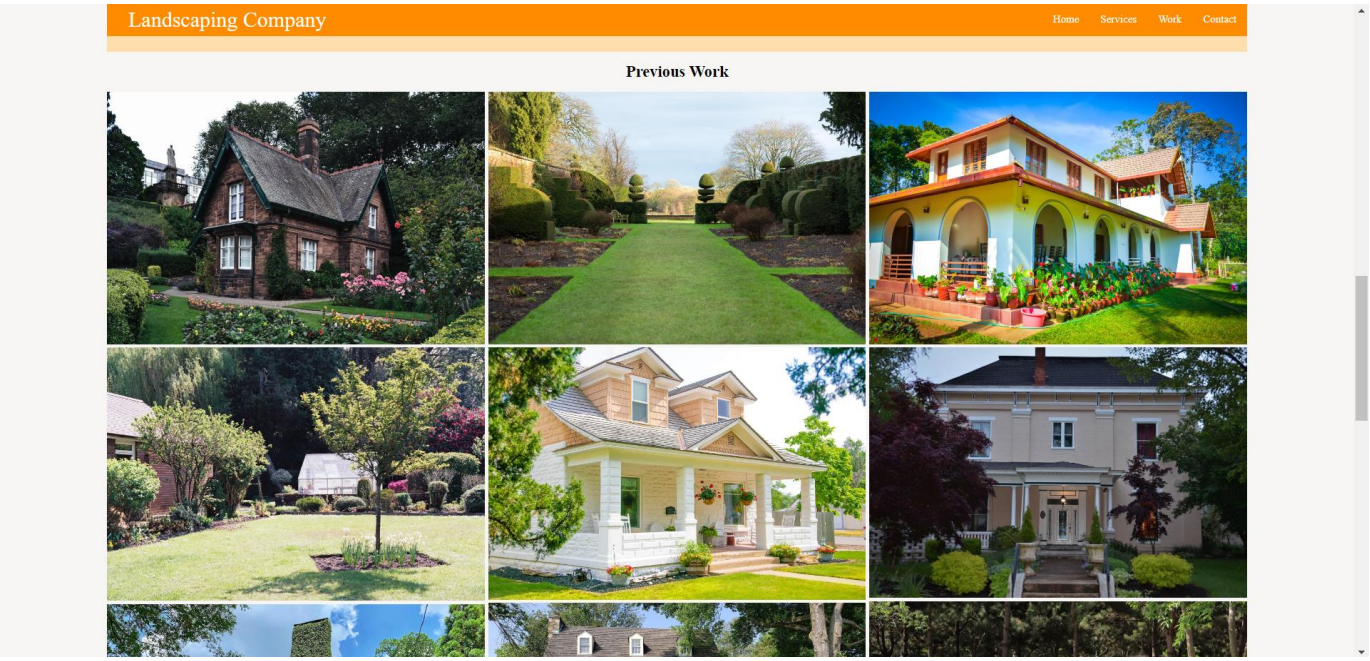

Рисунок 3.10 – Галерея (рисунок виконано самостійно)

Галерея повинна бути зручною для перегляду, з можливістю збільшення зображень для детального огляду. Кожне зображення повинно мати підпис, що описує проект, який воно представляє. Це допоможе користувачам краще оцінити якість виконаних робіт.

Розділ "Contact" (див. рис. 3.11) містить контактну інформацію компанії та контактну форму для зворотного зв’язку.

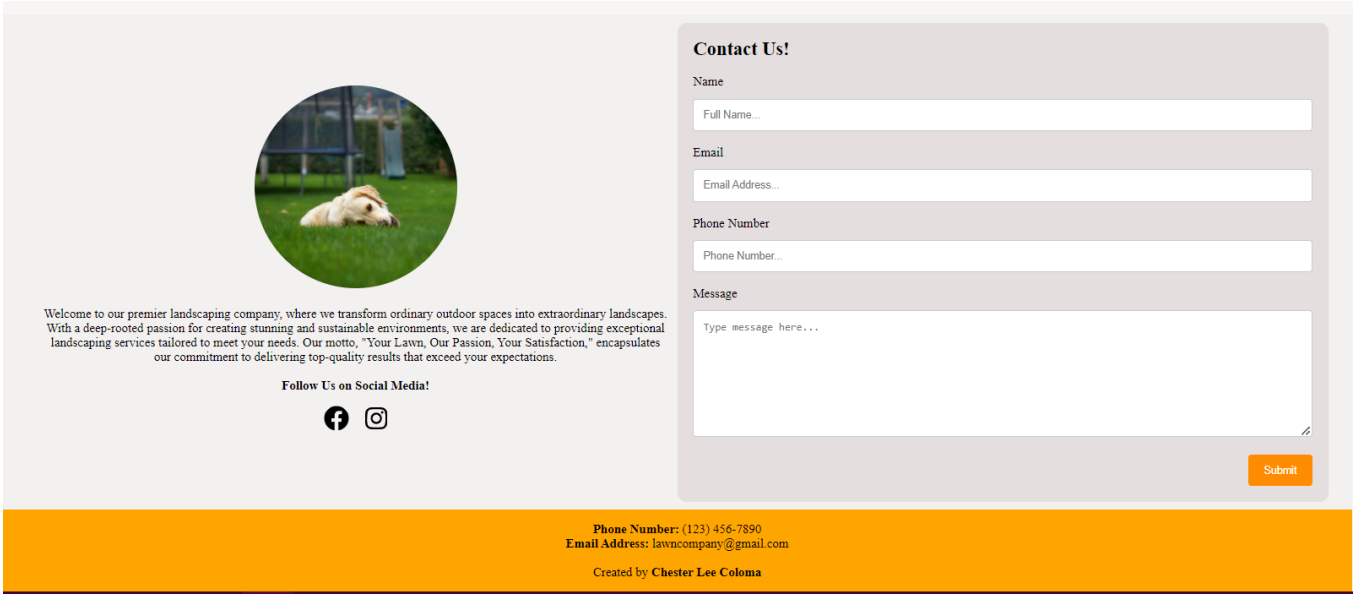

Рисунок 3.11 – Розділ "Contact"(рисунок виконано самостійно)

Контактна інформація повинна включати номер телефону, електронну адресу та посилання на соціальні мережі. Ця інформація повинна бути розташована у верхній частині розділу для швидкого доступу.

Контактна форма повинна бути інтуїтивно зрозумілою та включати поля для введення імені, електронної адреси, номера телефону та повідомлення. Всі поля повинні бути обов’язковими для заповнення, а кнопка відправки повинна бути чітко видимою та легко натискатися. Важливо забезпечити валідацію введених даних для запобігання помилкам та некоректній інформації.

Розділ "Custom design" (див. рис. 3.12) містить поле для проєктування власного ланшафтного дизайну. Ця форма надає можливість розташовувати рослини та певні елементи декору на полотні таким чином, щоб це відповідало інтересам споживача.

Рисунок 3.12 – Розділ "Contact"(рисунок виконано самостійно)

Користувачі можуть вибирати об'єкти з панелі інструментів і перетягувати їх на канвас. Це забезпечує інтерактивність і дозволяє користувачам легко

розміщувати елементи на проекті. Об'єкти, розміщені на канвасі, можуть бути переміщені, змінені за розміром або видалені. Це дає користувачам можливість точно налаштувати розташування кожного елемента відповідно до їхнього бачення.

Центральним елементом інтерфейсу є великий інтерактивний канвас, де користувачі можуть візуально розміщувати об'єкти. Канвас повинен мати високу роздільну здатність, щоб забезпечити чіткість зображень та детальність проекту. З лівого боку або зверху канвасу розташована панель інструментів, яка містить список доступних для додавання об'єктів. Об'єкти поділені на категорії, такі як "Plants" і "Houses", що полегшує їх пошук та використання.

Однією з ключових функцій веб-сервісу "Flexible landscape design" є можливість додавання рослин та пейзажів до проектів. Користувачі можуть вибирати з великої бібліотеки рослин та елементів ландшафту, які класифікуються за типами, такими як дерева, кущі, квіти, водойми та архітектурні елементи. Ця функція дозволяє користувачам створювати реалістичні та деталізовані ландшафтні проекти, враховуючи всі бажані елементи.

Користувачі можуть додавати рослини до свого проекту, перетягуючи їх з панелі інструментів на канвас. Рослини можна розміщувати, змінювати їх розмір, обертати та видаляти. Це забезпечує максимальну гнучкість у процесі створення дизайну та дозволяє користувачам експериментувати з різними комбінаціями елементів.

Для забезпечення ефективної роботи системи та уникнення перевантаження, встановлюються певні обмеження на кількість рослин, які можуть бути додані до одного проекту. Це обмеження залежить від загальної площі проекту та розміру окремих елементів. Такий підхід дозволяє зберегти високу продуктивність системи та забезпечити плавну роботу навіть при великій кількості елементів у проекті. Окрім того, розмір файлу проекту обмежено 50мб, хоча він не впливає на кількість рослин, так як інформація про їх розташування зберігається однією картинкою та текстом.

Для зареєстрованих користувачів повинен бути доступний розділ профілю, де вони можуть переглядати та редагувати свою особисту інформацію, зберігати

обрані послуги та проекти, а також залишати відгуки про послуги компанії. Інтерфейс профілю повинен бути простим та інтуїтивно зрозумілим, забезпечуючи легкий доступ до всіх необхідних функцій.

Коли користувач додає рослини або створює дизайн на канвасі, ці зміни автоматично зберігаються в проекті. Інші користувачі не можуть переглядати ці проекти, вони зберігаються тільки для даного користувача.

Інтерфейс веб-сервісу "Flexible landscape design" повинен бути адаптивним, забезпечуючи коректне відображення на різних пристроях. Всі елементи інтерфейсу повинні автоматично підлаштовуватися під розмір екрану, забезпечуючи зручність використання на мобільних телефонах, планшетах та настільних комп’ютерах.

Важливо забезпечити доступність для користувачів з обмеженими можливостями. Це включає дотримання стандартів доступності, таких як використання семантичного HTML для структурування контенту, забезпечення достатнього контрасту між текстом і фоном, а також надання альтернативних текстів для зображень. Крім того, всі інтерактивні елементи, такі як кнопки та форми, повинні підтримувати навігацію з клавіатури.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

4.1 Обрані технології

Для розробки веб-сервісу "Flexible landscape design" були обрані сучасні та ефективні технології, які забезпечують високу продуктивність, зручність використання та масштабованість системи.

Фронтенд частина проєкту була реалізована з використанням HTML, CSS та JavaScript. HTML забезпечує структурування вмісту веб-сторінок, що дозволяє розділити контент на логічні блоки та забезпечити правильне семантичне оформлення. CSS відповідає за стиль і оформлення веб-сторінок, включаючи колірну гаму, шрифти та розташування елементів. Використання CSS дозволяє створити адаптивний дизайн, що коректно відображається на різних пристроях, таких як настільні комп’ютери, планшети та мобільні телефони. JavaScript додає інтерактивність і динамічну взаємодію з користувачами, забезпечуючи функціональність таких елементів, як навігаційне меню, форми зворотного зв'язку та галереї зображень. Вибір цих технологій обумовлений їхньою популярністю, широким використанням у веб-розробці та підтримкою всіх сучасних браузерів, що гарантує високу швидкість завантаження сторінок і зручність для користувачів

Для бекенд-розробки був обраний Node.js у поєднанні з Express.js. Node.js забезпечує виконання серверного коду на основі подієвої моделі, що дозволяє обробляти велику кількість запитів з мінімальними затримками. Express.js, як фреймворк для Node.js, спрощує розробку серверних додатків, забезпечуючи необхідні інструменти для створення RESTful API, управління маршрутами та обробки запитів. Вибір цих технологій забезпечує високу продуктивність серверної частини, гнучкість у розробці та підтримку великої кількості одночасних користувачів

Для зберігання даних була обрана база даних MongoDB, яка є нереляційною (NoSQL) базою даних. MongoDB була обрана через її здатність ефективно управляти великими обсягами даних, підтримку гнучкої схеми даних та високу масштабованість. Однією з ключових переваг MongoDB є можливість швидкого та простого виконання складних запитів до бази даних, що забезпечує високу

продуктивність системи. Крім того, MongoDB підтримує горизонтальне масштабування, що дозволяє легко збільшувати потужності системи в міру зростання кількості користувачів і обсягів даних .

Для тестування системи були обрані інструменти Mocha та Chai. Mocha є фреймворком для тестування JavaScript, який дозволяє писати асинхронні тести для Node.js-додатків, забезпечуючи перевірку правильності роботи серверних модулів і API. Chai використовується для написання тверджень (assertions) в тестах, що спрощує процес перевірки результатів тестування. Вибір цих інструментів обумовлений їхньою ефективністю, простотою використання та підтримкою широкого спектру тестових сценаріїв, що дозволяє забезпечити високу якість і надійність програмного забезпечення

Таким чином, обрані технології для фронтенду і бекенду, база даних та інструменти для тестування забезпечують надійну, продуктивну та зручну у використанні систему, яка відповідає всім вимогам сучасного веб-розроблення та потребам користувачів.

4.2 Архітектура системи

Архітектура системи "Flexible landscape design" побудована на принципах модульності та гнучкості, що забезпечує легкість підтримки, масштабування та розширення функціоналу. Загальна структура системи складається з клієнтської частини (фронтенду) та серверної частини (бекенду), які взаємодіють між собою через API.

Фронтенд відповідає за відображення інформації користувачам і реалізований з використанням технологій HTML, CSS і JavaScript. Основними компонентами фронтенду є навігаційне меню, секції для презентації послуг, галерея проектів та контактна форма. Всі ці компоненти взаємодіють з бекендом для отримання та відправлення даних.

Бекенд частина системи реалізована на базі Node.js та Express.js. Вона відповідає за обробку запитів від клієнтської частини, управління даними та забезпечення безпеки системи. Основні компоненти бекенду включають модулі

управління користувачами, послугами, проектами, зворотнім зв'язком та безпекою. Кожен з цих модулів реалізований як окремий компонент, що дозволяє легко вносити зміни та додавати нові функції.

Модуль управління користувачами відповідає за реєстрацію, аутентифікацію та управління профілями користувачів. Він взаємодіє з базою даних для зберігання інформації про користувачів та забезпечує безпеку облікових записів через шифрування паролів та використання токенів для аутентифікації. Приклад коду для реєстрації користувача виглядає наступним чином:

```
app.post('/register', async (req, res) => {
    const { name, email, password } = req.body;
    const hashedPassword = await bcrypt.hash(password, 10);
    const newUser = new User({ name, email, password: hashedPassword
});
    await newUser.save();
    res.status(201).send('User registered');
});
```

Модуль управління послугами забезпечує створення, редагування, видалення та перегляд інформації про послуги, що надаються компанією. Цей модуль включає кінцеві точки API, які дозволяють адміністраторам керувати послугами, а користувачам переглядати їх.

Модуль управління проектами дозволяє компанії демонструвати виконані роботи через галерею проектів. Кожен проект містить зображення та опис, що зберігаються в базі даних. Цей модуль також включає можливість додавання, редагування та видалення проектів через відповідний API.

Модуль зворотного зв'язку забезпечує можливість користувачам зв'язуватися з компанією через контактну форму. Повідомлення зберігаються в базі даних та можуть бути переглянуті адміністраторами через адмін-панель. Приклад коду для обробки контактної форми:

```
app.post('/contact', async (req, res) => {
    const { name, email, message } = req.body;
    const newMessage = new ContactMessage({ name, email, message });
    await newMessage.save();
    res.status(200).send('Message received');});
```

Модуль безпеки відповідає за захист даних користувачів та забезпечення безпечної взаємодії з системою. Він включає механізми шифрування паролів, використання токенів для аутентифікації та захист від типових веб-атак, таких як

SQL-ін'єкції та XSS. Всі взаємодії між модулями здійснюються через захищені API.

Для забезпечення масштабованості системи використовуються сучасні підходи, такі як мікросервісна архітектура. Це дозволяє розподіляти навантаження між різними сервісами та легко додавати нові модулі без суттєвого впливу на продуктивність системи. Модульний підхід також сприяє масштабованості, дозволяючи розробникам зосереджуватися на окремих компонентах без ризику порушення роботи всієї системи.

4.3 Взаємодія компонентів

Взаємодія між фронтендом і бекендом у веб-сервісі "Flexible landscape design" здійснюється через RESTful API. Ця архітектура дозволяє легко масштабувати і підтримувати систему, забезпечуючи ефективну обробку запитів і відповідей між різними компонентами.

Фронтенд, реалізований з використанням HTML, CSS і JavaScript, відповідає за відображення інформації користувачам і забезпечує інтерактивний інтерфейс. Він включає такі основні компоненти, як навігаційне меню, секції для презентації послуг, галерею проектів і контактну форму. Для взаємодії з бекендом використовуються AJAX-запити, які дозволяють асинхронно отримувати і відправляти дані, забезпечуючи динамічну взаємодію без необхідності перезавантаження сторінки.

Бекенд, реалізований на базі Node.js і Express.js, відповідає за обробку запитів від клієнтської частини, управління даними і забезпечення безпеки системи. Основні компоненти бекенду включають модулі управління користувачами, послугами, проектами, зворотним зв'язком і безпекою. Кожен з цих модулів реалізований як окремий компонент, що дозволяє легко вносити зміни і додавати нові функції.

Для комунікації між фронтендом і бекендом використовуються стандартні HTTP-протоколи. Запити можуть бути різних типів, таких як GET для отримання даних, POST для створення нових записів, PUT для оновлення існуючих записів і DELETE для видалення. Ці запити здійснюються через RESTful API, що забезпечує

структурований і зрозумілий спосіб взаємодії між компонентами.

Запити від фронтенду обробляються на сервері за допомогою Express.js, який маршрутизує їх до відповідних обробників. Наприклад, при реєстрації або вході користувача, дані надсилаються на сервер, де вони перевіряються і обробляються. У разі успішної аутентифікації користувач отримує токен доступу, який використовується для подальших запитів до захищених ресурсів:

```
app.post('/login', async (req, res) => {
  const { username, password } = req.body;
  const user = await User.findOne({ username, password });
  if (user) {
    req.session.user = user;
    res.json({ success: true });
  } else {
    res.status(401).json({ success: false, message: 'Invalid credentials' });
  }});
```

Цей фрагмент коду показує, як сервер обробляє запит на вхід користувача, перевіряючи його облікові дані і відповідаючи з результатом.

Управління станом додатку здійснюється як на клієнтській, так і на серверній частинах. На фронтенді використовується JavaScript для зберігання і оновлення стану інтерфейсу користувача. Наприклад, при завантаженні списку проектів або послуг, отримані дані зберігаються в змінних або станах компонентів, що дозволяє швидко оновлювати інтерфейс без необхідності повторних запитів до сервера.

На бекенді стан додатку управляється за допомогою сесій і бази даних. Сесії використовуються для зберігання інформації про поточного користувача, а база даних MongoDB забезпечує зберігання і доступ до даних про користувачів, послуги, проекти та повідомлення.

Різні ендпоїнти API забезпечують доступ до різних ресурсів і функцій системи. Наприклад, для управління користувачами є ендпоїнти для реєстрації, входу, виходу і оновлення профілю користувача. Для управління проектами і послугами є ендпоїнти для створення, редагування, видалення і отримання списку проектів або послуг.

Фронтенд отримує дані про рослини та декор через ендпоїнти `/plants` та `/decors`. Ці дані використовуються для відображення елементів, які користувач може використовувати при створенні дизайнів:

```
app.get('/plants', async (req, res) => {
  const plants = await DecorObject.find({ type: 'plant' });
  res.json(plants);
});
app.get('/decors', async (req, res) => {
  const decors = await DecorObject.find({ type: 'decor' });
  res.json(decors);
});
```

Ендпоїнт `/backgrounds` повертає список доступних фонів для використання в користувацьких дизайнах:

```
app.get('/backgrounds', async (req, res) => {
  const backgrounds = await Background.find();
  res.json(backgrounds);
});
```

Для входу та виходу користувачів використовуються ендпоїнти `/login` та `/logout`. Вони обробляють дані користувача та керують сесіями:

```
app.post('/login', async (req, res) => {
  const { username, password } = req.body;
  const user = await User.findOne({ username, password });
  if (user) {
    req.session.user = user;
    res.json({ success: true });
  } else {
    res.status(401).json({ success: false, message: 'Invalid
credentials' });
  }
});
app.post('/logout', (req, res) => {
  req.session.destroy();
  res.json({ success: true });
});
```

Користувачі можуть отримувати список своїх проектів, створювати нові проекти, а також зберігати свої дизайни за допомогою наступних ендпоїнтів.

Проекти користувача знаходяться у базі даних за його ID. Цей ендпоїнт повертає список проектів поточного користувача:

```
app.get('/projects', checkAuthenticated, async (req, res) => {
  const projects = await Project.find({ userId: req.session.user._id
});
  res.json(projects);});
```

Для створення нового проекту використовується ендпоїнт POST `/projects`, який приймає дані проекту від клієнта і зберігає їх у базі даних:

```
app.post('/projects', checkAuthenticated, async (req, res) => {
const project = new Project({ ...req.body, userId:
req.session.user._id });
  await project.save();
  res.json(project);
});
```

Адміністратор може управляти користувачами за допомогою ендпоїнтів для

отримання списку користувачів, створення нових та видалення існуючих:

```
app.get('/admin/users', checkAdmin, async (req, res) => {
  const users = await User.find();
  res.json(users);});
app.post('/admin/users', checkAdmin, async (req, res) => {
  const user = new User(req.body);
  await user.save();
  res.json(user);});
app.delete('/admin/users/:id', checkAdmin, async (req, res) => {
  await User.findByIdAndDelete(req.params.id);
  res.send('User deleted');});
```

Користувачі та адміністратори можуть завантажувати свої роботи через ендпоїнт `/admin/add-work`. Завантажені зображення зберігаються на сервері і можуть бути використані для подальшого перегляду:

```
app.post('/admin/add-work', checkAdmin, upload.single('work-image'),
async (req, res) => {
  res.json({ success: true, url:
`/assets/gallery/${req.file.filename}` });
});
```

Для отримання списку завантажених робіт використовується GET-запит до ендпоїнту `/work`:

```
app.get('/work', async (req, res) => {
  const workItems = [];
  fs.readdirSync('public/assets/gallery').forEach(file => {
    workItems.push({ url: `/assets/gallery/${file}`, name: file });
});
  res.json(workItems);});
```

Користувачі можуть зберігати свої кастомні дизайни через POST-запит до ендпоїнту `/save-design`. Цей ендпоїнт приймає дані дизайну та зображення, зберігає їх у базі даних та на сервері:

```
app.post('/save-design', (req, res) => {
  if (!req.session.user) {
return res.status(403).send('Forbidden'); }
  const { name, design, image } = req.body;
  const imageData = image.replace(/^data:image\/png;base64,/, '');
  const fileName = `${Date.now()}.png`;
  const filePath = path.join(__dirname, 'public', 'assets',
'customs', fileName);
  fs.writeFile(filePath, imageData, 'base64', async (err) => {
    if (err) {
      return res.status(500).send('Error saving design'); }
    try {
      const project = new Project({
        userId: req.session.user._id,name,design, url:
`/assets/customs/${fileName}`, });
      await project.save();
      res.json({ success: true, url: `/assets/customs/${fileName}` });
    } catch (saveError) {
      res.status(500).send('Error saving design'); } });});
```

Користувачі можуть надсилати свої повідомлення через форму зворотного зв'язку, яка обробляється ендпоїнтом POST `/contact`. Адміністратор може переглядати ці повідомлення через GET-запит до ендпоїнту `/messages`:

```
app.post('/contact', async (req, res) => {
  const message = new Message(req.body);
  await message.save();
  res.json({ success: true });});
app.get('/messages', async (req, res) => {
  const messages = await Message.find();
  res.json(messages);});
```

Фронтенд може отримувати інформацію про поточну роль користувача через ендпоїнт `/current-user-role`, який повертає роль користувача на основі його сесії:

```
app.get('/current-user-role', (req, res) => {
  if (req.session.user) {
    res.json({ role: req.session.user.role }); } else {
    res.json({ role: 'guest' }); }});
```

Таким чином, різні ендпоїнти забезпечують усі необхідні функції для взаємодії між фронтендом і бекендом, управління даними та користувачами, а також підтримки динамічних функцій веб-сервісу.

4.4 Управління даними

Управління даними у веб-сервісі "Flexible landscape design" здійснюється за допомогою різних методів і технологій, що забезпечують надійне зберігання та ефективний доступ до інформації. Основними компонентами для управління даними є база даних MongoDB, ORM-бібліотека Mongoose, а також спеціальні маршрути (ендпоїнти) для обробки запитів від клієнтської частини.

Основним інструментом для зберігання даних є MongoDB, документно-орієнтована база даних, яка дозволяє зберігати дані у вигляді документів JSON. Використання MongoDB обґрунтоване її гнучкістю, масштабованістю та простотою інтеграції з Node.js, на якому побудований бекенд нашого веб-сервісу.

Для підключення до MongoDB використовується бібліотека Mongoose, яка забезпечує зручний інтерфейс для взаємодії з базою даних, а також підтримує схеми і моделі, що дозволяє структурувати дані. Підключення до бази даних здійснюється на початку запуску сервера:

```
mongoose.connect('mongodb://localhost:27017/landscaping', {
  useNewUrlParser: true,
```

Продовження коду:

```
  useUnifiedTopology: true
}).then(() => {
  createDefaultAdmin();
  populateDatabase();
}).catch(err => {
  console.error('Database connection error:', err);});
```

Для структурування даних використовуються схеми Mongoose, які визначають структуру документів у колекціях. Наприклад, схема користувача визначає такі поля, як `username`, `password` та `role`:

```
const userSchema = new Schema({
  username: { type: String, required: true, unique: true },
  password: { type: String, required: true },
  role: { type: String, default: 'user' }});
```

Аналогічно визначаються схеми для проектів, повідомлень, об'єктів декору та фонів:

```
const projectSchema = new Schema({
  userId: { type: Schema.Types.ObjectId, ref: 'User', required: true
},
  name: { type: String, required: true },
  url: { type: String, required: true },
  design: { type: String, required: true }});
const messageSchema = new Schema({
  name: { type: String, required: true },
  email: { type: String, required: true },
  phoneNumber: { type: String, required: true },
  message: { type: String, required: true }});
const backgroundSchema = new Schema({
  name: { type: String, required: true },
  description: { type: String, required: true },
  imageUrl: { type: String, required: true },
  horizon: { type: Number, required: true, default: 50 },
  width: { type: Number, required: true },
  height: { type: Number, required: true }});
const decorObjectSchema = new Schema({
  name: { type: String, required: true },
 description: { type: String, required: true },
  minSize: { type: Number, required: true },
  maxSize: { type: Number, required: true },
  imageUrl: { type: String, required: true },
  type: { type: String, required: true }
});
```

Для збереження нових даних у базі використовується метод `save()`, який створює новий документ у відповідній колекції. Наприклад, при створенні нового користувача, проекту або об'єкту декору:

```
const user = new User(req.body);
await user.save();
const project = new Project({ ...req.body, userId: req.session.user._id
});
await project.save();
```

Продовження коду:

```
    const decor = new DecorObject({ name, description, minSize, maxSize,
imageUrl, type: 'decor' });
    await decor.save();
```

Для отримання даних з бази використовуються методи `find()` та `findOne()`. Наприклад, для отримання списку всіх користувачів, проектів або об'єктів декору:

```
    const users = await User.find();
    const projects = await Project.find({ userId: req.session.user._id });
    const plants = await DecorObject.find({ type: 'plant' });
```

Для завантаження файлів використовується бібліотека `multer`, яка дозволяє обробляти файли, завантажені через форми. Файли зберігаються на сервері, а шляхи до них зберігаються у базі даних. Наприклад, при завантаженні зображень для об'єктів декору або фонів:

```
    const storage = multer.diskStorage({
      destination: (req, file, cb) => {
        cb(null, 'public/assets/decor');  },
      filename: (req, file, cb) => {
        cb(null, Date.now() + path.extname(file.originalname));  },
    });
    const upload = multer({ storage });
    app.post('/admin/add-decor', checkAdmin, upload.single('image'), async
(req, res) => {
      const { name, description, minSize, maxSize } = req.body;
      const imageUrl = `/assets/decor/${req.file.filename}`;
      const decor = new DecorObject({ name, description, minSize, maxSize,
imageUrl, type: 'decor' });
      await decor.save();
      res.json({ success: true });});
```

Для поповнення бази даних початковими даними використовується функція `populateDatabase()`, яка читає файли з директорій і додає їх у відповідні колекції бази даних. Це забезпечує наявність початкових даних для користувачів, об'єктів декору та фонів:

```
    async function populateDatabase() {
      const treeFiles = fs.readdirSync(plantsDir);
      for (const file of treeFiles) {
        const existingPlant = await DecorObject.findOne({ imageUrl:
`/assets/plants/${file}` });
        if (!existingPlant) {
          const sizes = parseSizes(file);
          const newPlant = new DecorObject({
            name: path.parse(file).name,
            description: `Description of ${path.parse(file).name}`,
            minSize: sizes.minSize,
            maxSize: sizes.maxSize,
            imageUrl: `/assets/plants/${file}`,
            type: 'plant'
          });
          await newPlant.save();}}
```

Продовження цього коду ініціалізації бази даних початковими даними наведено у додатку Б.

Таким чином, управління даними у веб-сервісі "Flexible landscape design" здійснюється через комбінацію MongoDB для зберігання документів, Mongoose для структуризації та доступу до даних, а також спеціальних маршрутів для обробки запитів від клієнтів і завантаження файлів. Це забезпечує надійне і гнучке зберігання інформації, а також ефективну обробку запитів користувачів.

4.5 Реалізація логіки серверної частини

Інтерактивність веб-додатку "Flexible landscape design" забезпечується за допомогою різних алгоритмів та функцій, реалізованих на серверній частині. Ці функції включають обробку даних користувачів, керування проектами, завантаження файлів, а також підтримку інтерактивних компонентів, таких як канвас для створення дизайнів. В цьому розділі ми розглянемо кілька основних функцій, які забезпечують ці можливості, детально пояснюючи їх реалізацію та логіку.

Однією з ключових функцій є `populateDatabase()`, яка відповідає за ініціалізацію бази даних. Ця функція читає файли з локальних директорій і додає їх до бази даних, забезпечуючи наявність початкових даних для роботи додатку. Вона є важливою для встановлення і початкового налаштування системи, особливо під час розгортання або оновлення:

```
async function populateDatabase() {
  // Function to parse minSize and maxSize from filename
  const parseSizes = (filename) => {
    const nameParts = path.parse(filename).name.split('_');
    if (nameParts.length >= 3 && !isNaN(nameParts[1]) &&
!isNaN(nameParts[2])) {
      return {
        minSize: parseInt(nameParts[1], 10),
        maxSize: parseInt(nameParts[2], 10)
      };
    }
    // Default sizes if parsing fails
    return { minSize: 30, maxSize: 300 };
  };
```

Продовження цього коду ініціалізації бази даних за допомогою populateDatabase() наведено у додатку Б.

Ця функція читає файли з трьох основних директорій: для рослин, декору та фонів. Вона аналізує назви файлів для визначення мінімальних та максимальних розмірів елементів, а потім додає їх у відповідні колекції бази даних. Це забезпечує базовий набір даних, необхідний для початкової роботи системи. Завдяки цьому, при першому запуску додатку, користувачам доступні всі необхідні елементи для створення дизайнів.

Наступна важлива функція — це `calculatePlantSize`, яка використовується для обчислення розмірів рослин на канвасі в залежності від їхньої позиції. Ця функція є критичною для забезпечення реалістичного вигляду дизайнів, створених користувачами:

```
function calculatePlantSize(y, canvasHeight, minSize, maxSize) {
  const horizonY = canvasHeight * (currentHorizon / 100);
  const distanceFromHorizon = y - horizonY;
  const yRelative = distanceFromHorizon / horizonY;
  minSize = minSize + (minSize) * (currentHorizon / 100);
  let size = minSize + (maxSize - minSize) * Math.log1p(yRelative *
(Math.E - 1));
if (y < (horizonY - (minSize * 0.2))) {
    size = minSize;} return size;}
```

Ця функція бере координату Y рослини, висоту канвасу, а також мінімальні та максимальні розміри рослини. Вона обчислює відстань від горизонту і на основі цього розраховує відносний розмір рослини. Якщо рослина розташована вище горизонту, її розмір встановлюється на мінімальне значення. Цей алгоритм забезпечує реалістичне масштабування елементів в залежності від їхньої позиції на канвасі, що додає глибини і перспективи до дизайнів.

Ще одна важлива функція — це `drawCanvas`, яка відповідає за відображення канвасу з усіма розміщеними елементами. Ця функція оновлює канвас при кожній зміні, забезпечуючи інтерактивність і візуальну відповідність користувацьких дій:

```
function drawCanvas() {
  drawBackground(currentBackground.src);
  for (let plant of plants) {
    ctx.drawImage(plant.img,    plant.x,    plant.y,    plant.size,
plant.size); }}
function drawBackground(src) {
  const background = new Image();
  background.src = src;
  background.onload = function () {
    ctx.clearRect(0, 0, canvas.width, canvas.height);
    ctx.drawImage(background, 0, 0, canvas.width, canvas.height);
    for (let plant of plants) {
```

Продовження коду:

```
            ctx.drawImage(plant.img,    plant.x,    plant.y,    plant.size,
plant.size);} };
```

Функція `drawCanvas` спочатку викликає функцію `drawBackground`, яка очищує канвас і малює новий фон. Потім, вона перебирає всі розміщені рослини і декорує їх зображення на канвасі. Ця функція виконується щоразу, коли змінюється фон або додається новий елемент, забезпечуючи актуальне відображення всіх елементів.

Функція `drawBackground` відповідає за завантаження і відображення фону на канвасі. Вона очищує канвас перед малюванням нового фону і повторно малює всі елементи, щоб зберегти їх позиції на новому фоні. Ця функція забезпечує плавний перехід між різними фонами і реалістичне відображення усіх розміщених елементів.

Таким чином, основні функції серверної частини, такі як `populateDatabase`, `calculatePlantSize` і `drawCanvas`, забезпечують інтерактивність та ефективне управління даними у веб-додатку. Вони дозволяють користувачам легко створювати та маніпулювати дизайнами, забезпечуючи реалістичне відображення та інтерактивний досвід. Інші важливі функції включають обробку аутентифікації користувачів, збереження проектів та управління повідомленнями, що доповнює загальну функціональність додатку.

Веб-сервіс "Flexible landscape design" розроблений з урахуванням сучасних тенденцій адаптивного дизайну. Це означає, що інтерфейс автоматично підлаштовується під розмір екрану пристрою, на якому він використовується. Незалежно від того, чи це настільний комп’ютер, планшет або мобільний телефон, користувачі отримують зручний доступ до всіх функцій платформи.

Для забезпечення коректної роботи веб-сервісу рекомендується використовувати сучасні браузери, такі як Google Chrome, Mozilla Firefox, Microsoft Edge та Safari. Використання застарілих версій браузерів може призвести до некоректного відображення або функціонування деяких елементів інтерфейсу, тому важливо своєчасно оновлювати браузер до останньої версії.

Крім того, для забезпечення безпеки та конфіденційності даних користувачів,

веб-сервіс "Flexible landscape design" використовує протокол HTTPS для зашифрованого передавання даних між користувачем і сервером. Це дозволяє уникнути можливих атак з перехопленням даних та забезпечує захист особистої інформації користувачів.

Для підтримки інтерактивності та швидкості реакції веб-сервісу, клієнтська частина додатку використовує сучасні технології, які дозволяють створювати динамічний та зручний інтерфейс для користувачів.

5 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Загальні відомості

Тестування розробленого програмного забезпечення "Flexible landscape design" є критичним етапом в процесі розробки, який забезпечує якість, надійність та ефективність системи. В цьому розділі розглянемо цілі тестування, використані методи та процес планування тестування.

Основними цілями тестування системи "Flexible landscape design" були виявлення та виправлення помилок, перевірка відповідності функціональних і нефункціональних вимог, забезпечення стабільності та продуктивності системи, а також підтвердження того, що система відповідає очікуванням користувачів. Тестування було спрямоване на перевірку всіх компонентів системи, включаючи фронтенд, бекенд, базу даних та інтерактивні елементи, щоб гарантувати їх коректну взаємодію і відповідність специфікаціям.

Для забезпечення всебічного тестування системи були використані різні методи тестування, включаючи модульне тестування, інтеграційне тестування, системне тестування та тестування інтерфейсу користувача.

Модульне тестування було використане для перевірки окремих функцій і методів серверної частини. Це дозволило виявити помилки на ранніх етапах розробки і забезпечити високу якість коду. Модульне тестування включало тестування основних функцій, таких як populateDatabase, calculatePlantSize та drawCanvas.

Інтеграційне тестування перевіряло взаємодію між різними модулями системи. Це включало тестування взаємодії між фронтендом і бекендом, перевірку коректної роботи API та взаємодії з базою даних. Інтеграційне тестування допомогло виявити проблеми з сумісністю та забезпечити плавну роботу системи.

Системне тестування включало перевірку всієї системи як єдиного цілого. Це тестування було спрямоване на виявлення помилок, які могли виникнути при взаємодії всіх компонентів системи. Системне тестування включало перевірку всіх функціональних вимог, таких як автентифікація користувачів, управління проектами, завантаження файлів та обробка повідомлень.

Тестування інтерфейсу користувача перевіряло зручність та ефективність використання системи з точки зору кінцевого користувача. Це тестування включало перевірку інтерфейсу на різних пристроях та браузерах, а також тестування інтерактивних компонентів, таких як канвас для створення дизайнів.

Планування тестування розпочалося з визначення основних цілей та вимог до системи. Було розроблено детальний план тестування, який включав опис тестових сценаріїв, тестових випадків, методів тестування та критеріїв прийняття.

Першим кроком у плануванні було створення тестових сценаріїв для кожної функціональної вимоги системи. Ці сценарії включали докладний опис дій, які необхідно виконати, очікувані результати та критерії успішного проходження тесту. Тестові сценарії були розроблені для покриття всіх можливих варіантів використання системи, включаючи позитивні та негативні сценарії.

Наступним кроком було створення тестових випадків на основі тестових сценаріїв. Тестові випадки включали конкретні вхідні дані, очікувані результати та кроки для виконання тесту. Кожен тестовий випадок був пов'язаний з однією або декількома вимогами до системи, що дозволяло відстежувати покриття тестування.

Процес тестування був організований у кілька етапів, кожен з яких включав виконання тестових випадків, збір та аналіз результатів, а також виправлення виявлених помилок. Першим етапом було модульне тестування, яке виконувалося розробниками для перевірки окремих функцій. Після успішного проходження модульного тестування проводилося інтеграційне тестування для перевірки взаємодії між модулями. Завершальним етапом було системне тестування та тестування інтерфейсу користувача, яке включало перевірку всієї системи як єдиного цілого.

5.2 Функціональне тестування

Функціональне тестування є важливою частиною процесу забезпечення якості програмного забезпечення. Воно включає перевірку функціональних компонентів системи на відповідність вимогам та специфікаціям. У системі "Flexible landscape design" функціональне тестування охоплювало різні

компоненти, включаючи управління користувачами, створення і редагування проектів, завантаження зображень, а також інтерактивні функції дизайну.

Функціональне тестування включало перевірку наступних компонентів системи:

- аутентифікація користувачів: тестування реєстрації, входу, виходу та перевірки прав доступу;
- управління користувачами: перевірка можливості додавання, редагування та видалення користувачів адміністратором;
- створення та управління проектами: тестування функцій створення, редагування, збереження та видалення проектів користувачами;
- завантаження зображень: перевірка процесу завантаження зображень рослин, декору та фонів, а також їх коректного відображення у системі;
- інтерактивний дизайн: тестування функцій додавання, переміщення та зміни розміру елементів на канвасі.

Тестування користувацького інтерфейсу проводилось з метою перевірки зручності та ефективності використання системи кінцевими користувачами. Це включало перевірку відображення елементів інтерфейсу на різних розмірах екрану та пристроях, тестування інтерактивних компонентів, таких як канвас для створення дизайнів, форми для введення даних та кнопки для взаємодії з системою. Використовувались інструменти Selenium для автоматизації тестування інтерфейсу, що включало тестування сценаріїв навігації між сторінками, заповнення та відправки форм, а також взаємодії з інтерактивними елементами.

Для забезпечення всебічного тестування системи були розроблені різноманітні тестові сценарії, включаючи позитивні та негативні випадки. Наприклад, для аутентифікації користувачів розроблялись сценарії для успішного входу та виходу, неправильного введення пароля, спроби доступу до захищених ресурсів без авторизації. Управління користувачами включало сценарії для додавання нових користувачів, редагування існуючих користувачів, видалення користувачів та перевірки прав доступу. Створення та управління проектами охоплювало сценарії для створення нового проекту, редагування існуючого

проекту, збереження проекту з різними параметрами та видалення проекту. Завантаження зображень передбачало сценарії для завантаження зображень різних типів, перевірки коректності завантаження та відображення. Інтерактивний дизайн включав сценарії для додавання елементів на канвас, переміщення елементів, зміни їх розміру, збереження дизайну та перевірки коректного відображення збережених проектів.

Перед проведенням мануальних тестів веб-сервісу "Flexible landscape design" важливо переконатися, що всі основні функції додатку працюють належним чином та відповідають вимогам специфікації. Мануальні тести спрямовані на перевірку ключових аспектів, таких як вхід на головну сторінку, створення нового проекту, зміна розміру рослин у дизайні та відправлення повідомлення через контактну форму. Після проведення тестів аналізуються фактичні результати порівняно з очікуваними, щоб визначити коректність роботи додатку та його відповідність вимогам.

Ці мануальні тести спрямовані на забезпечення якості та надійності веб-сервісу "Flexible landscape design" та забезпечення задоволення користувачів та ефективної роботи системи (див. табл. 5.1).

Таблиця 5.1 – Результати мануальних тестів

№	Опис дії	Очікуваний результат	Фактичний результат	Результат виконання
1	Реєстрація нового користувача з валідними особистими даними	Користувач повинен бути успішно зареєстрований у системі	Користувач успішно зареєстрований	Пройдено
2	Зміна паролю інсайдером, який має доступ до аккаунту	Система має відхилити зміну паролю та повідомити користувача про спробу недозволеного доступу	Зміна паролю відхилена, користувач повідомлений про спробу недозволеного доступу	Пройдено

Кінець таблиці 5.1

3	Надсилання запиту через контактну форму з некоректними даними	Система повинна відхилити запит і повідомити користувача про необхідність виправлення введених даних	Запит відхилено, користувач повідомлений про необхідність виправлення введених даних	Пройдено
4	Додавання нового зображення в галерею робіт адміністратором	Зображення має бути успішно додане до галереї робіт	Зображення успішно додане до галереї робіт	Пройдено
5	Оновлення контактної інформації адміністратором	Контактна інформація має бути успішно оновлена у системі	Контактна інформація успішно оновлена у системі	Пройдено
6	Надсилання повідомлення від користувача через контактну форму з валідними даними	Повідомлення повинно бути успішно збережено у базі даних і відправлено адміністраторам для подальшого розгляду	Повідомлення успішно збережено у базі даних і відправлено адміністраторам	Пройдено

Усі шість тестів були успішно виконані, що свідчить про правильну та надійну роботу бекенд частини веб-сервісу "Flexible landscape design". Вони охопили різні аспекти функціональності, безпеки та взаємодії з користувачем, такі як реєстрація, управління профілем, обробка контенту, безпека, взаємодія через контактну форму та сповіщення адміністраторів. Всі тести підтвердили, що

система працює згідно з очікуваннями і відповідає вимогам до функціональності та безпеки.

5.3 Нефункціональне тестування

Нефункціональне тестування спрямоване на перевірку аспектів, що не стосуються безпосередньо функціональності системи, але мають важливе значення для її загальної якості та стабільності. Це включає тестування продуктивності, безпеки та сумісності з різними браузерами та пристроями.

Тестування продуктивності проводилось для оцінки швидкодії та здатності системи витримувати навантаження. Основні аспекти тестування продуктивності включали вимірювання часу відгуку системи на різні запити, такі як завантаження сторінок, обробка форм та взаємодія з канвасом. Навантажувальне тестування передбачало імітацію великої кількості одночасних користувачів для перевірки, як система справляється з високим навантаженням. Це включало перевірку стабільності та відсутності помилок при збільшенні кількості користувачів. Стрес-тестування перевіряло поведінку системи під екстремальними умовами, такими як значне збільшення обсягу даних або кількості запитів.

Тестування безпеки включало перевірку різних аспектів, щоб гарантувати захист даних користувачів та безпечну роботу системи. Це охоплювало перевірку автентифікації та авторизації, тестування механізмів входу в систему, управління сесіями та прав доступу.

Таким чином, нефункціональне тестування допомогло забезпечити високу продуктивність, безпеку та сумісність системи "Flexible landscape design", що є суттєвим для забезпечення позитивного користувацького досвіду та стабільної роботи додатку.

ВИСНОВКИ

У ході даної роботи було розроблено веб-сервіс "Flexible landscape design", який призначений для ефективного управління та презентації послуг у сфері ландшафтного дизайну. Основною метою проекту було створення функціонального та зручного для користувачів веб-сайту, що забезпечує інтуїтивно зрозумілий інтерфейс та надає можливість компаніям демонструвати свої роботи, взаємодіяти з клієнтами та підвищувати якість обслуговування.

На початкових етапах було проведено аналіз предметної галузі, який включав огляд існуючих рішень та виявлення їх слабких місць. Це дозволило визначити ключові вимоги до системи та сформулювати функціональні та нефункціональні вимоги. Основними функціональними вимогами стали реєстрація та авторизація користувачів, управління профілями, створення та редагування послуг і проектів, а також забезпечення зворотного зв'язку з користувачами.

Базуючись на визначених вимогах, було спроєктовано базу даних, яка забезпечує ефективне зберігання та обробку даних. Структура бази даних включає таблиці для зберігання інформації про користувачів, послуги, проекти та контактні повідомлення. Для кожної таблиці були визначені необхідні поля та зв'язки між ними, що забезпечує цілісність та консистентність даних.

Проєктування інтерфейсу користувача зосереджувалося на створенні інтуїтивно зрозумілого та адаптивного дизайну, що забезпечує зручність користувачів на різних пристроях. Було розроблено навігаційне меню, секції з девізом компанії, описом послуг, галереєю проектів та контактною формою. Особлива увага приділялася доступності інтерфейсу для користувачів з обмеженими можливостями.

Розробка модулів програмного забезпечення включала створення та взаємодію таких основних модулів, як управління користувачами, управління послугами, управління проектами, зворотний зв'язок та забезпечення безпеки. Кожен модуль був детально описаний, включаючи процеси створення, редагування та видалення даних, а також забезпечення безпеки та інтеграції з фронтендом через RESTful API.

Модуль управління користувачами реалізовував функції реєстрації, аутентифікації, авторизації та управління профілями користувачів. Модуль управління проектами забезпечував можливість додавання, редагування, видалення та перегляду проектів у вигляді галереї зображень. Було також розроблено модуль зворотного зв'язку, який дозволяє користувачам зв'язуватися з компанією через контактну форму, а адміністратори отримують повідомлення про нові запити.

В результаті виконаної роботи було створено сучасний, зручний та функціональний веб-сервіс, який відповідає вимогам ринку та забезпечує високу якість обслуговування клієнтів. Використання сучасних технологій, таких як JavaScript, забезпечило ефективність та продукти вність системи. Веб-сервіс "Flexible landscape design" дозволить компаніям підвищити свою конкурентоспроможність, забезпечуючи зручний доступ до інформації про послуги та проекти, а також покращуючи комунікацію з клієнтами.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Houzz [Електронний ресурс]. – Режим доступу: https://www.houzz.com. – Назва з екрану.
2. SketchUp [Електронний ресурс]. – Режим доступу: https://www.sketchup.com. – Назва з екрану.
3. Pro Landscape [Електронний ресурс]. – Режим доступу: https://www.prolandscape.com. – Назва з екрану.
4. Трофименко О. Г. Основи програмної інженерії : навч.-метод. посібник / О. Г. Трофименко, С. Ю. Манаков, Д. Г. Ларін. – Одеса : Фенікс, 2022. – 197 с.
5. Федоров А. Б. Веб-дизайн: теорія і практика / А. Б. Федоров, М. В. Чернов. – М.: Издательство МГУ, 2020. – 320 с.
6. Сидоров П. А. Безпека веб-додатків / П. А. Сидоров. – М.: Видання «Кодекс», 2021. – 270 с.
7. Золотарьов В. П. HTML і CSS для новачків / В. П. Золотарьов. – Л.: Видання «Світ», 2019. – 200 с.
8. Смит Дж. JavaScript для початківців / Дж. Смит. – Н.-Й.: Видавництво «TechBooks», 2021. – 280 с.
9. Коваленко О. Г. Тестування програмного забезпечення: методи і інструменти / О. Г. Коваленко. – Х.: ХНУРЕ, 2017. – 240 с.
10. Браун Л. В. Основи веб-дизайну / Л. В. Браун. – К.: Видання «Освіта», 2018. – 310 с.
11. Чернишев А. В. Проектування баз даних / А. В. Чернишев. – М.: Издательство «Вузовская книга», 2020. – 290 с.
12. Фармер Дж. Сучасні методи веб-розробки / Дж. Фармер. – Л.: Видання «IT Світ», 2019. – 350 с.
13. Карпова Н. В. UX-дизайн: основи і приклади / Н. В. Карпова. – Х.: ХНУРЕ, 2019. – 230 с.

ДОДАТОК А

Звіт результатів перевірки на унікальність тексту у базі ХНУРЕ

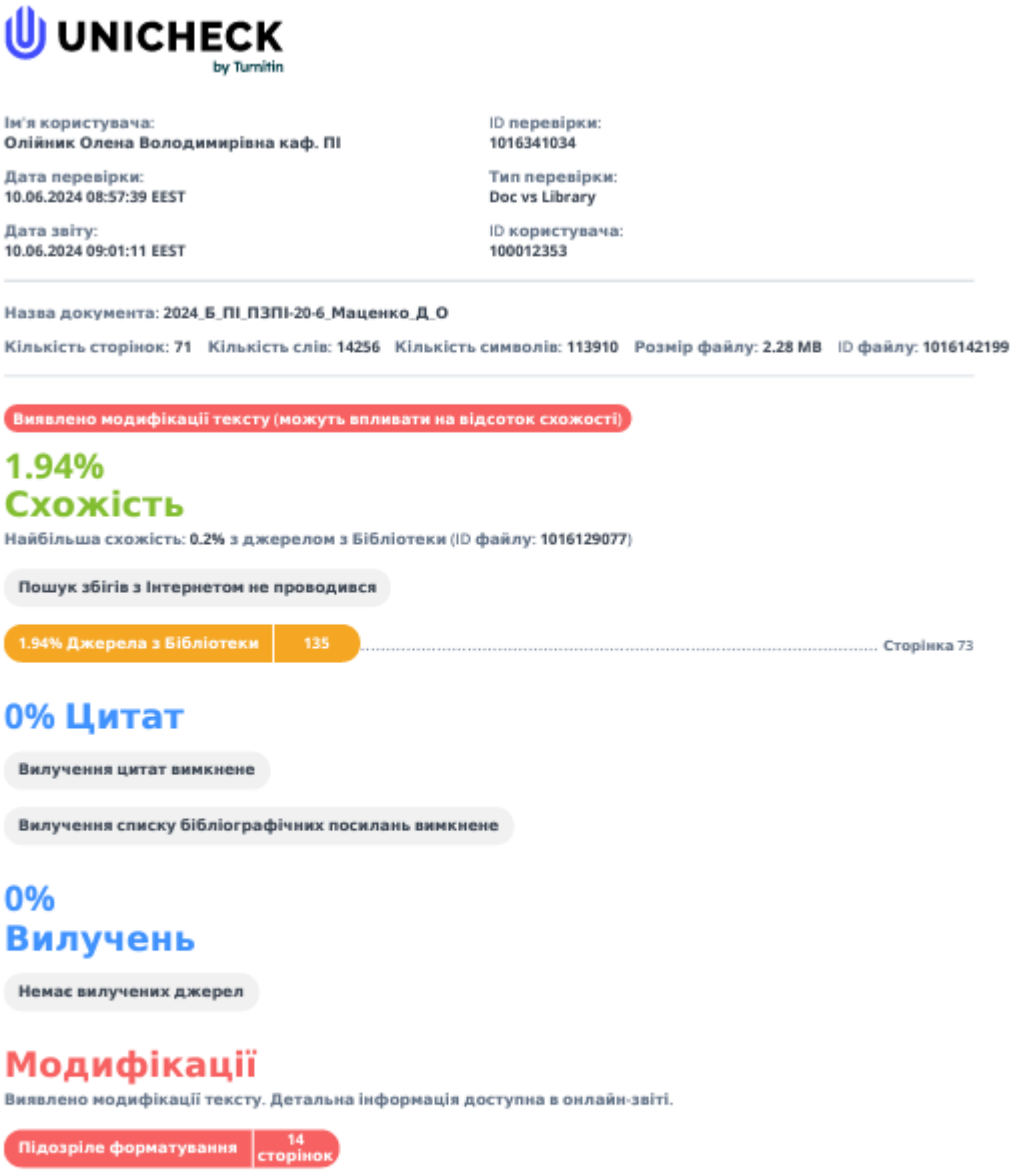

Рисунок А.1 – Результат перевірки плагіату

ДОДАТОК Б

Функції ініціалізації бази даних

Продовження коду ініціалізації бази даних початковими даними:

```
  const decorFiles = fs.readdirSync(decorDir);
  for (const file of decorFiles) {
    const existingDecor = await DecorObject.findOne({ imageUrl: `/assets/decor/${file}` });
    if (!existingDecor) {
      const sizes = parseSizes(file);
      const newDecor = new DecorObject({
        name: path.parse(file).name,
        description: `Description of ${path.parse(file).name}`,
        minSize: sizes.minSize,
        maxSize: sizes.maxSize,
        imageUrl: `/assets/decor/${file}`,
        type: 'decor'
      });
      await newDecor.save();}}}
```

Продовження коду ініціалізації бази даних за допомогою populateDatabase():

```
  // Populate plants (trees)
  const treeFiles = fs.readdirSync(plantsDir);
  for (const file of treeFiles) {
    const existingPlant = await DecorObject.findOne({ imageUrl: `/assets/plants/${file}` });
    if (!existingPlant) {
      const sizes = parseSizes(file);
      const newPlant = new DecorObject({
        name: path.parse(file).name,
        description: `Description of ${path.parse(file).name}`,
        minSize: sizes.minSize,
        maxSize: sizes.maxSize,
        imageUrl: `/assets/plants/${file}`,
        type: 'plant'
      });
      await newPlant.save();}}
```

```
        // Populate decor
        const decorFiles = fs.readdirSync(decorDir);
        for (const file of decorFiles) {
          const existingDecor = await DecorObject.findOne({ imageUrl: `/assets/decor/${file}` });
          if (!existingDecor) {
            const sizes = parseSizes(file);
            const newDecor = new DecorObject({
              name: path.parse(file).name,
              description: `Description of ${path.parse(file).name}`,
              minSize: sizes.minSize,
              maxSize: sizes.maxSize,
              imageUrl: `/assets/decor/${file}`,
              type: 'decor'
            });
            await newDecor.save();  }}
        // Populate backgrounds (houses)
        const houseFiles = fs.readdirSync(housesDir);
        for (const file of houseFiles) {
          const existingBackground = await Background.findOne({ imageUrl: `/assets/houses/${file}` });
          if (!existingBackground) {
            const nameParts = path.parse(file).name.split('_');
            const horizon = nameParts.length > 1 && !isNaN(nameParts[1]) ? parseInt(nameParts[1], 10) : 50;
            const newBackground = new Background({
              name: path.parse(file).name,
              description: `Description of ${path.parse(file).name}`,
              imageUrl: `/assets/houses/${file}`,
              horizon: horizon,
              width: 800,
              height: 600});
            await newBackground.save();}}}
```

ДОДАТОК В

Слайди презентації

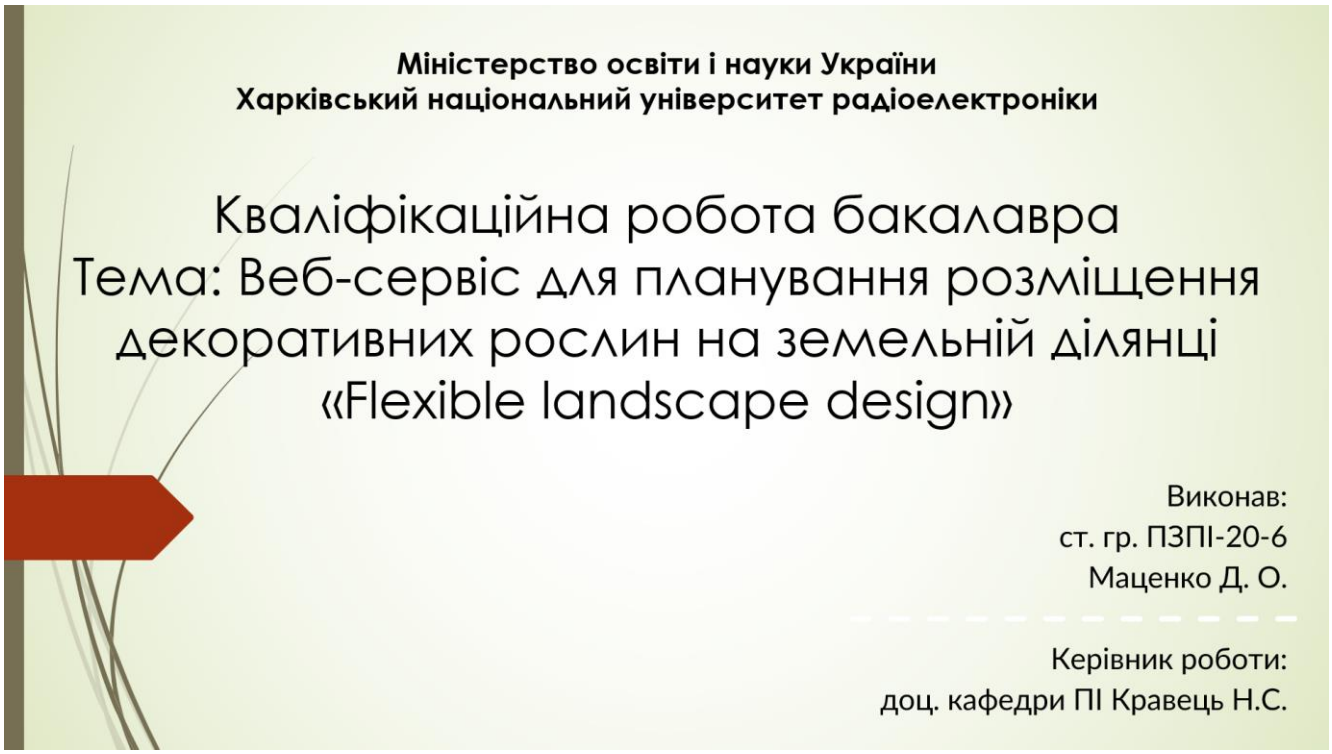

Рисунок В.1 – Слайд 1

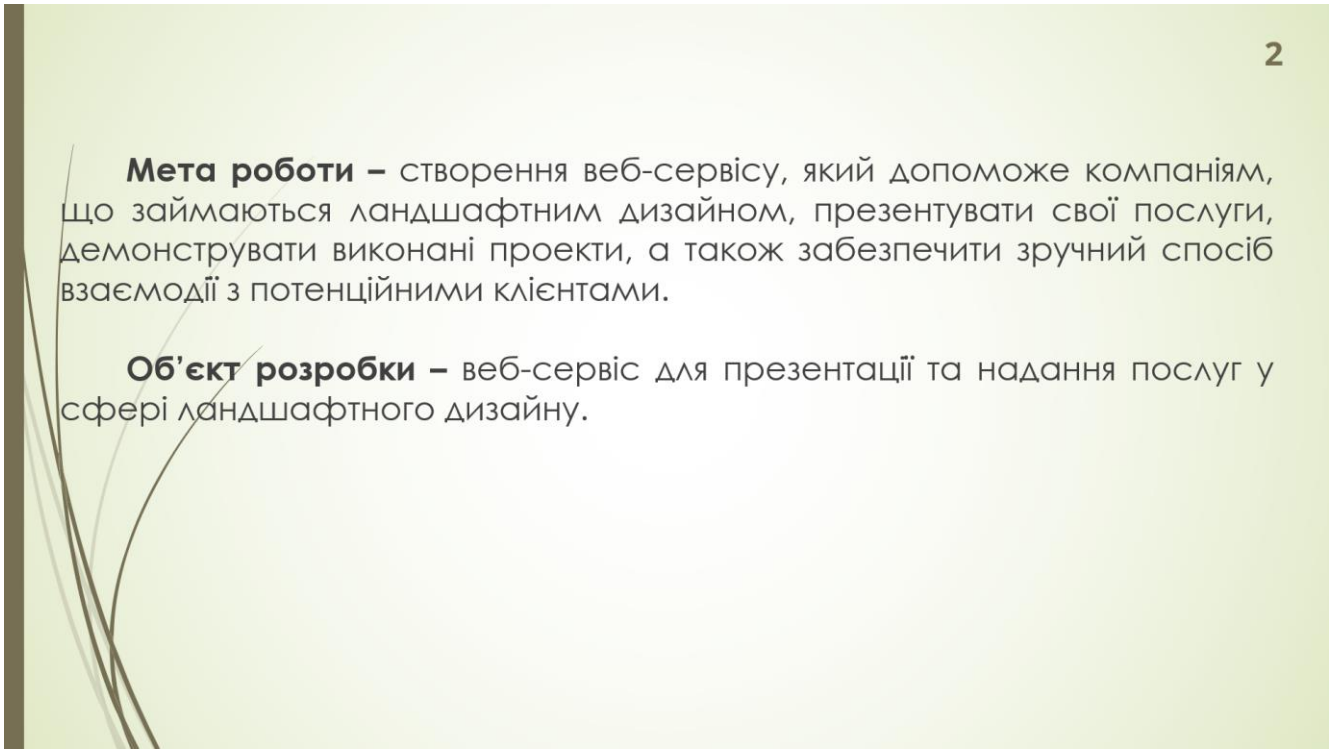

Рисунок В.2 – Слайд 2

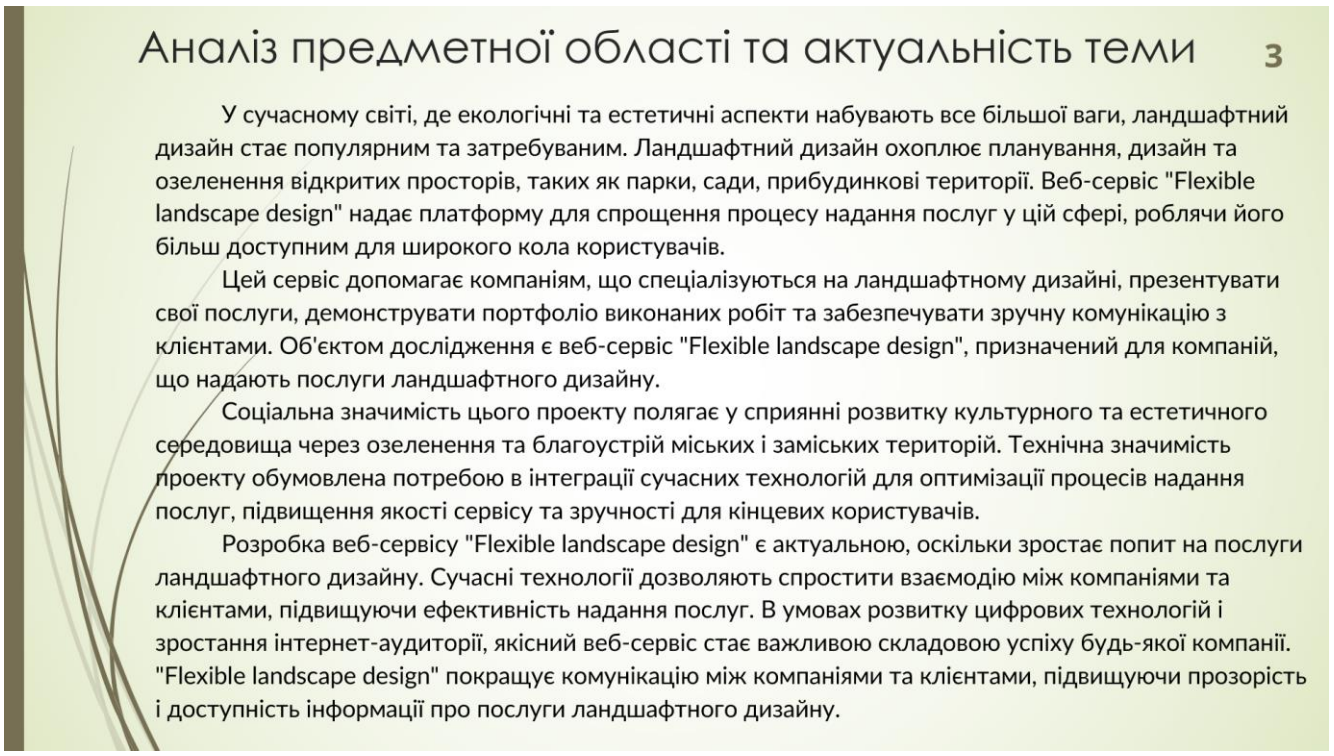

Рисунок В.3 – Слайд 3

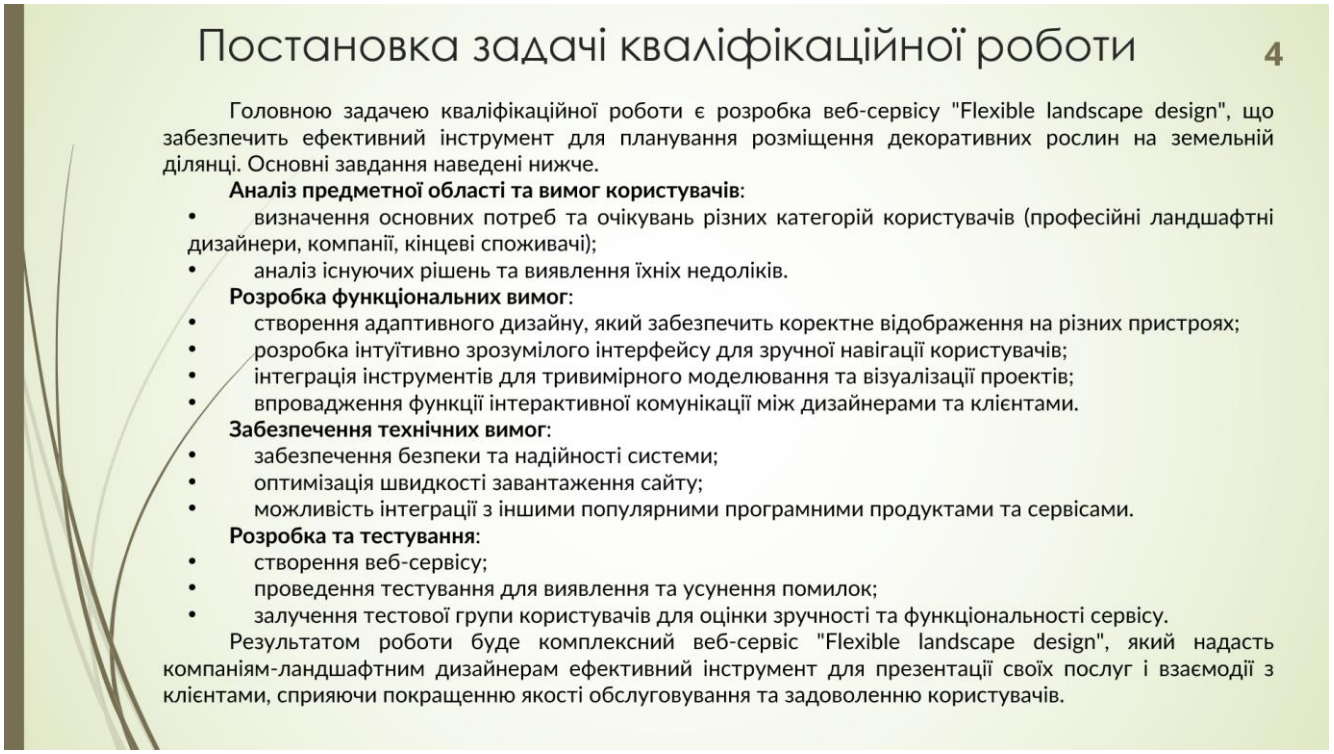

Рисунок В.4 – Слайд 4

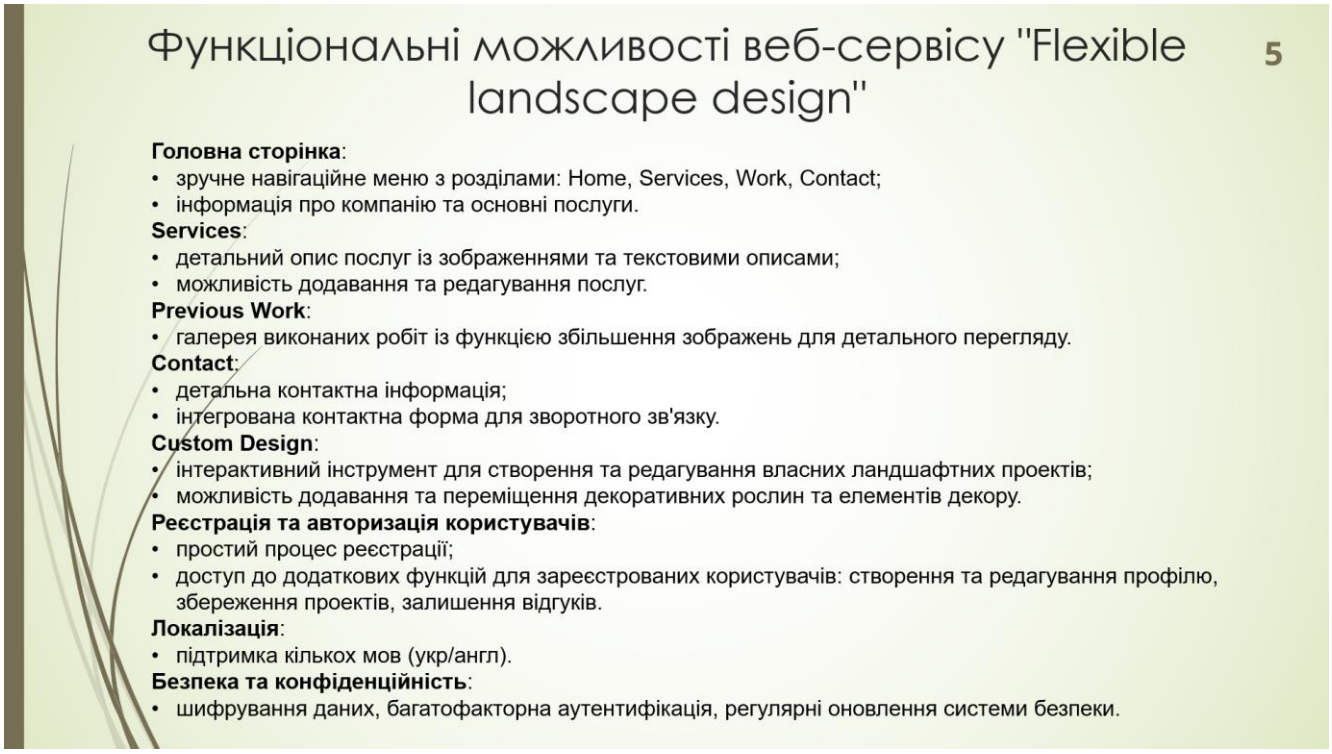

Рисунок В.5 – Слайд 5

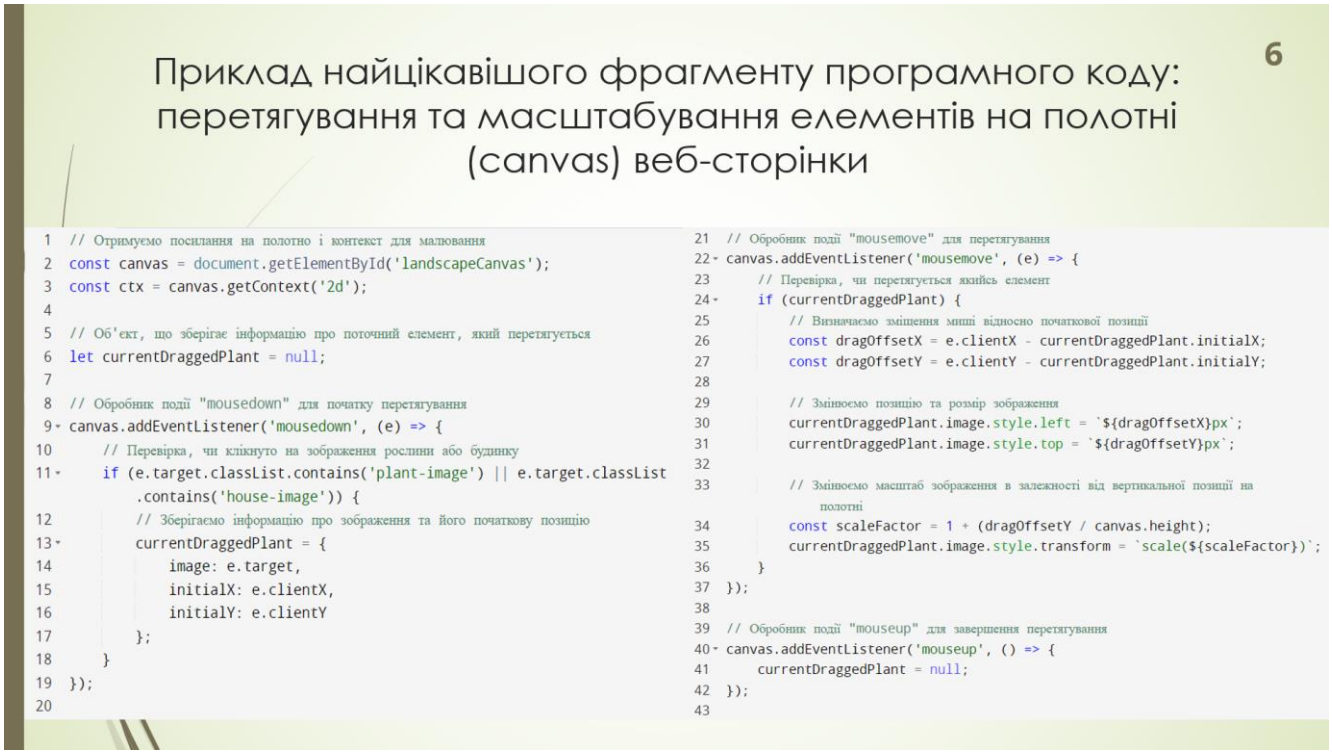

Рисунок В.6 – Слайд 6

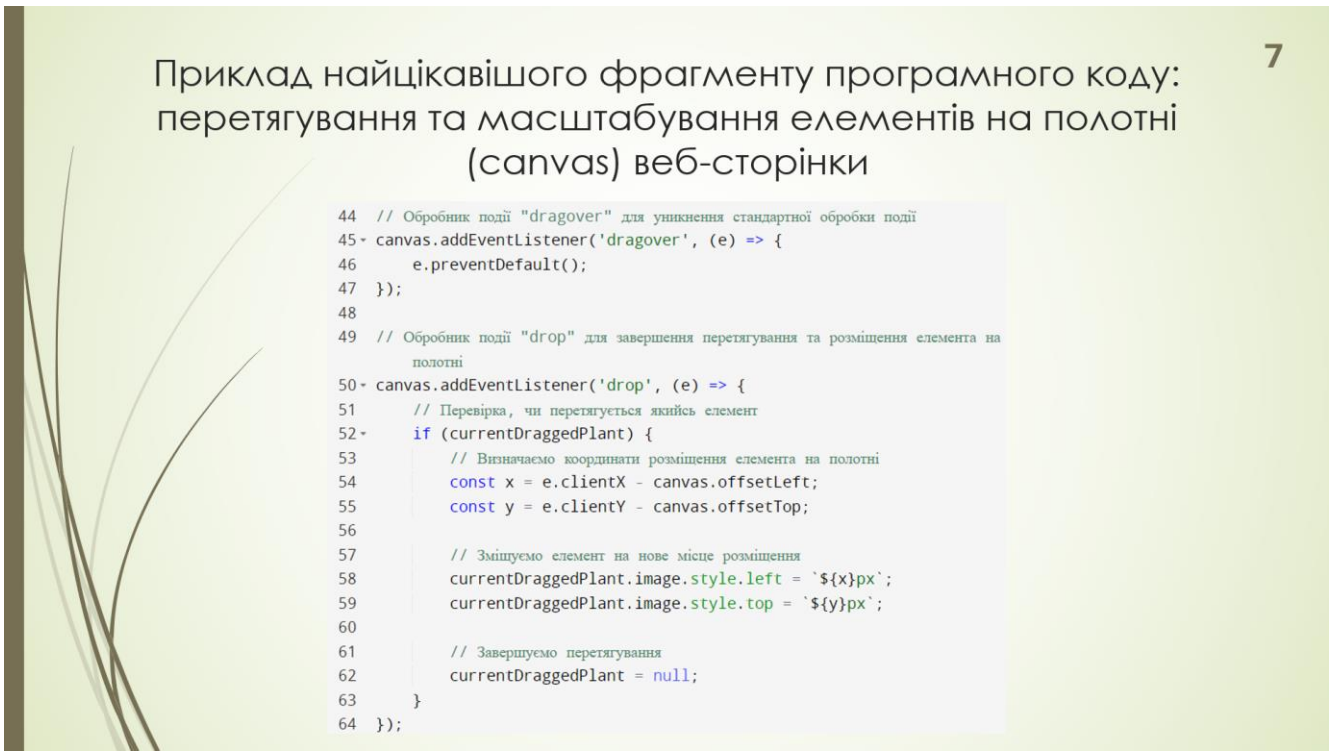

Рисунок В.7 – Слайд 7

Рисунок В.8 – Слайд 8

Рисунок В.9 – Слайд 9

Рисунок В.10 – Слайд 10

Рисунок В.11 – Слайд 11

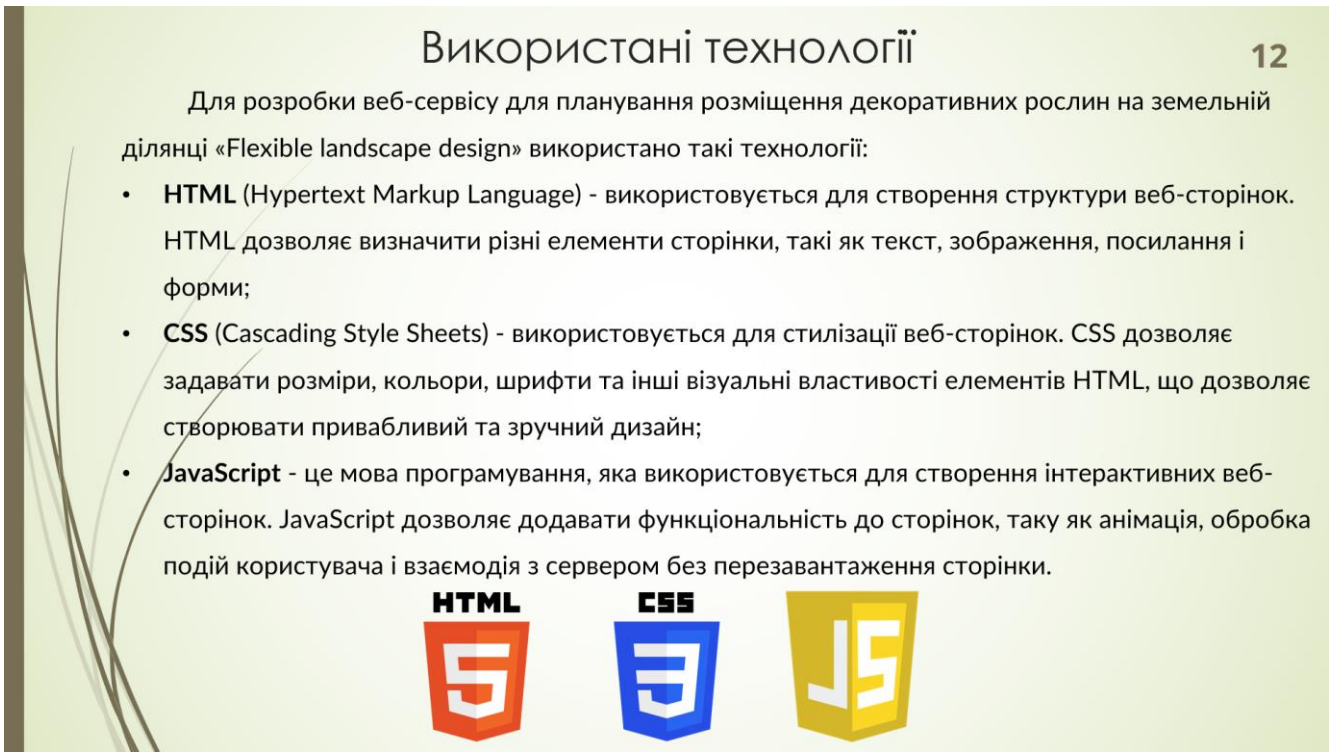

Рисунок В.12 – Слайд 12

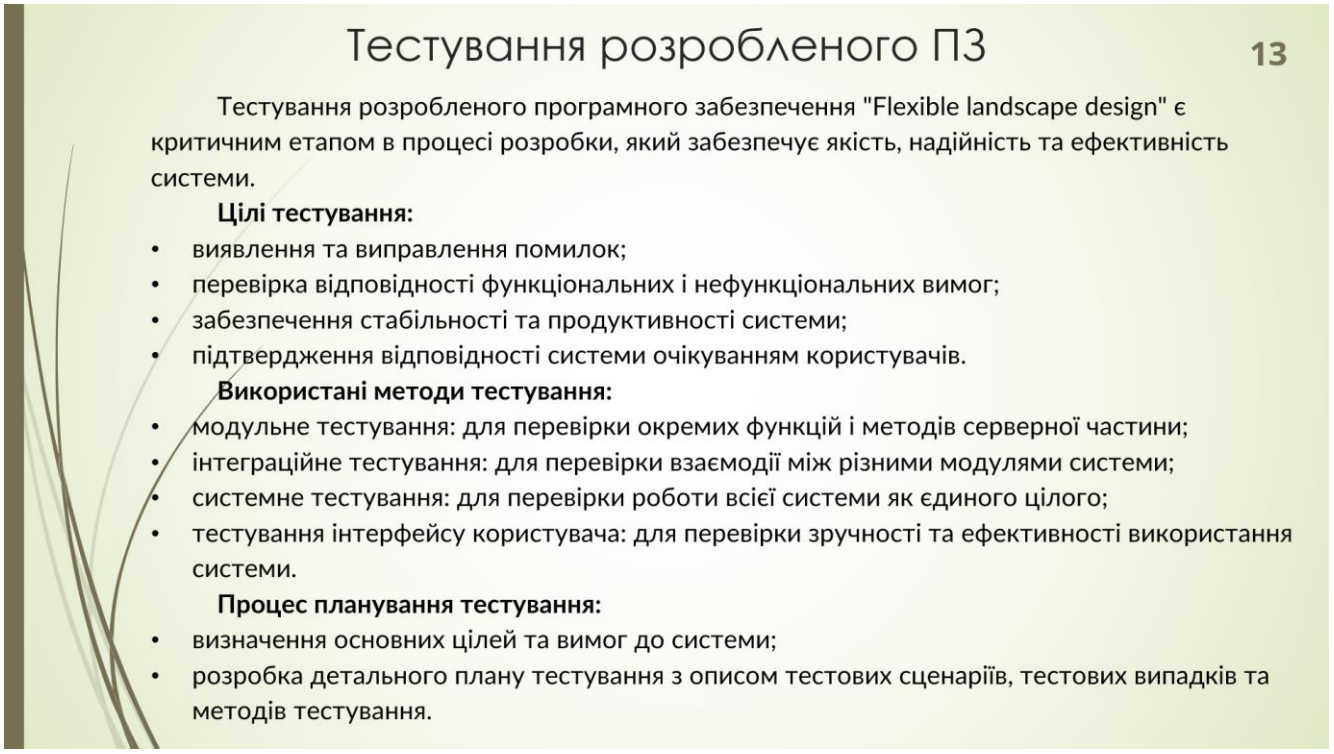

Рисунок В.13 – Слайд 13

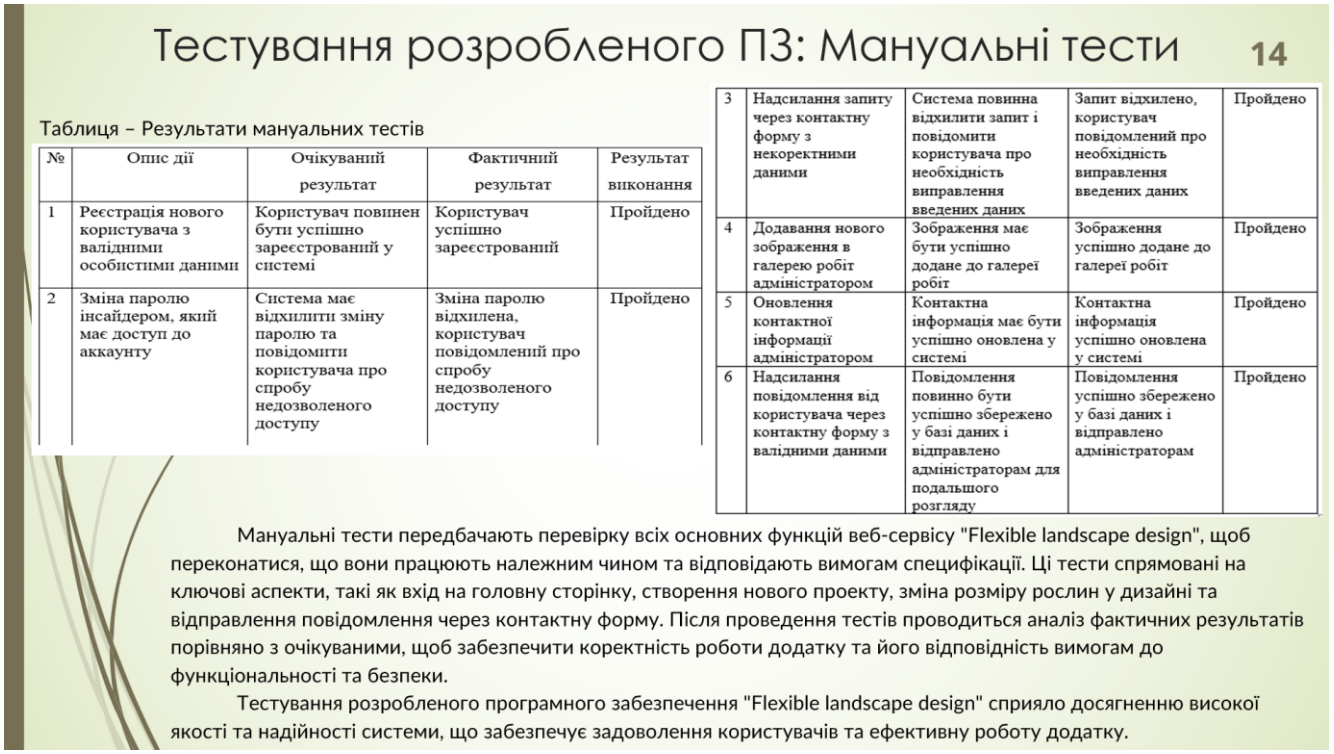

Рисунок В.14 – Слайд 14

Рисунок В.15 – Слайд 15

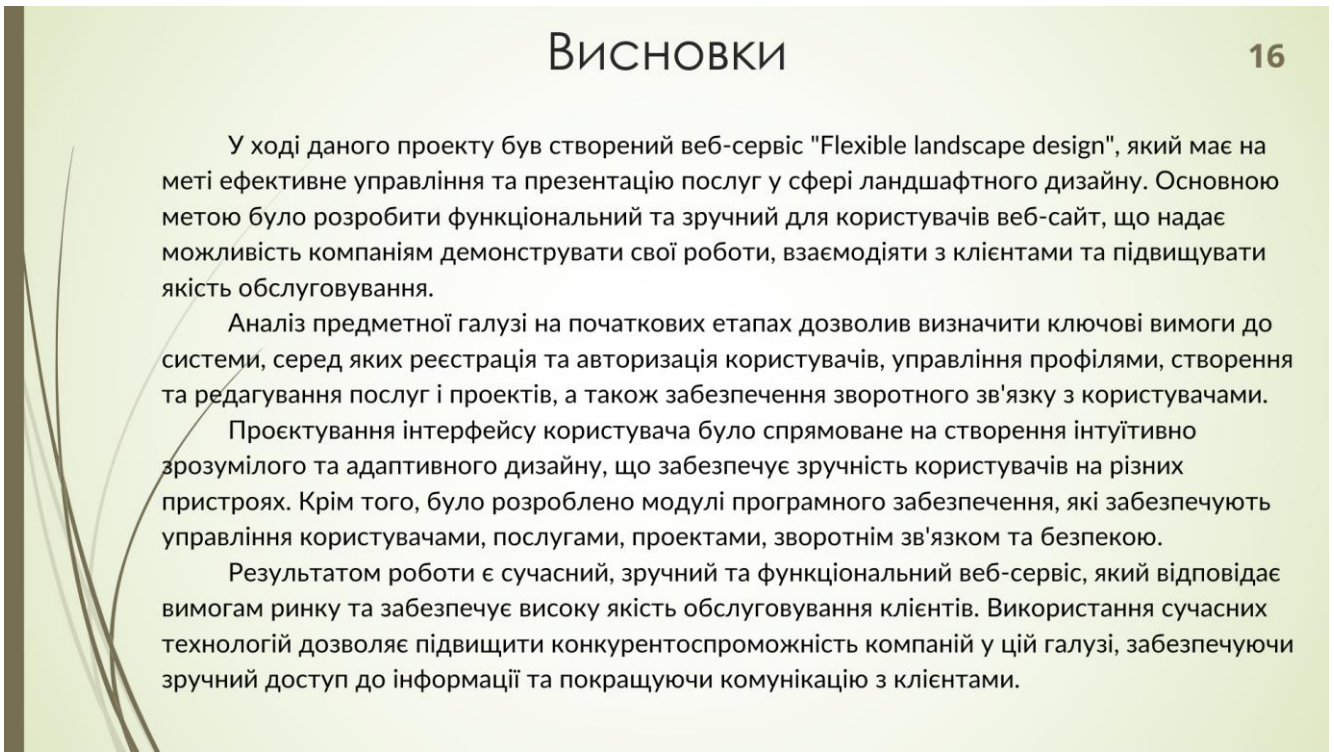

Рисунок В.16 – Слайд 16