

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

# **КВАЛІФІКАЦІЙНА РОБОТА**

на здобуття ступеня вищої освіти магістр

на тему: **«Формування стеку сучасних вебтехнологій для забезпечення  
функціональності інтерактивних вебдодатків на прикладі  
реалізації квест-тесту»**

Виконала: здобувачка вищої освіти  
за освітньою програмою  
Інформаційні управляючі системи та  
технології спеціальності  
126 Інформаційні системи та технології  
ступеня вищої освіти магістр  
групи 126ІСТ\_мз\_2024[1]  
Дузенко Олександра Олександрівна  
Керівник: Копішинська Олена  
Петрівна  
Рецензент: Муравльов Володимир  
В'ячеславович

**Полтава – 2025 року**

**ПОЛТАВСЬКИЙ ДЕРЖАВНИЙ АГРАРНИЙ УНІВЕРСИТЕТ  
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЕКОНОМІКИ,  
УПРАВЛІННЯ, ПРАВА ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ  
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ**

Освітня програма Інформаційні управляючі системи та технології  
Спеціальність 126 Інформаційні системи та технології  
Рівень вищої освіти другий (магістерський)

**ЗАТВЕРДЖУЮ**

Завідувач кафедри

\_\_\_\_\_ Юрій УТКІН

«25» лютого 2025 року

**ЗАВДАННЯ  
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ**

**Дузенко Олександрі Олександрівни**

1. Тема кваліфікаційної роботи:  
«Формування стеку сучасних вебтехнологій для забезпечення функціональності інтерактивних вебдодатків на прикладі реалізації квест-тесту»,  
Керівник роботи: к. ф.-м. н., доцент, професор кафедри інформаційних систем та технологій Копішинська Олена Петрівна.  
Затверджено наказом закладу вищої освіти від «14» жовтня 2025 року № 1243-ст
2. Строк подання здобувачем вищої освіти роботи «25» листопада 2025 р.
3. Вихідні дані до роботи: науково-технічна література, інформаційні джерела мережі інтернет, наукові публікації фахових та міжнародних видань, статистичні дані аналітичних компаній, стандарти і специфікації W3C <https://www.w3.org/standards/>, JavaScript-фреймворк React, середовище Visual Studio Code <https://code.visualstudio.com/>, Leonardo AI
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):  
Розділ 1. Теоретичний огляд технологій розробки і забезпечення функцій вебдодатків  
Розділ 2. Обґрунтування стеку технологій проектування вебдодатків із заданим функціоналом  
Розділ 3. Опис практичної реалізації повної версії вебдодатку
5. Перелік графічного матеріалу: схеми, рисунки, діаграми за темою та об'єктом дослідження.

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання отримав
Оцінювання економічної ефективності результатів дослідження	Калініченко О. В., к. е. н., доцент, доцент кафедри економіки та публічного управління	04.10.2025	07.11.2025

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

**КАЛЕНДАРНИЙ ПЛАН**

№ з/п	Назва етапів роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1.	Вибір і затвердження теми роботи	20.01.2025 р.	
2.	Складання і затвердження розгорнутого плану та завдання на кваліфікаційну роботу	17.02.2025 р. – 25.02.2025 р.	
3.	Опрацювання джерел інформації	26.02.2025 р. – 13.05.2025 р.	
4.	Збір, вивчення і обробка інформації, необхідної для виконання роботи	14.05.2025 р.– 01.07.2025 р.	
5.	Виконання теоретико-методологічного розділу роботи	21.07.2025 р.– 01.08.2025 р.	
6.	Виконання дослідницько-аналітичного розділу роботи	04.08.2025 р.– 29.08.2025 р.	
7.	Виконання проектно-рекомендаційного розділу роботи	01.09.2025 р.– 01.10.2025 р.	
8.	Оцінювання економічної ефективності результатів дослідження	04.10.2025 р.– 07.11.2025 р.	
9.	Оформлення тексту роботи	10.11.2025 р.– 24.11.2025 р.	
10.	Попередній захист роботи на кафедрі	25.11.2025 р.	
11.	Доопрацювання роботи з урахуванням зауважень і пропозицій	26.11.2025 р.- 30.11.2025 р.	
12.	Нормоконтроль	01.12.2025 р. – 02.12.2025 р.	
13.	Захист кваліфікаційної роботи	04.12.2025 р.	

**Здобувач вищої освіти**

**Олександра ДУЗЕНКО**

**Керівник роботи**

**Олена КОПШИНСЬКА**

**ПОЛТАВСЬКИЙ ДЕРЖАВНИЙ АГРАРНИЙ УНІВЕРСИТЕТ  
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЕКОНОМІКИ, УПРАВЛІННЯ,  
ПРАВА ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ  
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ**

**ДУЗЕНКО ОЛЕКСАНДРА ОЛЕКСАНДРІВНА**

**«ФОРМУВАННЯ СТЕКУ СУЧАСНИХ ВЕБТЕХНОЛОГІЙ ДЛЯ  
ЗАБЕЗПЕЧЕННЯ ФУНКЦІОНАЛЬНОСТІ ІНТЕРАКТИВНИХ  
ВЕБДОДАТКІВ НА ПРИКЛАДІ  
РЕАЛІЗАЦІЇ КВЕСТ-ТЕСТУ»**

Освітньо-професійна програма  
Інформаційні управляючі системи та технології  
Спеціальність 126 Інформаційні системи та технології  
Ступінь вищої освіти магістр

**РЕФЕРАТ**

кваліфікаційної роботи на здобуття кваліфікації –  
магістр з інформаційних систем та технологій

Полтава – 2025 року

Кваліфікаційна робота складається із вступу, 3 розділів, висновків, списку використаних джерел (52 найменування), додатків. Кваліфікаційна робота містить 8 таблиць, 29 рисунків, викладена на 64 сторінках.

## **Основний зміст роботи**

У першому розділі «Теоретичний огляд технологій розробки і забезпечення функцій вебдодатків» представлено різноплановий огляд сучасних вебтехнологій, рейтингових діаграм і графіків популярності за даними провідних аналітичних компаній. Особливу увагу приділено розмежуванню функцій та інструментарію frontend- та backend-розробки, доцільності використання систем керування контентом CMS.

У другому розділі «Обґрунтування стеку технологій проектування вебдодатків із заданим функціоналом» здійснено аналіз продуктивності найбільш популярних фреймворків JavaScript, середовищ програмування, показано ефективність методу прототипування при розробці складних нетривіальних вебдодатків, що дозволило вирішити поставлені задачі практичного застосування комплексу технологій і представит соціально-значимий результат.

У третьому розділі «Опис практичної реалізації повної версії вебдодатку» наведено результати програмної реалізації складного двомовного інтерфейсу користувача, забезпечення обробки даних і збереження даних в LocalStorage, які відображають поведінку і вибір користувача при проходженні багатоступеневого тесту психологічного характеру. Особливий підрозділ присвячений інноваційній методології використання генеративних моделей штучного інтелекту для створення відповідної до тематики тесту графіки.

## **Висновки**

1. У ході виконання кваліфікаційної роботи проаналізовано значну кількість різноманітного інструментарію веброзробника, який впливає на якість та функціонал вебдодатків. В роботі описані та порівняні frontend-рішення, серед яких HTML&CSS, JavaScript і його фреймворки, проведений аналіз допоміжного програмного забезпечення: систем контролю версій та хмарних репозитаріїв, інтегровані середовища розробки, пакетні менеджери та контейнеризація.

2. Стек технологій включав найоптимальніші інструменти: окрім традиційних frontend-технологій HTML&CSS, JavaScript, були використані можливості фреймворку React, Visual Studio Code – редактор коду, обробка сцен в JASON, технологія збереження даних в LocalStorage, backend-фреймворк Node.js, Git як система контролю версій, GitHub – хмарний репозиторій.

3. Застосування методу попереднього прототипування вебдодатку з огляду на специфіку його призначення і використання дозволило провести перевірку функцій

на реалістичній моделі і реалізувати алгоритм обробки даних та шляхів масштабування подібних сторінок у розгалуженій структурі.

4. Використання системи штучного інтелекту Leonardo.Ai замість графічних редакторів дозволило згенерувати серію необхідних графічних елементів, які виглядали професійно й привабливо. Для отримання запланованого результату важливим моментом є генерування відповідного запиту (промпту), який також виконувався за допомогою іншої моделі штучного інтелекту – ChatGPT.

5. Проведено економічний аналіз. Загальна вартість подібного проєкту складає орієнтовно 54 775,8 грн при роботі 3-х спеціалістів протягом 10 днів разом із хостингом без урахування витрат на адміністрування та подальше обслуговування вебдодатку.

6. Проведено економічний аналіз. Загальна вартість подібного проєкту складає орієнтовно 54 775,8 грн при роботі 3-х спеціалістів протягом 10 днів разом із хостингом без урахування витрат на адміністрування та подальше обслуговування вебдодатку.

7. Подальше вдосконалення роботи можливо в напрямку розвитку підтримки SEO, оскільки React не досить добре з ним взаємодіє. Також можливе розширення функціоналу вебдодатку, додавання мовних профілів.

Подібний додаток може бути частиною більш масштабного проєкту будь-якої системи забезпечення послуг, розвинутий у соціально-значимий корисний продукт, запит на які в суспільстві стабільно зростає. Наприклад, психологічна служба, юридична компанія або є вибір різних опцій дизайну, будувannya тощо.

Достовірність результатів роботи базується на використанні відомих фреймворків мов програмування, відкритих моделей штучного інтелекту (LLM), даних ІТ-компаній та аналітичних груп, застосуванні загально прийнятих методів дослідження та обробки результатів.

### **Список публікацій здобувача**

1. Копішинська О. П., Дузенко О.О., Кіріченко С.Р. Тенденції використання штучного інтелекту в області програмної інженерії. *Modern technologies in education, work and science* : The VIII International scientific and practical conference, October 20-22, 2025, Krakow, Poland. P.179-181. URL: <https://eu-conf.com/wp-content/uploads/2025/08/modern-technologies-in-education-work-and-science.pdf>

2. Дузенко О. Особливості інструментів backend-розробки для забезпечення функцій вебдодатків. *Студентські роботи за науковою тематикою кафедри інформаційних систем та технологій ННІ ЕУП та ІТ ПДАУ: XXII щорічний міждисциплінарний семінар, 25 листопада 2025 р., м. Полтава. Полтава: ПДАУ, 2025. С. 48-50.*

## АНОТАЦІЯ

Дузенко О. О. «Формування стеку сучасних вебтехнологій для забезпечення функціональності інтерактивних вебдодатків на прикладі реалізації квест-тесту». Кваліфікаційна робота на правах рукопису.

Кваліфікаційна робота на здобуття ступеня вищої освіти магістр за освітньо-професійною програмою «Інформаційні управляючі системи та технології» спеціальності 126 Інформаційні системи та технології. Полтавський державний аграрний університет, Полтава, 2025.

Робота присвячена комплексному дослідженню різноманітних засобів розробки складних функціональних вебдодатків та імплементації обраного стеку технологій для вирішення поставленого завдання. Представлена оптимальна траєкторія від постановки завдання до досягнення мети як послідовність кроків життєвого циклу проєкту: дослідження-порівняння-вибір-обґрунтування-прототипування (моделювання)-реалізація-перевірка-застосування.

Ключові слова: HTML&CSS, JavaScript, JASON, LocalStorage, Git, прототипування, Leonardo.Ai.

## ANNOTATION

Duzhenko O. O. «Formation of a Modern Web Technology Stack to Ensure the Functionality of Interactive Web Applications: The Case of a Quest-Test Implementation». Qualification work (manuscript).

Qualification work for obtaining the master's degree in the educational and professional program *Information Management Systems and Technologies*, specialty 126 *Information Systems and Technologies*. Poltava State Agrarian University, Poltava, 2025.

The work is devoted to a comprehensive study of various tools for developing complex functional web applications and the implementation of a selected technology stack to solve a specific task. An optimal trajectory from task formulation to goal achievement is presented as a sequence of project life cycle stages: research – comparison – selection – justification – prototyping (modeling) – implementation – testing – application.

**Keywords:** HTML & CSS, JavaScript, JSON, LocalStorage, Git, prototyping, Leonardo.Ai.

## ЗМІСТ

ВСТУП .....	6
РОЗДІЛ 1. ТЕОРЕТИЧНИЙ ОГЛЯД ТЕХНОЛОГІЙ РОЗРОБКИ І ЗАБЕЗПЕЧЕННЯ ФУНКЦІЙ ВЕБДОДАТКІВ.....	9
1.1 Стан сучасних вебтехнологій та прикладні аспекти їх застосування .....	9
1.2 Особливості інструментів frontend- та backend-розробки для забезпечення функцій вебдодатків.....	13
1.3 Особливості систем керування контентом CMS .....	20
Висновки до розділу 1 .....	24
РОЗДІЛ 2. ОБҐРУНТУВАННЯ СТЕКУ ТЕХНОЛОГІЙ ПРОЄКТУВАННЯ ВЕБДОДАТКІВ ІЗ ЗАДАНИМ ФУНКЦІОНАЛОМ.....	25
2.1 Аналіз вимог до функцій та структури інтерактивного інформаційного вебдодатку.....	25
2.2 Визначення інструментарію проєктування та розробка прототипу вебдодатку.....	30
2.3 Порівняльний аналіз середовищ розробки та контролю версій.....	37
Висновки до розділу 2 .....	42
РОЗДІЛ 3 ОПИС ПРАКТИЧНОЇ РЕАЛІЗАЦІЇ ПОВНОЇ ВЕРСІЇ ВЕБДОДАТКУ .....	43
3.1 Програмна реалізація інтерфейсу користувача інтерактивного вебдодатку на прикладі квест-тесту.....	43
3.2 Застосування генеративного штучного інтелекту в розробці елементів графіки для сучасного дизайну вебдодатку та адаптивність.....	50
3.3 Економічне оцінювання вартості розробки вебдодатку з урахуванням застосованих вебтехнологій.....	57
Висновки до розділу 3 .....	60
ВИСНОВКИ.....	62
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	64
ДОДАТКИ.....	69

## ВСТУП

*Актуальність* теми кваліфікаційної роботи пов'язана із необхідністю систематизації відомостей про сучасні вебтехнології та застосуванням обґрунтованих критеріїв формування стеку відповідних інструментів, які здатні забезпечити широку функціональність створюваних вебдодатків заданого призначення. Технології прикладної вебіндустрії швидко видозмінюються, трансформуються, потребують системної раціоналізації на тлі удосконалення, розподілу функцій, появи нових мов програмування, фреймворків, бібліотек, зрештою, залучення штучного інтелекту.

Як відомо, вебдодатки використовуються в усіх сферах, де є процес отримання, переробки і використання інформації. Таким чином, постійний пошук шляхів вдосконалення функціональності та зовнішньої привабливості, інтерактивності вебдодатків у всіх соціальних, виробничих, наукових та бізнес аспектах, розвинення технологічних навичок веброзробника є об'єктивною необхідністю.

Розуміння та впровадження в роботу новітніх технологій стає критично важливим для досягнення конкурентних переваг вебдизайнерів та профільних ІТ-компаній.

*Зв'язок роботи з науковими програмами, темами.* Робота виконана у відповідності до науково–дослідної ініціативної теми «Організаційно–методологічні аспекти впровадження інформаційно–комунікаційних систем і технологій в управлінні діяльністю сучасних організацій та підприємств за умов переходу до цифрової економіки» ДРН 0123U105060, яка реалізується на кафедрі інформаційних систем та технологій Полтавського державного аграрного університету (ПДАУ).

*Метою* кваліфікаційної роботи є узагальнення теоретичних та практичних знань щодо вибору інструментів і методів дизайну сучасних інтерактивних вебдодатків, призначених для забезпечення зручної взаємодії користувача.

*Завдання* кваліфікаційної роботи включають:

- дослідження стану сучасних вебтехнологій розробки інтерфейсу користувача;
- проведення аналізу та вибору відповідного інструментарію веброзробника для реалізації вебдодатку обраної тематики;
- реалізація продуманого UI/UX-дизайну, в тому числі, графічного дизайну та контенту окремих сторінок;
- реалізацію технології багатомовного інтерфейсу;
- застосування оптимальної технології обробки даних, які є результатом дій користувача при використанні вебдодатку за призначенням;
- прототипування та розроблення завершеного варіанту сучасного інтерактивного вебдодатку для цільової аудиторії;
- оцінювання економічності обраних методів, технологій.

*Об'єктом дослідження* кваліфікаційної роботи є вибір та комбінування спеціалізованих вебтехнологій і програмних середовищ для розробки інтерактивного вебдодатку з ефективними функціями.

*Предметом дослідження* є властивості і технічні аспекти застосування сучасних вебтехнологій та інтегрованих програмних середовищ.

*Методи досліджень*: інформаційно-пошуковий, аналітичний, спостереження, методи порівняльного оцінювання, використання штучного інтелекту для генерації графічних елементів, економічне оцінювання вартості програмного продукту.

*Інформаційну базу* кваліфікаційної роботи складають наукові статті, тематична література, статистичні дані аналітичних компаній, які розміщені на вебсайтах у вільному доступі, власні спостереження та фахові дослідження з тематики поставленої задачі.

*Елементи наукової новизни* полягають у здійсненні нетривіального підходу до вибору вебтехнологій і створення оригінального вебдодатку із заданими функціями та методами обробки даних, тестуванні ефективності різних моделей штучного інтелекту для створення унікальних зображень.

*Практична значущість отриманих результатів* полягає в досягненні оригінального поєднання відомих технологій розробки вебдодатків із демонстрацією ефективності особливих розширень, зокрема функцій обробки подій, раціональних комбінацій дизайну користувача, залучення штучного інтелекту до вирішення складних рутинних завдань.

*Апробація результатів* дослідження відбувалася шляхом оприлюднення доповідей на міжнародних наукових конференціях.

*Публікації.* За результатами проведеного дослідження опубліковано тези доповідей на конференціях: «Тенденції використання штучного інтелекту в області програмної інженерії», матеріали VIII Міжнародна науково-практичної конференції «Modern technologies in education, work and science», 20-22 жовтня, 2025 року, м. Краків, Польща; та під час щорічного XXII міждисциплінарного семінару «Студентські роботи за науковою тематикою кафедри інформаційних систем та технологій», 25 листопада 2025 року, м. Полтава.

*Структура та обсяг кваліфікаційної роботи.* Пояснювальна записка кваліфікаційної роботи складається зі вступу, 3 розділів, висновків, списку використаних джерел (52 найменування), додатків. Кваліфікаційна робота містить 8 таблиць, 29 рисунків, викладена на 63 сторінках.

# РОЗДІЛ 1

## ТЕОРЕТИЧНИЙ ОГЛЯД ТЕХНОЛОГІЙ РОЗРОБКИ І ЗАБЕЗПЕЧЕННЯ ФУНКЦІЙ ВЕБДОДАТКІВ

### 1.1 Стан сучасних вебтехнологій та прикладні аспекти їх застосування

Через постійний розвиток та ускладнення архітектури вебдодатків, вимоги до швидкості, функціоналу та дизайну зростають з кожним роком. Невідповідність сучасним тенденціям може серйозно вплинути на взаємодію з користувачем та продуктивність сайту [1]. Вважається, що вебдодатки мають бути доступні цілодобово з будь-якої точки світу та бути адаптованими для використання з будь-якого пристрою з будь-яким розміром екрану. Окрім цього, важливими складовими сучасного вебдодатку є гнучка і масштабована система безпеки та багатий досвід користувача, побудований на клієнті [2].

З моменту створення і до наших днів інтернет зазнав значних змін. Спершу були статичні HTML-сторінки (Web 1.0). На зміну Web 1.0 прийшов Web 2.0, а разом із ним динамічні сайти та змога взаємодіяти з користувачами, що дало великий простір для реалізації ідей розробників. Наразі на підході третя ітерація – Web 3.0, яка базуватиметься на децентралізації, використанні штучного інтелекту, одноранговій системі комунікацій та обміну інформацією з використанням технології blockchain [3].

Технології дійсно швидко змінюються: те що було популярним вчора, сьогодні вже може бути не актуальним. У певних тематичних сегментах втрачають актуальність багатосторінкові сайти, адже їхня швидкість та продуктивність часто не задовольняють сучасні потреби. Це пов'язано з тим, що на кожну дію користувача запитується нова сторінка з серверу, та відбувається повне перезавантаження на клієнтській частині. На зміну їм прийшли односторінкові сайти Single Page Applications (SPA), які отримують сторінку з сервера лише один раз, а всі інші операції виконуються завдяки

асинхронним AJAX та JavaScript скриптам [4]. Подібні зміни вимагають нових ідей та підходів до розробки. Наприклад, поява концепції «компонентів» у веброботці дала змогу створювати елементи, які можна повторно використовувати в різних частинах коду. Окрім цього, з'явилися різні бібліотеки для управління станом додатку (Redux, MobX). Обов'язковою стала можливість інтеграції вебзастосунків з різними платформами і сервісами (соціальні мережі, платіжні системи, API сторонніх сервісів).

Для створення real-time застосунків, таких як чати, було створено технологію websockets, яка дозволяє встановлювати постійне двостороннє з'єднання між користувачами, без необхідності постійно створювати нові конекти, як при використанні традиційних методів [5].

Саме через необхідність постійного створення нового інноваційного інструментарію для реалізації подібних ідей сформувалась та невпинно розвивається величезна система вебтехнологій: спеціалізовані мови програмування, фреймворки, бібліотеки, бази даних та системи керування, інтегровані середовища розробки тощо.

На сучасному етапі розвитку інтернету існує велике розмаїття вебсайтів, спрямованих на виконання різних завдань та задоволення різних потреб користувачів. Різні категорії вебсайтів вимагають різних підходів до створення. Згідно досліджень популярності вебсайтів різного призначення за даними [6] показано, що найпопулярнішими типами сайтів є чати, соціальні мережі, пошукові двигуни та всемогливий e-Commerce: інтернет-магазини, аукціони, банкінг.

Сайти можна класифікувати по-різному. Для дослідження саме в розрізі використовуваного інструментарію їх можна умовно поділити на категорії за такими показниками як: призначення, складність архітектури, перелік типових програмних рішень, використаних для розробки. Результати дослідження наведено в табл. 1.1.

Як видно з таблиці, кожна категорія вимагає відповідного підходу та інструментарію для успішної реалізації.

Таблиця 1.1 – Порівняльна характеристика вебсайтів за категоріями

Категорія вебсайтів	Опис	Характеристика	Типовий інструментарій
Сайти–лендінги	Односторінкові вебсайти створені з метою реклами і маркетингу. Користувачі потрапляють на такі сайти через посилання в електронному повідомленні або рекламу в соц. мережах.	Простий дизайн, чітка ідея, максимально концентрований контент.	HTML та CSS, або можна використати генератор сайтів: Wix, Jekyll, Hugo.
Багатосторінкові інформаційні сайти	Спрямовані на надання користувачам інформації різного характеру: новинні портали, сайти компаній, освітні ресурси ті інші.	Розгалужена структура, акцент на текстовому і мультимедійному контенті, можливість взаємодії з користувачем через зворотній зв'язок.	CMS (Content Management Systems), або ж фреймворки: Django, Ruby on Rails, Express.js.
Соціальні мережі	Платформи для взаємодії та обміну інформацією. Блоги фокусуються на індивідуальному контенті, соціальні мережі – на взаємодії, форуми – на обговореннях.	Складні за будовою сайти, наявна можливість публікації та обміну контентом, профілі користувачів, механізми коментування та «лайків».	Для створення платформ використовують технології, Facebook та Instagram фреймворк React.
e-Commerce	Інтернет–магазини, майданчики оголошень (по типу OLX), сайти зі знижками інтернет–аукціони, системи онлайн–платежів.	Каталог товарів з можливістю фільтрації та пошуку, корзина покупок та система онлайн–оплати, особисті кабінети.	Для сайтів такого типу зазвичай використовуються спеціалізовані CMS: Joomla, OpenCart.
Інтерфейси масштабних проєктів	Це можуть бути корпоративні портали, системи управління великими даними, системи управління проєктами, банківські сайти, пошукові системи.	Складна архітектура з багатьма взаємодіючими модулями, забезпечення безпеки та управління доступом, формування аналітичних звітів.	Для розробки подібних вебсайтів використовуються потужні фреймворки: Angular, Nest, Laravel, Spring.
Інтерфейси хмарних обчислювальних систем	Вебінтерфейси для взаємодії з хмарними системами, наприклад: консолі управління хмарними платформами, такими як AWS Management Console чи Google Cloud Console.	Управління ресурсами хмарної інфраструктури, моніторинг та налаштування хмарних послуг, взаємодія з різними сервісами хмарної платформи.	Для таких систем, аналогічно з інтерфейсами масштабних проєктів краще використовувати потужні фреймворки.

Важливою властивістю сучасних вебдодатків є адаптивність. Зараз популярним є підхід «Mobile First» і, наприклад, автори роботи [7] закликають завжди дотримуватися цього принципу. Цей принцип означає, що застосунки повинні розроблятися таким чином, щоб в першу чергу гарно виглядати на мобільних пристроях, а вже після цього на широких екранах.

Сайт, створений за принципом «Mobile First», має наступні переваги: високе ранжування, швидке завантаження, сприяння здійсненню швидкої покупки. До появи таких інструментів, як CSS та JavaScript, такий підхід, та й взагалі адаптивність вебсторінки була неможливою. На сьогодні це важливо: якщо подивитись на статистику інтернет-трафіку за 2024-2025 рр., то можна побачити, що більше, ніж 60% трафіку, за аналітичними даними Statcounter складають користувачі саме мобільних пристроїв (рис. 1.1) [8]. В Україні, ця цифра є меншою і складає за останні 5 років 38,98 %.

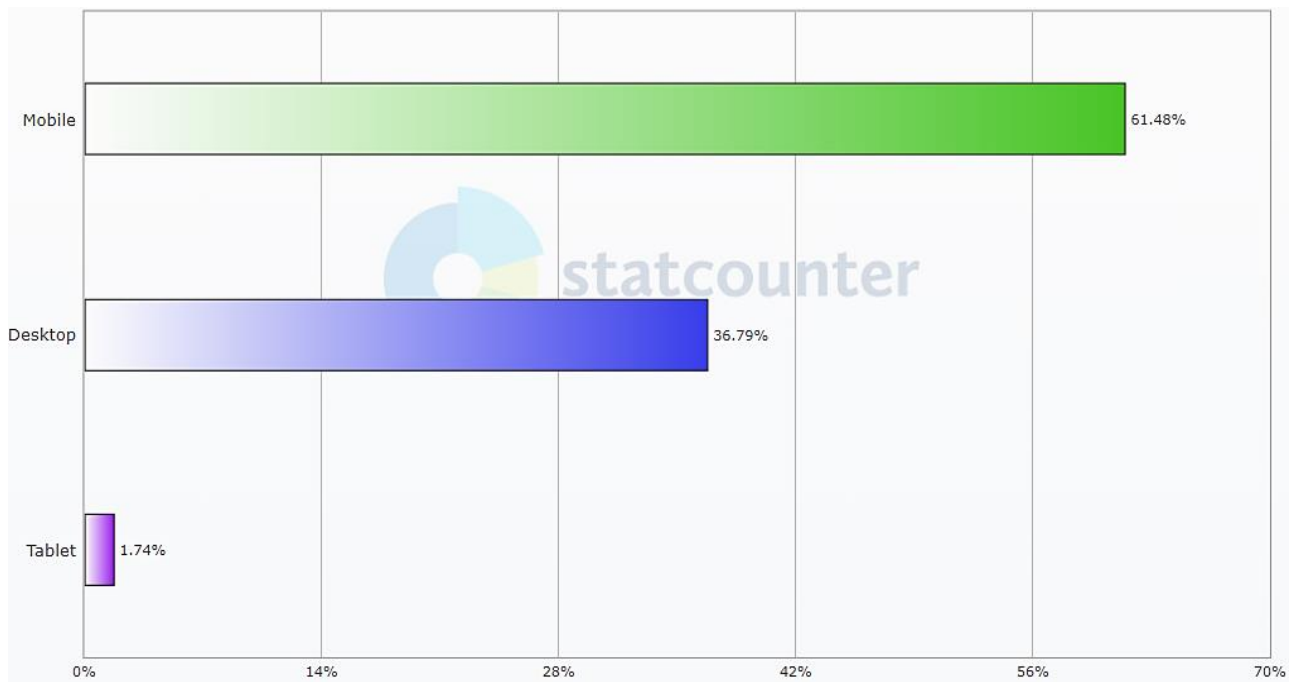


Рисунок 1.1 – Графік використання інтернет-трафіку користувачами мобільних пристроїв в світі станом на вересень 2025 рр. [8]

Крім того, розвиток мобільних технологій також призводить до необхідності впровадження в розробку специфічних фреймворків та

інструментів, таких як React Native, Flutter або Xamarin. За допомогою цих інструментів розробники можуть створювати кроссплатформені додатки, які працюють як в браузерах, так і на iOS та Android. Такий підхід дозволяє писати один код, який буде виконуватись на різних видах девайсів, що суттєво економить фінансові ресурси замовника та час розробника [9].

Враховуючи стрімкий розвиток галузі та постійну появу нових інноваційних інструментів, а також велику конкуренцію між вебстудіями, одним із головних завдань сучасного веброзробника є аналіз та підбір найкращого рішення саме для потреб його проєкту.

## **1.2 Особливості інструментів frontend- та backend-розробки для забезпечення функцій вебдодатків**

При розробці сучасного вебдодатку розрізняють два поняття як напрямки: frontend- та backend-розробка. Вони тісно пов'язані між собою, але це зовсім різні напрями програмування як по типу задач, що вони вирішують, так і по інструментарію, що використовується.

Під поняттям frontend мається на увазі видима для користувача (клієнтська) частина застосунку (не обов'язково сайту), а під поняттям backend – все те, що приховано від користувача (задній план, за екраном). Важливо розуміти, що для обох складових вебдодатку існує велика кількість різноманітних інструментів розробки, і однією із задач сучасного веброзробника є підбір найкращого набору інструментів саме для його потреб. Для цього потрібно провести аналіз наявних рішень.

Обов'язковим для реалізації frontend-частини сайту є лише один засіб – мова розмітки гіпертексту HTML. Все інше, включаючи CSS, JavaScript та різні бібліотеки, фреймворки або препроцесори не є обов'язковим з точки зору браузера, але в реальних умовах без подібних інструментів не створюється практично жоден вебдодаток у світі.

HTML (Hypertext Markup Language) – основа будь якого сайту, його каркас. За допомогою нього описується сама структура сайту, елементи які на ньому будуть знаходитися [10]. Препроцесори дозволяють розширити функціонал HTML при роботі з верстанням. Написаний за допомогою препроцесора код після інтерпретації сервером перетворюється в звичайний HTML. Прикладом таких препроцесорів може бути Pug або Haml.

До переваг використання препроцесорів HTML відноситься:

- простіша робота з великим об'ємом коду;
- дотримання принципу DRY (Don't Repeat Yourself);
- зменшення загального часу, необхідного на розробку, та спрощення організації коду [11].

Головним завданням HTML є розміщувати елементи на сторінці. Оскільки опис їх додаткових властивостей за допомогою лише атрибутів є досить трудомістким та нераціональним, то в свій час постала необхідність в інструменті, який би вказував браузеру, як ці об'єкти виглядають. Рішенням проблеми стали каскадні таблиці стилів CSS (Cascading Style Sheets).

Каскадні таблиці стилів (CSS) – це мова таблиць стилів, яка використовується для опису представлення документа, написаного на HTML або XML (включаючи діалекти XML, такі як SVG, MathML або XHTML). За допомогою сучасного CSS можна керувати всіма властивостями (розмір, колір, положення одних елементів відносно других, поведінку курсора при наведенні та багато інших) елементів вебдодатків. При цьому властивості використовуються каскадно, багаторазово, з варіативним набором селекторів. CSS є однією з основних мов відкритого Інтернету та стандартизована у веббраузерах відповідно до специфікацій W3C [12].

Для CSS, так само, як і для HTML, існують препроцесори, а також фреймворки. Найпопулярнішими препроцесорами для CSS є Stylus, SCSS, LESS. Загальна ідея CSS-фреймворків – допомогти побудувати гарно структурований вебсайт з можливістю подальшого розвитку, а також додати додаткові функції для покращення процесу розробки стилів. Фреймворки

дозволяють створювати сучасні вебінтерфейси, але також вони накладають свою специфіку та обмеження [13].

JavaScript – мова програмування, яка використовується веброзробниками для створення динамічних і інтерактивних елементів сайту. Існування JavaScript зробило можливим існування вебдодатків – застосунків, в яких оновлення інформації відбувається без оновлення всієї сторінки [14]. У JavaScript фактично немає конкурентів: 98% вебсайтів використовують саме цю мову. Але альтернативи все ж є: Dart, CoffeeScript, Naxe, ClojureScript. Вважати їх повноцінними аналогами не можна, адже в результаті всі вони все рівно компілюються в JavaScript код. Тому JavaScript зберігає лідируючі позиції протягом тривалого часу, при цьому змінюється, удосконалюється.

Використання стандартного JavaScript хоч і цілком можливе, але часто розробники обирають якусь готову бібліотеку, або фреймворк. Графік популярності базований на кількості питань, що стосуються відповідного фреймворку на Stack Overflow наведено на рис. 1.2.

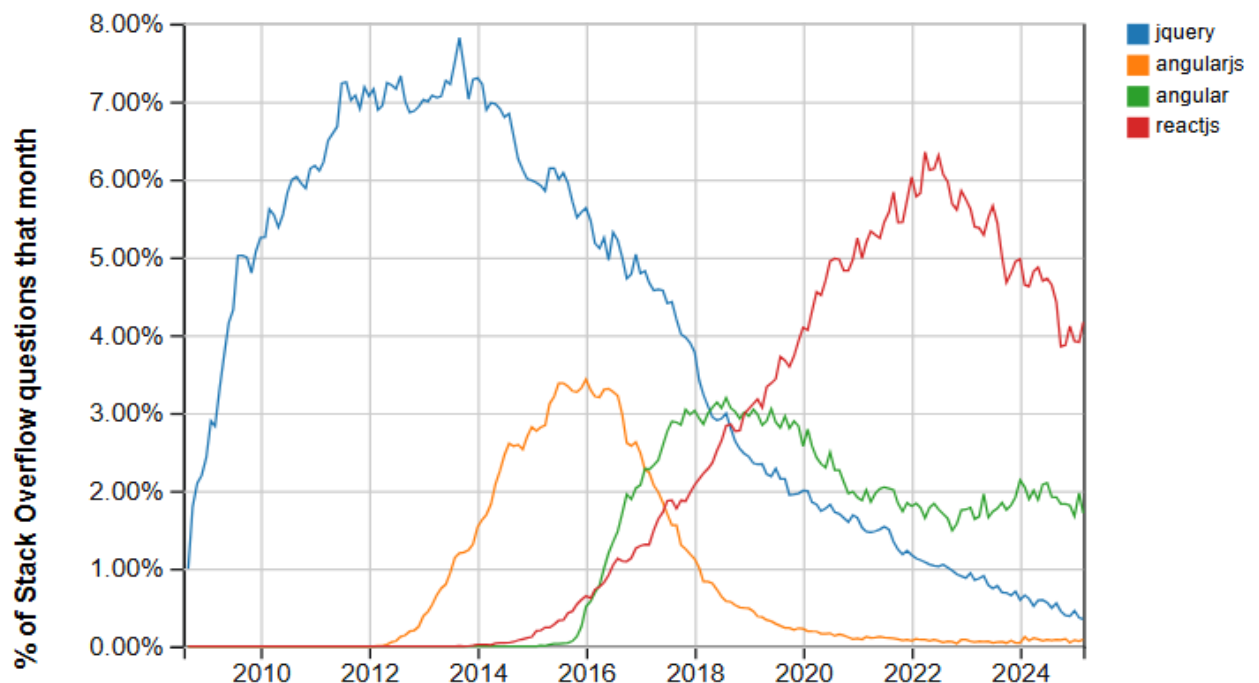


Рисунок 1.2 – Графік популярності фреймворків на Stack Overflow станом на кінець 2024 р. [15]

На графіку (див. рис. 1.2) видно, що найпопулярнішим фреймворком з великим відривом від інших є React, хоча за останній рік він дещо погіршив свої показники. JQuery, який був неймовірно популярним в 2012–2018 рр., з тих пір втратив значну частину користувачів, але в нього все ще залишається стабільна база розробників, які надають перевагу саме цьому фреймворку.

Фреймворк можна розглядати як певний набір інструментів, стандартів, підходів до написання коду. Саме він керує розробником та тим, як буде написаний код. Фреймворк обмежує розробника, не даючи вийти за свої рамки. Додатки, написані на одному фреймворкові, будуть мати майже однакову структуру файлів, ідентичні патерни розробки та підхід до вирішення задач. Окрім цього, більшість сучасних фреймворків мають великий арсенал вбудованого функціоналу, що вберігає розробників від «винайдення велосипедів», а також надає розробникам можливість сфокусуватися на більш важливих аспектах розробки, наприклад на бізнес-логіці.

Окремо варто розглядати TypeScript – мову програмування, що розширює JavaScript шляхом надання можливості статичної типізації. Використання TypeScript зазвичай не є обов’язковим (за винятком деяких фреймворків, наприклад – Angular), але його використання дуже бажано, особливо в проектах з великою кількістю класів, функцій та змінних. Розробнику, який використовує TypeScript не треба буде витратити час на те щоб дізнатися, які поля є у об’єкта, або що повертає та чи інша функція. В цьому йому допоможе IDE, яка повністю підтримує TypeScript, та буде показувати розробнику всю необхідну інформацію своєю функцією autocomplete (яка дуже погано працює з звичайним JavaScript) [16].

Клієнт спілкується з серверною частиною за допомогою HTTP-запитів. В цих запитах міститься посилання на кінцеву точку (URL-адреса на сервері, яка може приймати запити), тип запиту, та, за необхідності, додаткова корисна інформація – payload. Графічне зображення класичної взаємодії клієнту (frontend) та серверу (backend) наведено на рис. 1.3.

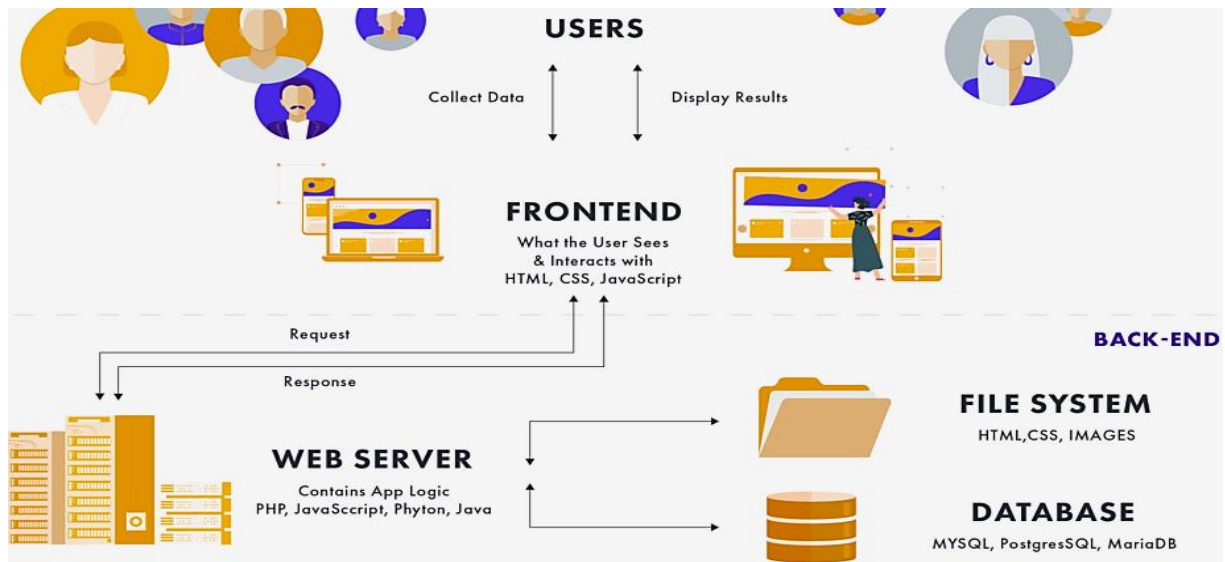


Рисунок 1.3 – Взаємодія frontend- і backend-частин вебдодатку

Після обробки запиту сервер повертає відповідь клієнту, яка містить в собі інформацію, що передбачена розробником. Сервер (тобто backend частина вебдодатку) повинен виконувати наступні функції: обробка HTTP-запитів, обробка бізнес-логіки, взаємодія з базою даних, авторизація та аутентифікація, логування. Вебсервер зберігає та надає доступ до контенту вебсайту – зображень, тексту, відео

Хоча вебсервери, зазвичай, розміщують вебсайти, доступні в інтернеті, вони також можуть використовуватися для зв'язку між вебклієнтами та серверами в локальних мережах, таких як Intranet компанії. Вебсервери також можуть обмежувати швидкість відповіді окремим клієнтам, щоб не допустити перевантаження ресурсів та задовільнити запити більшості користувачів [17].

Прикладами таких програмних засобів є Apache та Nginx. Результати дослідження [18] показали, що в більшості тестувань Apache виявився швидшим за Nginx, з чого можна зробити висновок, що Apache краще підходить для вебсайтів з великою кількістю запитів в хвилину.

Backend-частину вебсайту зазвичай пишуть на таких мовах програмування як Java, Python, Ruby, Node.js, PHP. Проте, зараз майже ніхто не використовує мову програмування саму по собі. Переважна більшість backend-розробників в своїй роботі використовують фреймворки.

Розробники можуть використовувати ту саму мову і синтаксис для всього стека програми. Це спрощує навчання та дозволяє розробникам бути більш універсальними. Деякі частини коду можуть бути легко перевикористані між клієнтом і сервером. Це особливо корисно для валідації даних, форматування та бізнес-логіки. JavaScript працює з JSON безпосередньо, що спрощує обмін даними між клієнтом та сервером.

Це може забезпечити кращу продуктивність для певних типів програм порівняно з PHP. npm (менеджер пакетів для Node.js) надає величезну кількість бібліотек та інструментів. Багато цих інструментів можуть використовуватися як на клієнті, так і на сервері. Можливість створювати програми, які можуть рендеруватися як на сервері, так і на клієнті, покращуючи продуктивність та SEO. Node.js, відмінно підходить для додатків реального часу завдяки вбудованій підтримці WebSockets або використовуємо сокет.іо. Інструменти для розробки, тестування та налагодження можуть бути спільними для клієнтської та серверної частин.

Найпопулярніші backend-фреймворки станом на 2024 р. наведено на рис. 1.4.

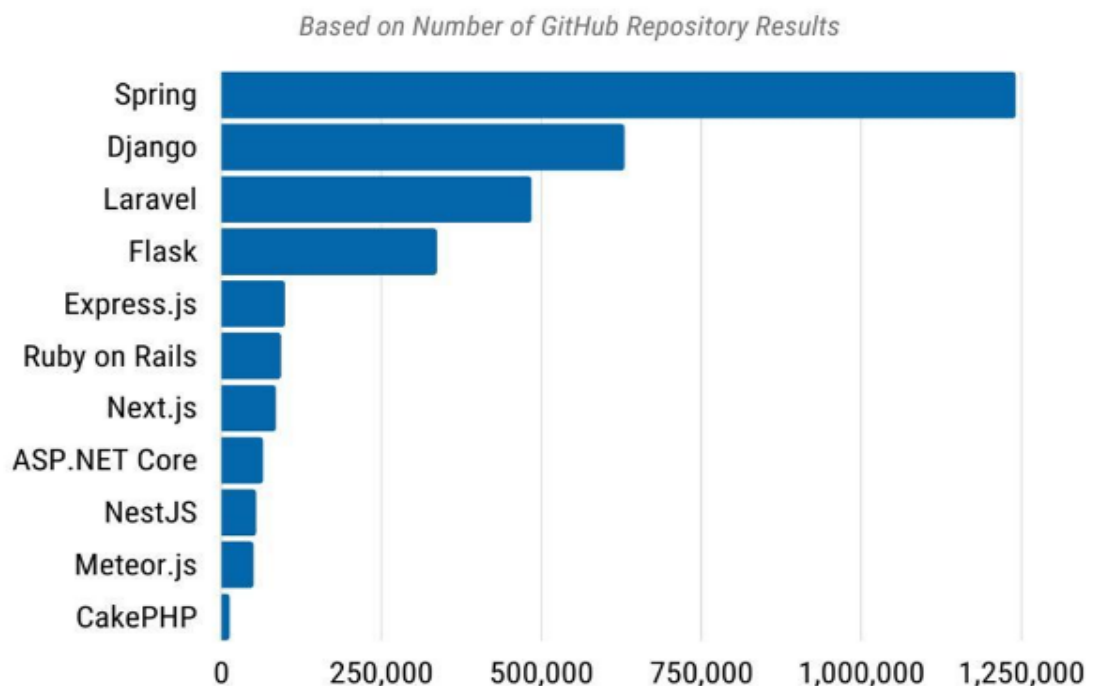


Рисунок 1.4 – Діаграма популярності backend-фреймворків [19]

На діаграмі видно, що до першої п'ятірки популярними фреймворками є: Spring (Java), Django (Python), Laravel (PHP), Flask та Express.JS (Node.js).

Окрім мов програмування, важливою складовою серверу є бази даних. База даних (БД) – це організована структура, призначена для зберігання, зміни й обробки взаємопов'язаної інформації. Саме в них зберігається вся інформація, пов'язана з роботою застосунку. Бази даних бувають: централізовані, хмарні, реляційні, нереляційні, об'єктно-орієнтовані, операційні та розподілені. Найпопулярнішими є реляційні, нереляційні (NoSQL) та об'єктно-орієнтовані БД.

В реляційних БД інформація зберігається у вигляді таблиць (відношень) пов'язаних між собою. Реляційна БД передбачає наявність ключів. Первинний ключ використовується для встановлення унікального ідентифікатору запису, а зовнішній ключ для встановлення зв'язків між таблицями. Головна перевага реляційних баз – ACID (Atomicity, Consistency, Isolation, Durability). Цей принцип гарантує надійність та цілісність даних в процесі їх обробки.

В нереляційних БД можливо сховище виду «ключ–значення», документоорієнтоване сховище, графові бази та bigtable–подібні бази. В таких базах немає чітких зв'язків між даними та немає чіткої структури. В NoSQL використовується підхід CAP (Consistency, Availability, Partition Tolerance). Перевагою NoSQL є швидкість та продуктивність [20].

В об'єктно-орієнтованих БД інформація представлена у вигляді об'єктів, як в об'єктно-орієнтованих мовах програмування. Візуально представляється у вигляді дерева, вузлами якого є об'єкти. Для опису властивостей (полів) об'єкту використовуються стандартні та конструйовані користувачем типи. Перевагою таких баз є чітка типізація та відсутність неспівпадіння моделі даних в застосунку та БД.

Для звернення до реляційних та об'єктно-орієнтованих баз даних використовуються SQL-запити – спеціальні конструкції написані за допомогою мови SQL. При використанні NoSQL баз форма запитів відрізняється в залежності від обраної БД, наприклад в MongoDB

використовується мова запитів, подібна до JavaScript – MongoDB Query Language.

Отже, реляційні БД ідеально підходять для роботи з даними, структура яких не вимагає частих змін, нереляційні – для зберігання великих об'ємів неструктурованої інформації, а об'єктно-орієнтовані БД – для проєктів, де потрібна потужна база з високою функціональністю.

Для управління базами даних використовуються системи управління базами даних (СУБД). Загальне призначення СУБД полягає в наданні централізованої системи для зручного та ефективного управління даними. Перевагами використання СУБД є: багатокористувацький доступ до БД, надійність даних, можливість створення бекапів. До недоліків відноситься: комплексність, вартість, вразливість БД, часті оновлення [21]. За видами СУБД поділяються на реляційні (RDBMS), документові (DoDBMS), стовпчасті (CDBMS). Окрім цього існують також СУБД, які спеціалізуються на різних типах сховищ, наприклад системи управління базами даних у пам'яті (IMDBMS), а також системи управління хмарними базами даних, де постачальник SaaS (Software as a Service) відповідає за обслуговування БД, наприклад MongoDB Atlas.

Отже, є мови програмування, є бази даних, системи управління базами даних, але це не все. Наприклад, серверну частину потрібно тестувати. Для вирішення таких питань існує сучасне рішення – інструменти тестування та налагодження API, або REST-клієнти. Також, значна увага приділяється керуванню контентом.

### **1.3 Особливості систем керування контентом CMS**

Система керування контентом (Content Management Systems, CMS) – це програмне забезпечення, яке надає певний рівень автоматизації для задач ефективного управління контентом. Зазвичай CMS являє собою

багатокористувацьке ПЗ на основі сервера, яке дозволяє взаємодіяти з даними в репозитарії. CMS дозволяють редакторам створювати новий контент, редагувати існуючий і робити його доступним для інших людей. При цьому їм не обов'язково розбиратися в програмному коді [22]. Для використання CMS не потрібно встановлювати спеціальне програмне забезпечення. Для адміністрування та редагування використовується звичайний браузер (Google Chrome або аналогічний). Інтуїтивний інтерфейс і простота роботи з системою полегшує управління сайтом і знижує подальші витрати на підтримку вебресурсу.

Центральний елемент системи – ядро CMS, відіграє ключову роль у виконанні основних функцій. Його завдання включає обробку запитів, керування базою даних та забезпечення основних механізмів безпеки. Це є фундаментальною складовою, що забезпечує стабільну та роботу CMS.

Система управління контентом нероздільно пов'язана із базою даних, яка виконує роль сховища для структурованих даних. Зазвичай використовується реляційна база даних для ефективного зберігання інформації, що дозволяє забезпечити швидкий та організований доступ до великої кількості даних. Це особливо актуально, якщо враховувати, що одним з основних видів вебсайтів, на якому використовується CMS, є інтернет-магазини, які вимагають особливо чіткої структуризації даних на сервері, а також потребують швидкої реакції на дії користувача.

Окрім стандартного вбудованого функціоналу, CMS зазвичай дозволяють використовувати плагіни для вирішення специфічних задач. Наприклад готова корзина для електронного магазину, або система онлайн-оплати покупок [23].

За напрямками виконуваних задач CMS можна розділити таким чином:

- системи керування вебконтентом (WCM) – тип системи, що розрахований на масову доставку контенту через вебсайт. Такі CMS чудово справляються з відокремленням вмісту від зовнішнього вигляду та публікаціями на декількох каналах;

– системи керування корпоративним вмістом (ЕСМ) – призначені для управління контентом, який не розрахований на масового користувача. Такі системи можуть використовуватись для керування робочими документами: резюме, звіти, службові записки;

– системи керування цифровими активами (DAM) – системи спрямовані на менеджмент та керівництво цифровими активами, як-от зображення, аудіо, відео та інші. DAM добре підходить для метаданих;

– системи керування записами (RMS) – системи для управління інформацією про операції та записами, створеними як побічний продукт бізнес-операцій (продажі, контракти тощо). RMS чудово справляється з контролем доступу.

Зважаючи на графік популярності CMS (рис. 1.5), WordPress є лідером з великим відривом (43.2%) і відмінно підходить для блогів та корпоративних вебсайтів завдяки легкості використання та розширюваності. Shopify ідеально підходить для онлайн торгівлі.

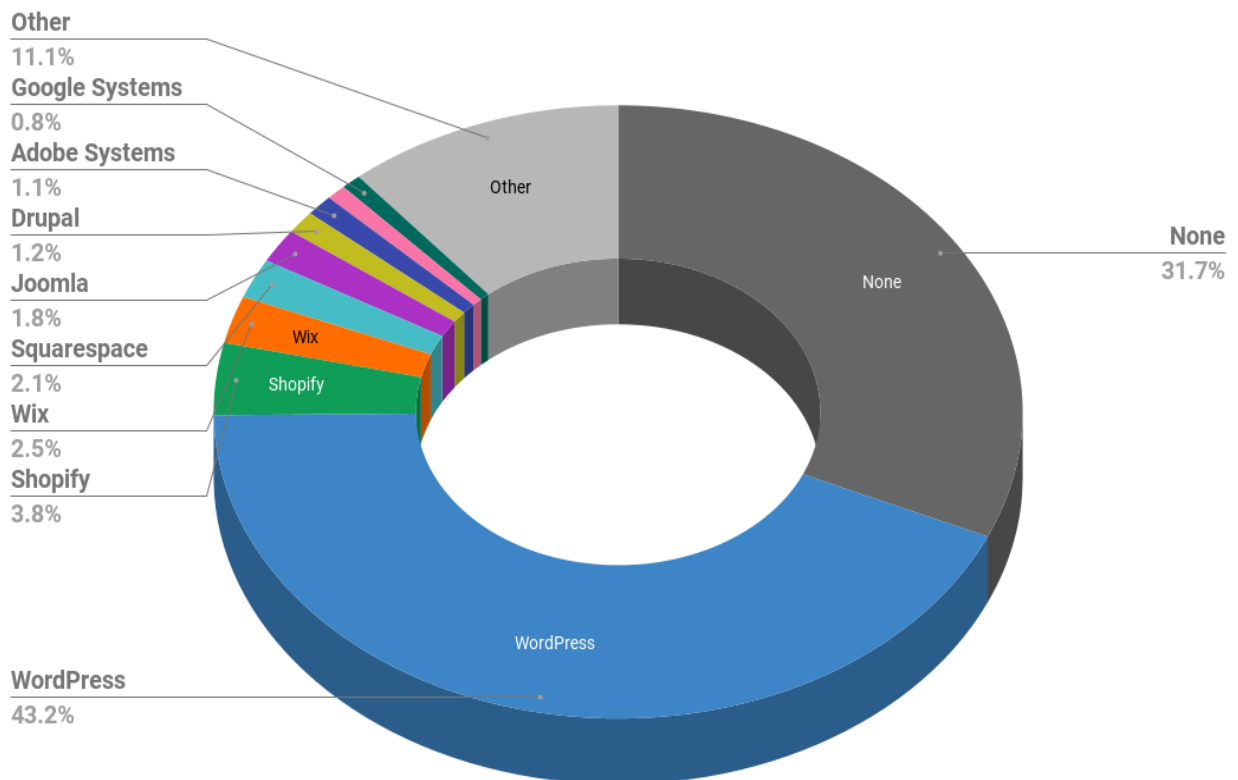


Рисунок 1.5 – Рейтинг популярності CMS [24]

Joomla використовуються для корпоративних та соціальних мереж, тоді як Drupal варто розглядати для складних великих проєктів, орієнтованих на гнучкість та безпеку.

Гарними прикладами вдалого використання CMS є особисті блоги, а також невеликі e-Commerce проєкти. Використовувати системи управління контентом не варто в тих випадках, коли потрібен специфічний функціонал, або є необхідність в його постійному розширенні. Також варто уникати CMS, якщо йде мова про розробку масштабних проєктів, для яких важлива швидкодія та високий рівень безпеки. Найбільш суттєві властивості різних CMS узагальнені в табл. 1.2.

Таблиця 1.2 – Узагальнення сильних і слабких сторін різних CMS

Переваги	Недоліки
<ul style="list-style-type: none"> <li>– великий вибір CMS;</li> <li>– висока швидкість розробки і дешевизна;</li> <li>– не потрібно вміти писати код;</li> <li>– простий і зрозумілий інтерфейс;</li> <li>– підтримка одночасної роботи багатьох користувачів;</li> <li>– велика кількість безкоштовних готових шаблонів;</li> <li>– можливість кастомізації;</li> <li>– гарне тимчасове рішення, поки повноцінний сайт в розробці;</li> <li>– більшість CMS мають модульну архітектуру, а їх функціональність легко розширюється за допомогою плагинів;</li> <li>– навіть безкоштовні CMS мають «технічну підтримку» у вигляді спільноти користувачів.</li> </ul>	<ul style="list-style-type: none"> <li>– низька адаптивність (flexabilty);</li> <li>– чим специфічніше завдання, тим менше шанс, що CMS підійде для цього;</li> <li>– потрібен час на засвоєння конкретної CMS;</li> <li>– для простих сайтів функціональність CMS, як правило, виявляється надмірною;</li> <li>– може зламатися, якщо адміністратор неправильно його налаштував;</li> <li>– у багатьох CMS є проблеми з безпекою.</li> </ul>

Отже, можна зробити висновок, що CMS є чудовими інструментами, якщо їх використовувати в правильному руслі. Якщо вебдодаток має значний обсяг контенту, який доволі часто потрібно оновлювати, то саме без CMS-рішень обійтися важко. Трудомісткі задачі рекомендується покладати на автоматизовані системи. Більше половини всіх активних вебсайтів використовують CMS, які ще називають «двигуном» сайту.

## Висновки до розділу 1

Поглиблений аналіз сучасного стану та тенденцій розвитку вебтехнологій дозволяє обґрунтувати низку тверджень. Перш за все, наразі відбувається стрімкий розвиток вимог та інструментів для розробки вебдодатків, які є невід'ємною частиною існування сучасного бізнесу, інформаційної підтримки всіх без виключення напрямків суспільно-економічного життя.

Увесь інструментарій поділяється на два суттєво різних напрямки – frontend і backend розробки, тобто дизайн інтерфейсу користувача та обробка даних, робота з базами даних на боці сервера. Відповідно й вебтехнології забезпечують вирішення специфікації цих завдань.

Однією з головних тенденцій є раціоналізація використання коду шляхом застосування фреймворків та бібліотек, препроцесорів, іноді CMS. Першою і обов'язковою технологією є мова розмітки HTML, який інтегрується з іншими мовами програмування.

Проведений аналіз особливостей різних вебсайтів за характеристиками, призначенням та оптимальним стеком технологій показав, що не існує універсального інструменту для кожної задачі. Було розглянуто особливості використання різних інструментів сучасного веброзробника. Додатково представлені різні узагальнені статистичні показники у вигляді графіків, діаграм популярності та порівняльні таблиці відповідних технологій, на які можна опиратися під час вибору інструментарію.

## РОЗДІЛ 2

### ОБҐРУНТУВАННЯ СТЕКУ ТЕХНОЛОГІЙ ПРОЄКТУВАННЯ ВЕБДОДАТКІВ ІЗ ЗАДАНИМ ФУНКЦІОНАЛОМ

#### **2.1 Аналіз вимог до функцій та структури інтерактивного інформаційного вебдодатку**

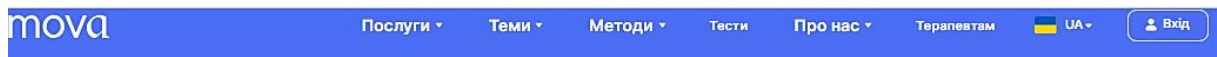
Практичною частиною кваліфікаційної роботи є застосування обраних технологій і засобів до розробки інтерактивного вебдодатку, який забезпечує взаємодію користувача зі спеціально підібраним контентом, здійснює обробку даних та має відповідний дизайн. Для виконання завдань була реалізована оригінальна ідея – розробити вебдодаток зі спеціальним функціоналом у вигляді квест-тесту із соціально-психологічним контентом. Взаємодія відбувається так, що користувач послідовно відповідає на питання, з головної сторінки потрапляє на нові сторінки в залежності від обраного варіанту відповіді, і в кінці отримує свій психологічний портрет. Питання плануються такі, що пов'язані з поведінкою в складних непередбачуваних ситуаціях. На кожній сторінці описана сцена і є три варіанти відповіді. Користувач обирає відповідь і потрапляє на наступний етап. В кінці отримує свій поведінковий психологічний портрет. При цьому передбачено кнопки вибору двох мов спілкування – опитування буде проводитися українською або англійською мовами. Цільова аудиторія не має вікових або інших обмежень, контент забезпечує виявлення варіантів особистісних характеристик користувача шляхом продуманої навігації по тематичним розділам додатку. Таким чином, основний задум – інтерактивний психологічний тест-квест із розгалуженим сюжетом та спеціально підібраними ілюстраціями.

Для реалізації поставленої цілі необхідно вирішити наступні задачі:

- вивчити особливості вебсайтів, на яких реалізовано подача текстової інформації соціально-психологічного характеру, елементи проходження тестів та обробка даних і збереження результатів;

- провести аналіз найбільш раціональних інструментальних рішень для реалізації основних вимог, трендів та в результаті – розроблення технічного завдання;
- визначити детальну структуру майбутнього вебдодатку;
- розробити прототип та протестувати для виявлення складних пунктів реалізації;
- підібрати або створити власний сучасний графічний матеріал за тематикою розділів вебдодатку;
- реалізувати вебдодаток із використанням сучасних технологій, використовуючи спеціальні середовища розробки та контролю версій, провести тестування.

Аналіз подібних застосунків в мережі показав, що таких розробок є недостатня кількість у порівнянні з суспільними потребами, дизайн часто не відповідає завданням сайту, дисонує, створює протилежний настрій. Приклад інтерфейсу додатку з багатьма тестовими темами [25] показано на рис. 2.1, інтерфейс початкової сторінки одного з тестів – на рис. 2.2. та в додатку А.



## Тести

Кожен день ми реалізуємо свої цілі та плани, працюємо, навчаємося і займаємося волонтерством. Ці справи потребують як фізичних, так і психологічних ресурсів. Щоб мати енергію, яку можна передати іншим, спочатку ми повинні піклуватися про себе.

Ми пропонуємо для вас тести, аби перевірити свої знання про психологію та порефлексувати над власним емоційним станом та ресурсом, які дають нам витривалість та силу для досягнення маленьких перемог.

Підкреслимо, що попри медичне підґрунтя деяких тестів, їх не можна використовувати для самодіагностування. Якщо у вас є потреба та запит, ви можете звернутися до психотерапевтів та психіатрів на нашій платформі.



Рисунок 2.1 – Дизайн інтерфейсу одного із сайтів для психологічних тестів

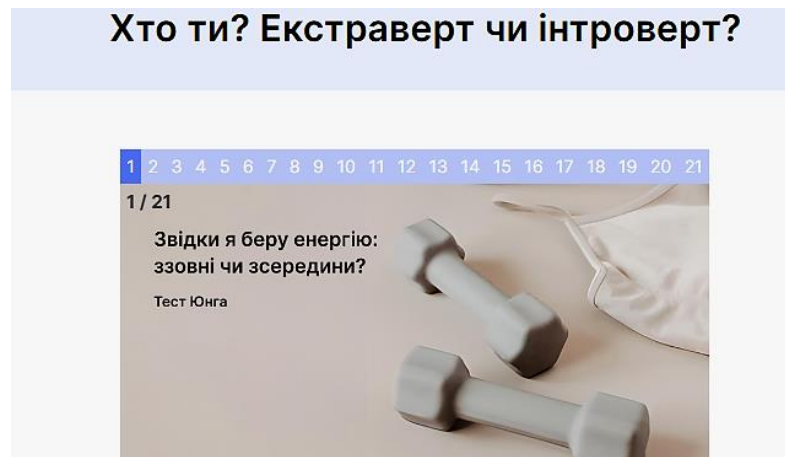


Рисунок 2.2 – Інтерфейс сторінки для початку проходження тесту [25]

Пробне використання запропонованих тестів на кількох задовільних за різними критеріями вебсайтів показали системні недоліки (дизайн, взаємодія користувача, структура та ін.) таких продуктів, наприклад, [25-27], та дозволили виробити систему вимог для досягнення кращого результату. Типові недоліки технічного та дизайнерського плану зведено в табл. 2.1.

Таблиця 2.1 – Узагальнення аналізу сайтів із психологічними тестами

Назва елементу/ частини вебдодатку	Наявні рішення з позитивною оцінкою	Сучасні вимоги і план покращення
Меню	Меню з багатьох пунктів з переходом на тематичні сторінки, розгортається списком	Меню навігації закріплюється, адаптивне, гамбургер-меню
Мовний інтерфейс	Часто одна мова або Укр/Рос	Текст і тести реалізується українською та європейськими мовами через вибір із списку
Присутність графіки в дизайні сторінок	Великі сторінки, наявні зображення, текстова змістовна інформація, галереї, партнери	Зображення ретельно підібрані, авторські, інтерактивні, наявні динамічні ефекти, продумана кольорова гама з настроєм, дотримання кількості кольорів та акцентів
Статистичні дані	Наявність певної статистики, наприклад відвідуваність, консультації, успішні історії	Статистика оновлюється завдяки функціям JavaScript. Розташування після основної пропозиції
Стрічка відгуків	Присутня, якщо є послуги	Організувати карусель або інший динамічний елемент
Інформація про платні послуги	Послуги частково платні	Резервування можливості демоверсії, спробувати безкоштовно, благодійні сторінки і послуги

Окрім технологічного аналізу, зрозуміло, що є високий соціальний запит на отримання доступу до різноманітних тестів, служб допомоги психологічного характеру, які допомагають розкрити низку проблем та зорієнтувати у світі викликів. Для різних вікових груп бажано знайти рішення.

Дизайн такого сайту повинен бути ергономічним та враховувати визначальні потреби користувачів. Критичну роль у забезпеченні задоволення користувачів відіграють: чітка структура навігації, мінімалістичний та адаптивний дизайн, інтерактивні елементи взаємодії, підтримка високоякісного мультимедійного контенту. Наперед було враховано, що вебдодаток буде багатосторінковим, оскільки планується пройти тест за більше, ніж 5 рівнями. Частинами технічного завдання були вимоги до вебдодатку як технологічного продукту та дизайну.

На початку сформульовані вимоги до стилістики вебдодатку:

- світлі або м'які пастельні кольори;
- великі шрифти, простір між варіантами;
- плавні анімації (наприклад, fade-in сцени);
- прогрес-бар вгорі (не обов'язково відсотки, можна кроки: ● ○ ○ ○ ○).

Основні UX/UI ідеї, які заплановано реалізувати:

- прогрес-бар для показу, скільки вже рівнів пройдено;
- можливість повернутися на попереднє питання;
- адаптивний дизайн для мобільних пристроїв.

Наступним етапом є розроблення загальної структури додатку, яка представлена далі у вигляді списку.

1. Екран вітання (Welcome screen):

- опис призначення вебдодатку, цілі участі в запропонованому тест-квесті та перспективі ознайомлення з кінцевим результатом;
- опція вибору мови: UA / EN;
- кнопка «Почати».

2. Початкове запитання (впливає або на адаптацію сцен, або просто як частина профілю):

- вибір шляху опитування згідно гендеру.

### 3. Основна частина – інтерфейс опитування (на кожній сторінці):

- опис сцени (реалістична ситуація, наприклад: «У метро стається раптова зупинка, і люди починають панікувати...» або інша тема).

- 3 варіанти дій (наприклад: заспокоювати інших, вийти з вагона, сховатися в кутку або інші варіанти за темою питання).

- відповідь веде на іншу сторінку з новою сценою – це вже дерево сценаріїв (branching logic).

### 4. Фінальний екран – психологічний портрет:

- текстовий опис поведінкових особливостей (наприклад: «Ви – стратег. У непередбачуваних ситуаціях спершу аналізуєте, а потім дієте...» або інший опис). Можна додати:

- збереження результату (PDF, email, скріншот).

- поділитися в соцмережах.

### 5. В нижній частині рекомендується подати інформацію про розробника, авторське право, контакти.

Структурна схема сайту містить інформацію про зв'язки та кількість сторінок. Для реалізації поставлених задач необхідно обрати інструментарій розробника, який би дозволив:

- забезпечити зручну і раціональну реалізацію логіки у відповідному коді за допомогою сучасного інструментарію;

- обрати середовище розробки коду у відповідності до обраних засобів;

- реалізувати мультимовну підтримку;

- забезпечити функціонування логіки питань з варіантами вибору й переходу на новий рівень в залежності від цього вибору;

- обробку даних для збереження або очищення;

- джерело графічних зображень, які б підкреслили унікальність контенту вебдодатку.

## 2.2 Визначення інструментарію проєктування та розробка прототипу вебдодатку

Для виконання практичної частини роботи спершу необхідно визначитися з інструментарієм. Серед базових інструментів на основі аналізу, представленого в розділі 1, розглядалися класичні технології HTML&CSS, мова JavaScript, різноманітні фреймворки frontend-розробки, зокрема React або інші, окремі технології backend-розробки.

При виборі фреймворків були використані основні правила, які були вибрані шляхом вивчення досвіду практикуючих вебдизайнерів, зокрема [28]. Сутність критеріїв розкрито в наступних пунктах.

1. Якісна документація. Найчастіше навіть створений з нуля код може здаватися складним для розуміння, якщо його залишити і повернутися через якийсь час. Застосовуючи фреймворки робота проводиться з чужим кодом, тому наявність документації та навчальних матеріалів допоможуть використовувати можливості в повній мірі.

2. Ліцензія. У деяких платформ є суворі вимоги, ліцензія, що може викликати обмеження застосування чи поширення. Фреймворк потрібно підбирати так, щоб його ліцензія відповідала завданням проєкту.

3. Функціональність. Вибирати фреймворк потрібно під параметри конкретного проєкту, а не лише за рейтингами чи популярністю. У деяких випадках не потрібні варіанти з повним стеком, якщо для ПЗ потрібна функція маршрутизації.

4. Налаштування. Для розробника важливо мати можливість змінювати функціональні особливості, які зможуть відповідати унікальним завданням проєкту. Для цього фреймворк має гарантувати гнучкість.

5. Підтримка. Вибраний варіант повинен мати гарне ком'юніті для підтримки. Відразу після початку роботи з фреймворком це буде обов'язковою частиною програмного забезпечення. Без спільноти потрібно докласти набагато більше зусиль, іноді й повністю переписати код.

6. Бізнес-чинники. Обов'язково до уваги беруться потреби самого бізнесу, коли вибирається фреймворк. Для компанії, яка хоче бути партнером великої організації, потрібно орієнтуватися на структуру під більші фірми.

Розібравши всі тенденції та найкращі фреймворки на 2025 р. можна визначитися, який варіант буде відповідати та підходить під конкретний проект чи бізнес. Рендеринг на стороні сервера, швидке розгортання та SEO залишаються у пріоритеті.

Провівши аналіз, було прийнято рішення використати такі технології, як React та JSON. Вибір має чітке технічне і практичне обґрунтування (табл. 2.2), особливо для інтерактивного психологічного вебтесту.

Таблиця 2.2 – Обґрунтування вибору стеку технологій веброзробки

Потреба	Рішення	Обґрунтування
Інтерактивний інтерфейс	React	Динамічні переходи, без перезавантаження сторінки
Управління логікою і мовами	JSON-структура	Гнучка, розширювана, легко інтегрується
Мультимовність	JSON + i18next	Простий формат, масштабування під інші мови
Майбутнє розширення функцій	React-компоненти + бібліотеки	Легко додавати нові функції, анімації, аналітику

React – це фреймворк з відкритим кодом, створений компанією Facebook, яка використовується для створення користувацьких інтерфейсів. За допомогою React розробники можуть з легкістю створювати об'ємні вебдодатки, які здатні оновлювати інформацію на сторінці, без оновлення самої сторінки. Це реалізовано за допомогою Virtual DOM. Virtual DOM це концепція, в якій ідеальне, або «віртуальне» відображення UI зберігається в пам'яті і синхронізується з реальним за допомогою бібліотек–посередників. Коли в React інтерфейсі відбувається зміна, то відбувається re-rendering лише тієї частини DOM–дерева, яка зазнала змін [29]. Головною особливістю React є використання JSX – розширення синтаксису JavaScript. Це дозволяє писати HTML- структури всередині JavaScript файлу використовуючи синтаксис JSX.

На відміну від інших фреймворків, в яких зазвичай реалізується two-way data binding (як в Angular), в React реалізований однонаправлений потік даних. React має гарну зворотну сумісність: мало ймовірно, що після оновлення версії фреймворку в проєкті щось перестане працювати. React використовують для вебдодатків будь-якого розміру.

Особливості фреймворку React більш детально розкриті нижче.

1. Інтерактивність без перезавантаження сторінки:

- React дозволяє змінювати контент (сцени, питання, мову) динамічно, без оновлення сторінки;

- ідеально підходить для «step-by-step» логіки, що нам і потрібно в психологічному тесті.

2. Стан (State) і логіка переходів. `useState` та інші хуки дозволяють зручно зберігати:

- обрану мову,
- поточне питання,
- обрану відповідь.

Це простіше, ніж у класичних HTML+JS чи навіть у jQuery.

3. Компонентна структура:

- кожен блок (питання, опції, кнопки, перемикач мови) можна оформити як окремий компонент;

- спрощує масштабування (наприклад, у майбутньому додати таймер, чат або аналіз відповідей).

4. Можливість легко підключити бібліотеки, наприклад:

- `i18next` – для мультимовності;
- `Chart.js` або `Recharts` – для візуалізації результатів;
- `TailwindCSS` – для стилів.

Наступним кроком обґрунтовується, чому JSON обрано для мовної підтримки та логіки.

1. Гнучка структура даних, а саме:

- JSON легко читається і людиною, і машиною;

- його зручно редагувати вручну чи генерувати автоматично (наприклад, з Google Sheets);

- один і той самий файл може містити і логіку, і тексти – розділяти це не обов'язково.

## 2. Масштабування:

- щоб додати нову мову, просто додай нове поле: "fr": "...".

- щоб додати нову сцену, достатньо додати ще один об'єкт у JSON.

## 3. Локалізація:

- JSON-файли сумісні з i18next або будь-якою іншою бібліотекою для мультимовних інтерфейсів;

- доступ до тексту за ключем і мовою (scene[lang]) забезпечується швидко й ефективно.

На початку було здійснено прототипування сайту: розроблено спрощену функціональну модель майбутнього продукту [30], яка дозволила перевірити концепція та протестувати функціонал. Макет сторінки прототипу показано на рис. 2.3, з якого видно логіку побудови сцен, і перемикач мов інтерфейсу.

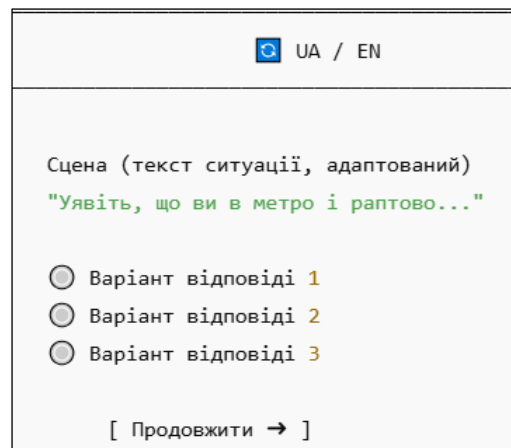


Рисунок 2.3 – Схема прототипу основної ідеї вебдодатку для однієї сцени

Згідно макету прототипу було необхідно розробити кілька скорочених прикладів з метою перевірки зручності та функціональності інструментарію:

- каркас у React (із прикладом однієї сцени, мультимовності і переходів);

- JSON-структуру для логіки питань;
- прототип дизайну в стилі «психологічного тесту».

На рис. 2.4 наведено код прикладу JSON-структури поведінки логіки питань при розробці прототипів окремих частин майбутнього вебдодатку.

```

1  {
2    "start": {
3      "id": "start",
4      "scene": {
5        "ua": "Уявіть, що ви опинилися в незнайомому місті без мобільного зв'язку...",
6        "en": "Imagine you are in an unfamiliar city with no mobile connection..."
7      },
8      "options": [
9        {
10         "text": {
11           "ua": "Шукаю місцевих ☹️ прошу допомоги",
12           "en": "Look for locals and ask for help"
13         },
14         "next": "scene_1"
15       },
16       {
17         "text": {
18           "ua": "Пробую знайти карту 🗺️ орієнтири",
19           "en": "Try to find a map or landmarks"
20         },
21         "next": "scene_2"
22       },
23       {
24         "text": {
25           "ua": "Паніка – не знаю, що робити",
26           "en": "Panic – no idea what to do"
27         },
28         "next": "scene_3"
29       }
30     ]
31   },
32   "scene_1": {
33     "id": "scene_1",
34     "scene": {
35       "ua": "Місцеві виявилися недружними та не бажають допомагати. Що далі?",
36       "en": "The locals seem unfriendly and unwilling to help. What next?"
37     },
38     "options": [
39       {
40         "text": {
41           "ua": "Відходжу та шукаю інші шляхи",
42           "en": "Step back and look for other options"
43         },
44         "next": "scene_4"
45       },
46       ...
47     ]
48   }
49 }

```

Рисунок 2.4 – Приклад JSON-структури логіки питань

Наведена структура дозволяє описати кожен окрему сцену, її текст можливо подавати двома мовами. Показано варіанти відповідей і до яких сцен

вони ведуть. Наступним кроком було створення коду прототипу на React цієї сцени на основі цього JSON (див. рис. 2.4) – з мовною підтримкою, кнопками, переходами й навіть прогресом.

Початковий код для психологічного тесту на React представлено в додатку Б. Інтерактивна візуалізація моделі тестування на різних мовах показана на рис. 2.5-2.6.

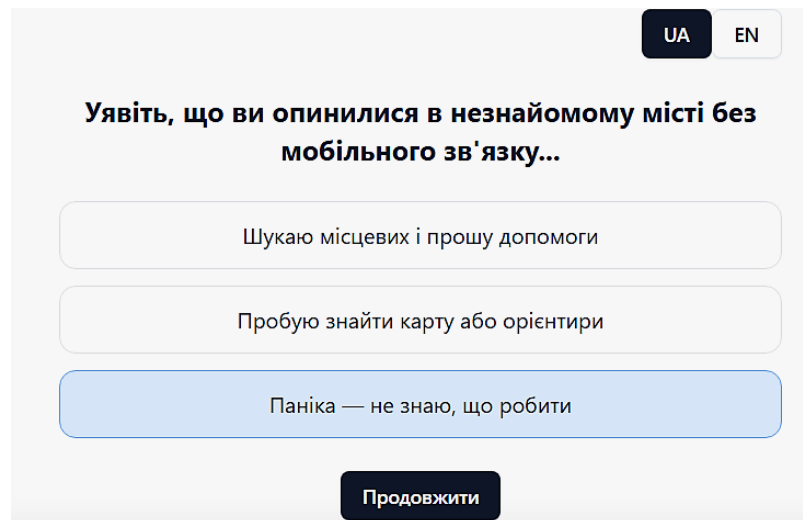


Рисунок 2.5 – Візуалізація сцени з вибором відповіді українською мовою

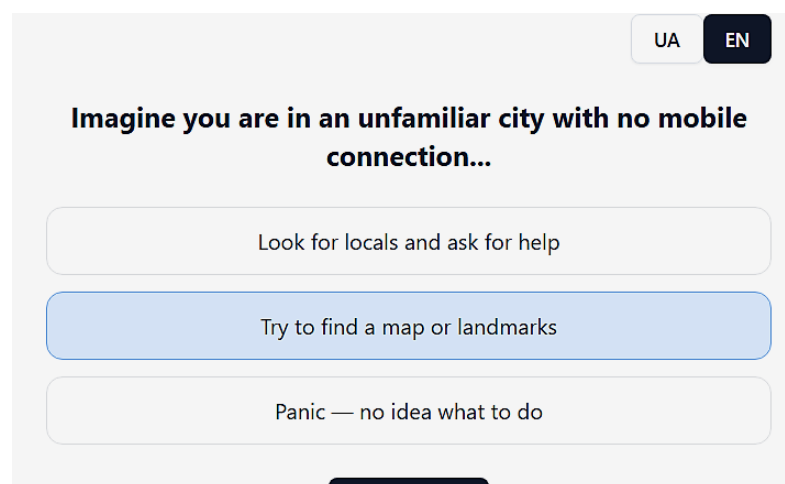


Рисунок 2.6 – Візуалізація в браузері прототипу сцени англійською мовою

У прототипі інтерфейсу експериментального вебдодатку реалізовано:

- вибір мови українська / англійська(UA/EN);
- відображення сцени та варіантів відповідей у вигляді кнопок;

- логіка переходу між питаннями на основі вибору;
- приклад із однією гілкою переходу;

Обраний комплекс технологій дозволяє не лише створити інтерактивний приклад на його основі, а також згодом додати більше сцен, прогрес-бар, фінальну оцінку (портрет) на основі виборів. Окрім React, знадобляться інші бібліотеки.

i18nextV4 – це бібліотеки JavaScript, що надають прості у використанні рішення для локалізації та інтернаціоналізації різних середовищ на основі JavaScript. Будучи однією з найстаріших бібліотек i18n, вона підтримує стандартні функції бібліотеки i18n, такі як інтерполяція та множина, і добре працює з асинхронними запитами [31]. При правильному налаштуванні вона визначає параметри мови браузера для автоматичного завантаження даних, специфічних для мовних стандартів.

Як і інші бібліотеки JavaScript, версії i18n, i18next та i18nextV4 використовують формат JSON для зберігання перекладів. Рядки, що очікують переведення, або поміщаються безпосередньо під значенням, або вкладаються всередину іншого об'єкта (наприклад, інтерпольовані значення). Під час перекладу слід переконатися, що всі змінні збережені. Далі слідує кілька правил, які необхідно враховувати.

1. Якщо ви використовуєте i18nextV4, виберіть формат i18next 4 (.json) під час завантаження.

2. Чат-боти штучного інтелекту можуть дуже ефективно генерувати список ключів з файлу .JSON.

3. У двох версіях множина обробляється по-різному. Щоб визначити ключі у множині, i18next використовує:

- для мов зі складними правилами множини, такими як українська, буде використовуватися `keyname_0`, `keyname_1`, `keyname_2` та `keyname_5`.
- для мов з простими правилами множини, такими як англійська, будуть використовуватися `keyname` та `keyname_plural`.
- i18nextV4 використовує закінчення `<plural_suffix>`;

– множина зі словами один, два, три або для простого реєстру `keyname_one` та `keyname_other`.

### 2.3 Порівняльний аналіз середовищ розробки та контролю версій

Окрім основного інструментарію вебтехнологій, перерахованого в попередніх підрозділах, існує велике різноманіття додаткових програмних засобів, спрямованих на полегшення роботи веброзробника. До таких інструментів можна віднести пакетні менеджери, системи контролю версій та репозитарії, редактори коду та інтегровані середовища розробки тощо. Для зручності та продуктивності роботи необхідно більш детально розібратися з кожним із видів інструментів за призначенням та унікальністю.

1. Пакетні менеджери. В загальному, призначення пакетних менеджерів полягає в керуванні, пошуку та встановленні пакетів залежностей, а також їх оновленні. Пакетами в Node.js називають один, або декілька JavaScript-файлів, що являють собою бібліотеку. Найпопулярнішими рішеннями на даний момент є `npm` та `yarn`. За своєю структурою вони дуже схожі і надають практично ідентичний функціонал.

Наразі всі менеджери пакетів отримують свої файли з реєстру пакетів NPM, тож будь-яка бібліотека, яка вам знадобиться, може бути встановлена обома рішеннями [32]. Розробники надають перевагу `npm`, оскільки він є вбудованим в Node.js і через це він часто є першим пакетним менеджером, з яким зіштовхується початківець.

При першому запуску проєкту, створюється ще один файл «`lock`», в якому знаходиться перелік всіх пакетів, які потрібно встановити [33]. Пакети в Node.js можуть займати суттєву кількість пам'яті і при використанні репозиторіїв, постійне завантаження встановлених пакетів є неоптимальним. Пакетні менеджери вирішують цю проблему. Достатньо скачати лише основні файли проєкту та файли «`package.json`» і «`lock`», після чого запустити команду

`npm install` або `yarn install`, в залежності від обраного менеджера. В результаті буде встановлено всі пакети відповідних версій.

Наразі використання пакетних менеджерів є фактично обов'язком при роботі з JavaScript та Node.js. Без них розробнику доведеться вручну шукати кожну бібліотеку та встановлювати її.

2. Системи контролю версій та репозиторії. Система контролю версій – це програмний засіб, який зберігає зміни в одному, або декількох файлах, та дозволяє в майбутньому повертатися до їх попередніх версій. Якщо ви пишете код, то використання таких систем є доцільним, оскільки вони дозволяють «відкатувати» зміни в випадку поломок, бачити хто і як змінював файли, та порівнювати зміни між різними версіями. Ці системи можна поділити на кілька категорій. Розглянемо детальніше кожну з них.

1. Локальні системи контролю версій. Якщо над проектом працює лише ви, то найпростішим рішенням буде використання локальних систем. Однією з найбільш популярних локальних систем є RCS. Вона зберігає набори відмінностей між файлами в спеціальному форматі на диску та дозволяє в будь-який момент відтворити файл. Мінусом таких систем є схильність до появи помилок. Легко переплутати файли і зробити не ті зміни, які хотіли [33].

2. Централізовані системи контролю версій. Важливим питанням є співпраця з іншими розробниками. Для того, щоб вирішити цю проблему було створено централізовані системи. Вони мають єдиний сервер, та певну кількість користувачів, які мають до нього доступ. Прикладами таких систем є: CVS, Subversion, Perforce.

До переваг можна віднести те, що кожному учаснику в тій чи іншій мірі відомо, чим займаються інші. Також адміністратор може надавати кожному учаснику доступ лише до тих файлів, з якими йому треба буде працювати. Недоліками таких систем є: відмова центрального серверу призводить до простою всієї команди розробників; при виходу з ладу жорсткого диску, ви втрачаєте весь проект, якщо у вас не було створено резервних копій [34].

3. Розподілені (децентралізовані) системи контролю версій реалізують найсучасніший підхід до контролю версій проєкту: учасники отримують не лише останній знімок файлів репозитарію, але повну копію та історію змін.

Таким чином, в кожного розробника зберігається повноцінна версія проєкту і в разі виходу з ладу серверу, з будь-якого локального репозитарію можна буде все відновити. Приклади таких систем: Git, Mercury, Bazaar, Darcs [35].

Враховуючи особливості різних видів систем, найкращим варіантом на сучасному етапі є саме розподілені системи контролю версій. На рис. 2.7 наведено рейтинг популярності їх використання. Видно, що лідером з величезним відривом є Git.



Рисунок 2.7 – Популярність систем контролю версій за [35]

Зазвичай сучасні проєкти зберігають на вебрепозитаріях, які повністю сумісні з Git. Такі сервіси дозволяють з легкістю керувати змінами в файлах проєкту. Їх основні функції полягають в запитах на зміну, сторонніх інтеграціях, клонуваннях репозитарію, ревью коду, відслідковування проблем. Прикладами таких сервісів є: GitHub, GitLab, Bitbucket.

Інтегровані середовища розробки (IDE). По-перше потрібно пояснити різницю між текстовими редакторами та IDE. Текстові редактори є легкими, а IDE – важкими, більш вимогливими до ресурсів, але в той же час надають значно більше корисних функцій. Прикладом текстового редактора саме для роботи з кодом може бути: Atom, Vim, Sublime Text, Notepad++. Такі редактори непогано підходять для невеликих проєктів, наприклад, лендінгів. Для більш серйозних проєктів бажано використовувати спеціалізовані IDE.

Вимогами до сучасних IDE є:

- підсвічування синтаксису мови програмування
- auto–complete коду
- можливість відладки коду;
- можливість встановлення плагінів;
- вбудована підтримка Git;
- підтримка препроцесорів [36].

Зазвичай, у кожній мові програмування є власні, спеціалізовані саме під неї IDE. Наприклад для Java це IntelliJ IDEA або Eclipse, для Python – PyCharm, для C# та C++ – Microsoft Visual Studio, для PHP – PhpStorm.

Окремо можна відзначити Visual Studio Code. По суті – це редактор, але він має таку багатий набір плагінів, тому його можна зробити повноцінним IDE для будь-якої мови програмування. Але деякі моменти будуть гірші порівняно з IDE. Наприклад, особливо слабким місцем VSC є відсутність якісного дебагера, що ускладнює виправлення помилок.

Наступним є середовище розробки. Потрібно провести порівняльний аналіз наявних рішень та обрати найбільш підходящий варіант. Оскільки для реалізації frontend частини було обрано React, а для backend частини – Node.js, будемо порівнювати наступні інструменти: Sublime Text, Visual Studio Code, WebStorm, Atom, Text Mate. Результати порівняння занесено в табл. 2.3.

Таблиця 2.3 – Порівняння середовищ розробки за відповідністю вимогам

Критерій	Sublime Text	Visual Studio Code	WebStorm	Atom	Notepad++
Зручність інтерфейсу	+	+	+	+/-	+/-
Підтримка мов програмування	+	+	+	+	-
Розширення та плагіни	+	+	+	+	+
Відлагодження (Debugging)	-	+	+	+	-
Спільна робота (Collaboration)	-	+	+	+	-
Інтеграція з системами керування версіями	-	+	+	+	-
Швидкодія та продуктивність	+	+	-	+	+
Наявність безкоштовної версії	+	+	+/-	+	+

У результаті порівняння (див. табл. 2.3) очевидним фаворитом виявився Visual Studio Code – легкий, але потужний редактор коду, доступний для Windows, Linux та MacOS. В ньому є вбудована підтримка JavaScript, TypeScript та Node.js. Редактор також містить вбудовані інструменти для навігації по коду, автодоповнення базових мовних конструкцій та контекстні підказки. Однією з головних переваг VSCode є екосистема плагінів [37]. Наприклад, для виконання завдань знадобляться наступні плагіни:

- ES7+ React/Redux/React Native – плагін, що спрощує роботу з React шляхом надання сніпсетів та підтримки ES7 синтаксису;
- Prettier – плагін для форматування коду. Дозволяє з легкістю писати код в одному стилі, підтримує JSX синтаксис;

Серед систем контролю версії безперечним лідером є Git. Це open–source інструмент, який дозволяє ефективно відстежувати і контролювати зміни в проєктах. До особливостей Git можна віднести:

- збереження абсолютно всієї історії проєкту у вигляді так званих commit'ів. В будь-який час можна повернути стан проєкту до попереднього;
- гілки, що являють собою окремі незалежні лінії розробки і дозволяють розробникам паралельно працювати над різними частинами проєкту;
- підтримка інтеграції з такими сервісами як GitHub, GitLab, Bitbucket.

Окрім системи контролю, потрібен ще онлайн-репозитарій. Це місце, де розробники можуть спільно працювати над програмним кодом. Git та зовнішні репозитарії наразі є нероздільними інструментами.

Використання таких інструментів є гарною практикою, оскільки це дозволить зберігати всі файли в хмарі, а у випадку форс-мажору, проєкт не прийдеться переробляти з нуля. Найпопулярнішими репозитаріями є: GitHub, GitLab, Bitbucket.

Масштаб проєкту не потребує CMS, оскільки контент при наявності заповнених сцен буде не так швидко змінюватися, а ціль додатку – залучити до проходження різних тестів якомога більшу аудиторію і зацікавити наявністю індивідуальних результатів, навчання.

## Висновки до розділу 2

В даному розділі виділено послідовність практичних завдань, які в сукупності дозволять реалізувати поставлену ціль із розробки спеціального вебдодатку на психологічну тематику із заданими функціями та сучасним дизайном. Для досягнення переваг було здійснено аналіз та узагальнення аналогічних за тематикою та функціями розробок, створено власний план реалізації задуманого проєкту. В якості елементів технічного завдання поставлені вимоги до дизайну, структури та функціоналу вебдодатку.

Для відповідного технологічного забезпечення проведений уточнений аналіз та остаточний вибір засобів і технологій згідно технічного завдання. Таким чином, сформульовано та поділено на логічні кроки задачу практичної частини кваліфікаційної роботи – розроблення сучасного вебдодатку.

В якості контрольного прикладу застосовано метод прототипування розробки та проведення корекції на тестовому прикладі. Під час розробки прототипу перевірено функціональність ключових структурних елементів додатку, методи додавання сцен під час проходження тесту, фіксацію відповідей та переведення до нових сцен, а також досягнення багатомовного інтерфейсу. При цьому переконливо оцінені всі позитивні результати використання фреймворків JavaScript, Jason файлів та інших, досягнуто сумісність бібліотек та формату збереження багатомовних даних, застосування LocalStorage.

Наступним етапом є реалізація повномасштабної версії запланованого вебдодатку та остаточна демонстрація його функціональності завдяки обраним інструментам серед значної пропозиції.

## РОЗДІЛ 3

### ОПИС ПРАКТИЧНОЇ РЕАЛІЗАЦІЇ ПОВНОЇ ВЕРСІЇ ВЕБДОДАТКУ

#### 3.1 Програмна реалізація інтерфейсу користувача інтерактивного вебдодатку на прикладі квест-тесту

Застосування методу прототипування функцій та структури майбутнього вебдодатку, вибір стеку необхідних технологій для технічного втілення задуму дозволили перейти до послідовної реалізації поставленого завдання, розпочати основну верстку. Робота над фронтендом починається зі створення макетів сторінок вебдодатку. Макет головної сторінки наведено в додатку В, коди – в додатку Г. Дизайн сайту виконаний в мінімалістичному стилі, з двома домінуючими кольорами – фіолетовий (з відтінками) і охра (темно-жовтий), які поєднуються з нейтральними. Лістинги стилів сторінок представлені в додатках Д, Е, Ж. Специфіка змісту та призначення вебдодатку не потребує хедеру й додаткового меню. У головній частині розміщено заголовок теми, анонс, кнопка «Почати» і кнопки вибору мовного інтерфейсу. Фрагмент структури головної сторінки у VSC представлено на рис. 3.1.

![Screenshot of Visual Studio Code showing a project explorer on the left and a code editor on the right. The project explorer shows a folder named 'SYTE_RYBKA' with various files including 'img', 'favicon.png', 'fin.css', 'index.html', 'index1.html', 'question.css', 'question.html', 'question.js', 'quiz.en.json', 'quiz.uk.json', 'README.md', 'results.html', 'results.js', 'script.js', 'style.css', and 'text.json'. The code editor shows the content of 'index.html' with the following code: <pre>2 <html lang=](script.js)

Рисунок 3.1 – Структура всього проекту та зміст частини `<head> ...</head>`

Як видно (див. рис. 3.1), в частині <head> представлено посилання на файли стилів, скрипти, а також fonts.googleapis для використання спеціальних вебшрифтів [38]. Основний шлях користувач проходить, послідовно натискаючи на варіанти відповіді, розміщені на кнопках або блоках. Після натискання «Старт» головного вікна відбувається перехід на першу тематичну сторінку й перший вибір (стать) – кнопки «Чоловік» або «Жінка» (рис. 3.2).

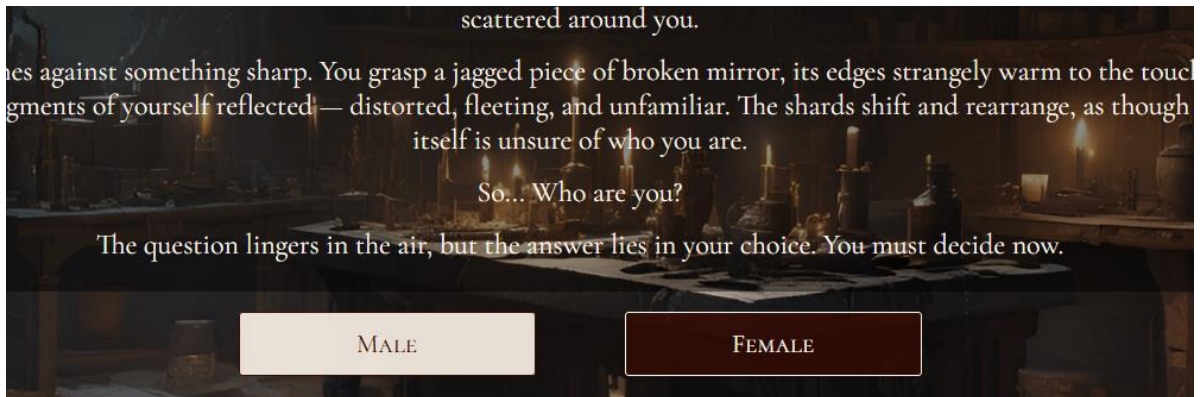


Рисунок 3.2 – Дизайн сторінки з вибором власної гілки згідно гендерної приналежності

Наступне вікно – перша сцена і «сюжетний» текст із варіантами відповіді, які представлені в трьох блоках на вибір (рис. 3.3).

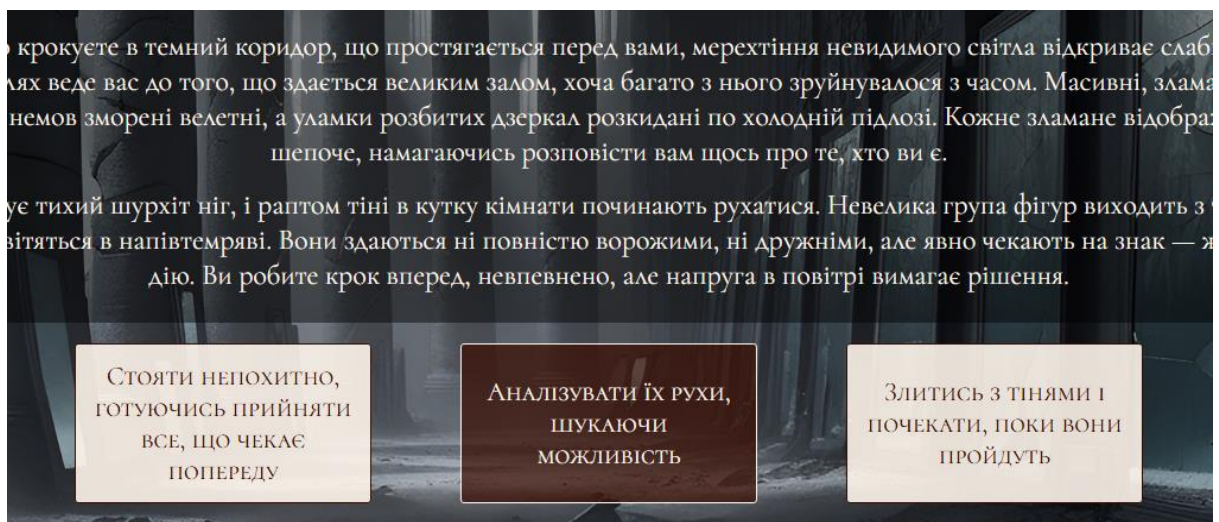
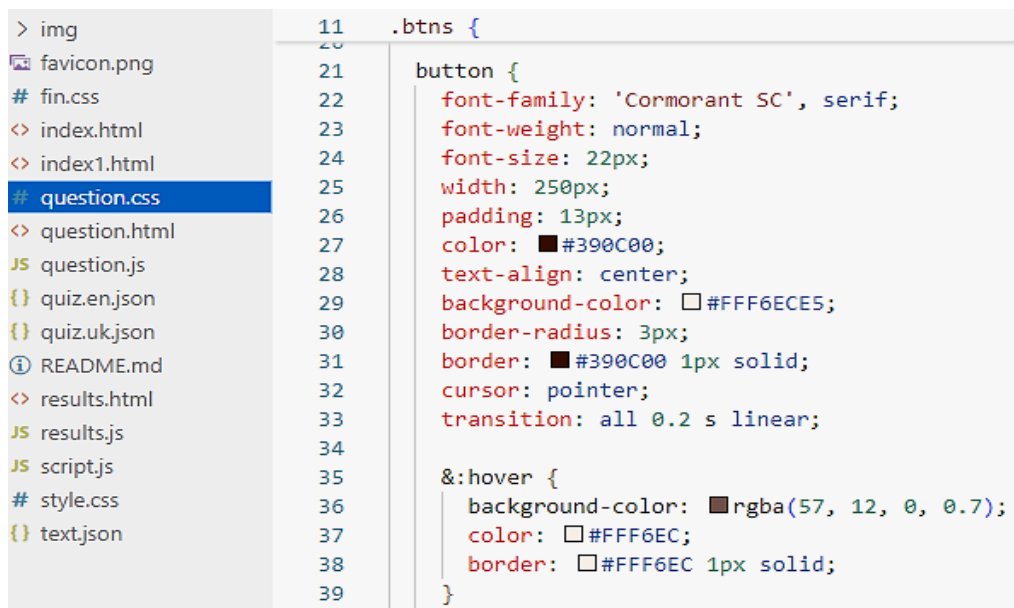


Рисунок 3.3 – Приклад оформлення фрагменту кожної сцени в тест-квесті на вебдодатку з трьома варіантами вибору подальшої власної траєкторії

Реалізація сцен і забезпечення вибору користувача були основними технічними задачами, які було вирішено шляхом комплексного застосування JASON-файлів та JS-скриптів.

Візуальні ефекти, такі як зміна кольору обраного для відповіді блоку, трансформації були забезпечені відповідними функціями CSS. Наприклад, на рис. 3.4 показано фрагмент опису дизайну та властивостей кнопки в question.css (додаток E) для вибору і фіксації відповіді.



```

11  .btns {
20
21  button {
22      font-family: 'Cormorant SC', serif;
23      font-weight: normal;
24      font-size: 22px;
25      width: 250px;
26      padding: 13px;
27      color: #390C00;
28      text-align: center;
29      background-color: #FFF6E5;
30      border-radius: 3px;
31      border: #390C00 1px solid;
32      cursor: pointer;
33      transition: all 0.2 s linear;
34
35      &:hover {
36          background-color: rgba(57, 12, 0, 0.7);
37          color: #FFF6EC;
38          border: #FFF6EC 1px solid;
39      }

```

Рисунок 3.4 – Фрагмент опису властивостей кнопки в question.css

Як бачимо (див. рис. 3.4), використані всі властивості шрифтів, границь `border-radius: 3px;` обрамлення `border: #390C00 1px solid;` та інші. Зміни дизайну при інтерактивній взаємодії [39] описані в класах `&:hover` та `&:active`.

Для опису сцен із питаннями і вибором відповідей створені відповідні JASON-файли [40]: варіант українською мовою, файл для варіанту англійською мовою. Питання оцінюються не традиційно «true» «false», а даються гілки на перехід до іншого питання. Фрагмент опису однієї сцени з одним питанням та врахуванням однієї з трьох відповідей показано на рис. 3.5. Таких питань створено 13 і далі слідує перехід до опису результатів – певного психологічного портрету, визначеного шляхом тестування.

```

21   {
22     "question": "Ви повільно встаєте на ноги, погляд затримується на зламаний поверхні найближчого дзеркала.
23     "answers": [
24       {
25         "text": "Стояти непохитно, готуючись прийняти все, що чекає попереду",
26         "points": {
27           "keeper": 1,
28           "revenant": 1
29         }
30       },
31       {
32         "text": "Аналізувати їх рухи, шукаючи можливість",
33         "points": {
34           "seeker": 1,
35           "conqueror": 1
36       }
37     },
38     {
39       "text": "Злитись з тінями [ ] почекаати, поки вони пройдуть",
40       "points": {
41         "veilwalker": 1,
42         "renegade": 1
43       }
44     }
45   ],
46   "background": "hall-bg.jpeg"
47 }

```

Рисунок 3.5 – Фрагмент обробки відповідей на 1 питання українською в Jason

Для обробки даних з question.json підключається набір скриптів, записаних у файлі question.js. Повний лістинг файлів скриптів представлено в додатку З (головна), додатку И (обробка питань), додаток К (фінал). Розглянемо окремі функції. Наприклад, на рис. 3.6 показано частину коду для обробки наданих відповідей і переадресації користувача на новий етап.

```

10  btnContainer.onclick = handleClick;
11
12  continueStartedQuiz();
13
14  getQuestions().then(startQuiz);
15
16  function continueStartedQuiz() {
17    //перевірка, чи існує індекс поточного питання та дані про відповіді
18    const index = localStorage.getItem('index');
19    const storedAnswers = localStorage.getItem('answers');
20
21    //якщо є збережений індекс, оновлюємо поточне питання
22    if (index) {
23      currentQuestion = +index;
24
25      if (currentQuestion > 12) {
26        location.href = "results.html";
27      }
28    }
29    if (storedAnswers) [ ]
30    answers = JSON.parse(storedAnswers);
31  }

```

Рисунок 3.6 – Частина коду JavaScript для обробки даних відповідей

Код обробки події натискання на кнопку для верифікації відповіді й переведення на відповідну гілку показано на рис. 3.7. Варіанти відповідей розглядаються як масиви, кожний ключ обробляється в циклі, визначаємо індекс поточної відповіді (рядок 81), нараховуємо бали для кожної відповіді (рядок 82). Далі оновлюємо відповіді для кожного ключа (рядки 85-86) і зберігаємо прогрес (рядок 90).

```

79 function handleClick(e) {
80   if (e.target.matches('button')) {
81     const i = Array.from(btnContainer.children).indexOf(e.target); // визначаємо індекс поточної
82     const {points} = questions[currentQuestion].answers[i]; // бали для вибраної відповіді
83
84     // оновлюємо відповіді для кожного ключа
85     Object.keys(points).forEach(key => {
86       | answers[key] = (answers[key] || 0) + points[key];
87     });
88     currentQuestion++;
89
90     storeProgress();
91
92     // якщо є наступне питання, показуємо його, інакше переходимо до результатів
93     if (currentQuestion < questions.length) {
94       | showQuestion(currentQuestion);
95     } else {
96       | showResults();
97     }

```

Рисунок 3.7 – Фрагмент коду JavaScript для обробки події натискання кнопки

Код містить окремі корисні функції, наприклад, збереження відповідей, повернення на окремий пункт або ж ануляцію історії. Для цього застосовується localStorage – це простий спосіб зберегти вибори користувача у браузері, навіть після перезавантаження сторінки. Необхідно пояснити, що localStorage:

- це API браузера, яке дозволяє зберігати ключ-значення у форматі рядків;
- дані не зникають при оновленні сторінки;
- ідеально підходить для збереження відповідей, мови, прогресу.

В практично реалізованому коді необхідно застосувати функцію типу saveAnswer, яка може бути описана таким чином:

```
function saveAnswer(sceneId, answerText) {
```

```

    const stored =
JSON.parse(localStorage.getItem('answers') || '{}');
    stored[sceneId] = answerText;
    localStorage.setItem('answers',
JSON.stringify(stored));
};

```

На рис. 3.8 показано фрагмент коду, в якому показано застосування описаної вище функції `storedAnswer` і `localStorage`.

```

16 function continueStartedQuiz() {
17     const index = localStorage.getItem('index');
18     const storedAnswers = localStorage.getItem('answers');
19     //якщо є збережений індекс, оновлюємо поточне питання
20     if (index) {
21         currentQuestion = +index;
22         if (currentQuestion > 12) {
23             location.href = "results.html";
24         }
25     }
26     if (storedAnswers) {
27         answers = JSON.parse(storedAnswers);
28     }
29 }

```

Рисунок 3.8 – Опис процедури збереження відповіді

Зрештою, на завершення тесту може бути цікаво подивитися всі відповіді в кінці за таким кодом:

```

function showResult() {
    const answers =
JSON.parse(localStorage.getItem('answers') || '{}');
    console.log('User answers:', answers);
}

```

На завершення опису функцій розробленого вебдодатку важливо оглянути технологію відправлення відповідей на бекенд. Відправка включає такі основні дії:

1. Зібрати дані відповідей у форматі JSON.

2. Використати `fetch()` або `axios` для POST-запиту.
3. Обробити відповідь (наприклад: «Ваш результат збережено»).

Приклад POST-запиту в React, на останньому кроці (тобто на етапі `end`), ми можемо викликати за таким кодом (рис. 3.9):

```
useEffect(() => {
  if (current === 'end') {
    fetch('https://your-backend.com/api/save-test', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json'
      },
      body: JSON.stringify({
        answers,
        language: lang,
        timestamp: new Date().toISOString()
      })
    })
    .then(res => res.json())
    .then(data => {
      console.log('Server response:', data);
    })
    .catch(err => {
      console.error('Error sending data to backend:', err);
    });
  }
});
```

Рисунок 3.9 – Фрагмент формування запиту на обробку даних

На бекенді відправлений запит потребує:

- прийняти POST-запит на `/api/save-test`;
- обробити JSON-тіло;
- зберегти в базу або на e-mail;
- відповісти `{ success: true }`.

Для програмування бекенду був обраний фреймворк Node.js (Express), ефективність якого доведена в подібних проєктах.

### 3.2 Застосування генеративного штучного інтелекту в розробці елементів графіки для сучасного дизайну вебдодатку та адаптивність

Тематика сайту та його сюжетність з елементами загадкового квесту потребували відповідного графічного дизайну. Було прийняте рішення скористатися допомогою штучного інтелекту (ШІ), що дає можливість непрофесіоналу отримати якісний матеріал за умови, що буде наданий гарний опис. Штучний інтелект для генерації зображень – це технологія, яка використовує алгоритми машинного навчання для створення графічного контенту з нуля або на основі текстових описів.

Надамо короткий опис порівняння (табл. 3.1) популярних генераторів зображень для створення реалістичних або стилізованих фентезі-ілюстрацій.

Таблиця 3.1 – Порівняльний аналіз моделей ШІ для генерації графіки

Модель, призначення	Розвинені можливості, +	Слабкі сторони, –
DALL·E (від OpenAI) [41] Стиль: більш реалістичний, обережний, гнучкий Для створення ілюстрацій із точним контролем або поєднанням елементів	+ інтегрований у ChatGPT + працює добре з інструкціями в тексті (“намалюй... у стилі...”) + підтримує inpainting (редагування зображення)	– іноді менш художній у стилях, ніж інші моделі
Midjourney [42]: підходить для: ілюстрацій у стилі обкладинок фентезі книг, ігор, concept-art	+ чудовий художній стиль: фентезі, sci-fi, деталізація + красиве освітлення, композиція, текстури	– не працює напряму у браузері – потрібен Discord – менше точності при складних запитах
Stable Diffusion (SD, SDXL) підходить для: повного контролю, фентезі стилів, кастомних моделей, fine-tuning [43]	+ Повністю відкритий код — можеш запускати на своєму комп’ютері + Тисячі кастомних моделей (наприклад “anime-fantasy”, “realistic RPG female”, тощо) + Плагіни: ControlNet, style adapters, LoRA	– Складніше для новачка без інтерфейсу
Leonardo.Ai [44]: підходить для: тих, хто хоче зробити фентезі графіку для ігор, книг, UI, без технічного клопоту	+ Онлайн платформа з friendly UI + Добре генерує ігрові об’єкти, персонажів, фентезі сцени + Підтримка LoRA моделей, img2img	– Потребує реєстрації, обмеження на безкоштовні кредити

Отже, такі моделі, як DALL-E, Midjourney чи Stable Diffusion, здатні створювати реалістичні чи стилізовані ілюстрації, портрети, ландшафти та навіть складні дизайнерські композиції. Основна мета цієї технології – автоматизація творчого процесу, розширення можливостей художників, дизайнерів та інших креативних спеціалістів. Завдяки ШІ, можна швидко генерувати візуальний контент для реклами, відеоігор, кінематографа, маркетингу та навіть наукових досліджень. При цьому використання таких зображень не потребує додаткових дозволів, достатньо просто визнати використання ШІ. Користь таких інструментів очевидна: вони допомагають зекономити час, знижують витрати на створення графіки, відкривають нові горизонти для творчості та дають змогу візуалізувати ідеї, які є супроводом інших технічних власних розробок [45]. Крім того, технології ШІ дозволяють людям без художніх навичок створювати якісні зображення, розширюючи доступ до творчості.

Для отримання графіки вебдодатку були використані пробні генерації безпосередньо в ChatGPT [46], а також в Leonardo.Ai. В першому рендеринг тривав (45 хвилин), в другому зроблені всі зображення. Ресурс Leonardo.Ai – потужний інструмент на основі ШІ, що спеціалізується на генерації зображень (рис. 3.10). Однією з головних переваг Leonardo.Ai є його висока якість рендерингу та гнучкість налаштувань. Платформа дає змогу створювати зображення як у реалістичному, так і у художньому стилі, функція редагування дозволяє покращувати або змінювати вже згенеровані зображення, додаючи їм деталей або коригуючи певні елементи. Leonardo.Ai має інтуїтивно зрозумілий інтерфейс, що значно спрощує процес генерації. Також сервіс забезпечує швидку обробку запитів і пропонує безкоштовний доступ з можливістю отримання додаткових кредитів для генерації. Це дозволило експериментувати з різними варіантами без значних фінансових витрат. Загалом, Leonardo.Ai став чудовим вибором для мого проєкту завдяки своїй функціональності, простоті використання та можливості отримати якісні результати навіть без досвіду роботи з подібними інструментами.

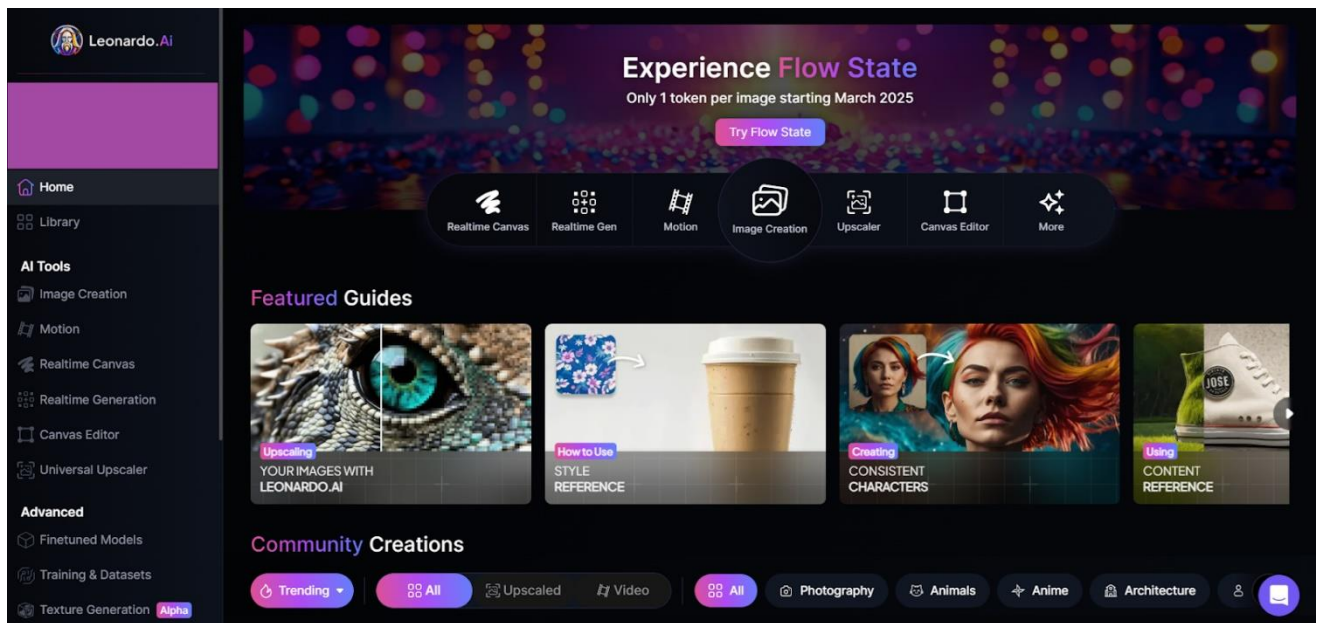


Рисунок 3.10 – Головна сторінка Leonardo.Ai [44]

Ресурс Leonardo.Ai був використаний для генерації персонажів та фонів. В контексті цілісного задуму це були не випадкові ідеї, а продумана система характерів і сенсу. Для кожного персонажа сцени, фону необхідно продумати деталі: який у нього характер, що вплинуло на його світогляд, яка у нього історія, ким він був у минулому і ким може стати в майбутньому. Чи народився він аристократом, чи став воїном через обставини? Чи є в нього сильні моральні принципи, або ж війна змінила його до невпізнання? На цьому етапі можна використовувати додаткові інструменти на базі ШІ, які допомагають генерувати бекграунд і характер персонажа, або ж опрацювати всі деталі самостійно. ШІ дозволяє швидше отримати варіанти, що може бути корисним для натхнення, але ручне створення дає більше унікальності та глибини. Такий підхід дозволяє не просто створювати випадкові зображення, а будувати логічно пов'язані світи та персонажів, які мають власну історію, мотивацію та розвиток.

Сформувавши більш чітке уявлення про героя та детальний опис, яким він буде у фінальному вигляді, потрібно сформулювати для побудови графіки відповідний промпт. Запит складається з кількох основних частин: спочатку йде прохання до дії (по суті, вказівка сформулювати промпт для генерації

зображення), далі йде опис персонажа, який включає його характер та ключові деталі, а також побажання щодо візуального оформлення, тобто стиль, колірна гама або специфічні художні елементи, які зроблять фінальне зображення більш точним і атмосферним. Сформований промпт буде виглядати, як показано в табл. 3.2. В оригіналі він буде поданий англійською.

Таблиця 3.2 – Готовий prompt, створений за допомогою ChatGPT для генерації зображення у Midjourney, Leonardo.Ai або Stable Diffusion

Оригінальний prompt англійською	Переклад для розробника
A tarot card portrait of a mystical seeker of truth, medieval necromancer-inspired clothing, deep brown and violet tones, ethereal lighting, ancient knight-like features, a sharp, inquisitive gaze, surrounded by symbols of knowledge and danger, glowing runes, parchment scrolls, alchemical diagrams. The figure radiates a deep hunger for hidden knowledge, standing on the edge between enlightenment and madness. Background subtly hints at forbidden tomes and whispering shadows. Highly detailed, painterly, vintage tarot card style, textured canvas, gothic-fantasy aesthetic, muted colors with mystical glow	Портрет містичного шукача істини, виконаний на картах Таро, одяг, натхненний середньовічними некромантами, насичено-коричневі та фіолетові тони, ефірне освітлення, риси обличчя, схожі на стародавніх лицарів, гострий, допитливий погляд, оточений символами знання та небезпеки, сяючими рунами, пергаментними сувоями, алхімічними діаграмами. Фігура випромінює глибоке прагнення до прихованих знань, стоячи на межі між просвітленням та божевіллям. Фон ледь помітно натякає на заборонені фоліанти та шепіт тіней. Висока деталізація, живописність, стиль вінтажних карт Таро, текстуроване полотно, готично-фентезійна естетика, приглушені кольори з містичним сяйвом.

Далі наведено пояснення складових, від яких буде відштовхуватися обрана модель ШІ для генерації зображення (табл. 3.3).

Таблиця 3.3 – Пояснення складових у промпті за призначенням

Елемент запиту	Що передає
"tarot card portrait"	Стиль і формат (фігура, символізм)
"medieval necromancer-inspired clothing"	Підкреслює магічне й темне походження
"brown and violet tones"	Атмосфера, палітра кольорів
"inquisitive gaze, surrounded by symbols of knowledge and danger"	Передає твою описану суть
"glowing runes, parchment scrolls, alchemical diagrams"	Візуальний мотив знань
"standing on the edge between enlightenment and madness"	Внутрішній конфлікт
"vintage tarot card style"	Стилізація під класичні таро
"gothic-fantasy aesthetic"	Додає глибини й художньої виразності

Меню створення зображень на сайті Leonardo.Ai (рис. 3.11) може спочатку здатися заплутаним, але насправді воно інтуїтивно зрозуміле. Потрібно вставити готовий промпт і обрати потрібну модель.

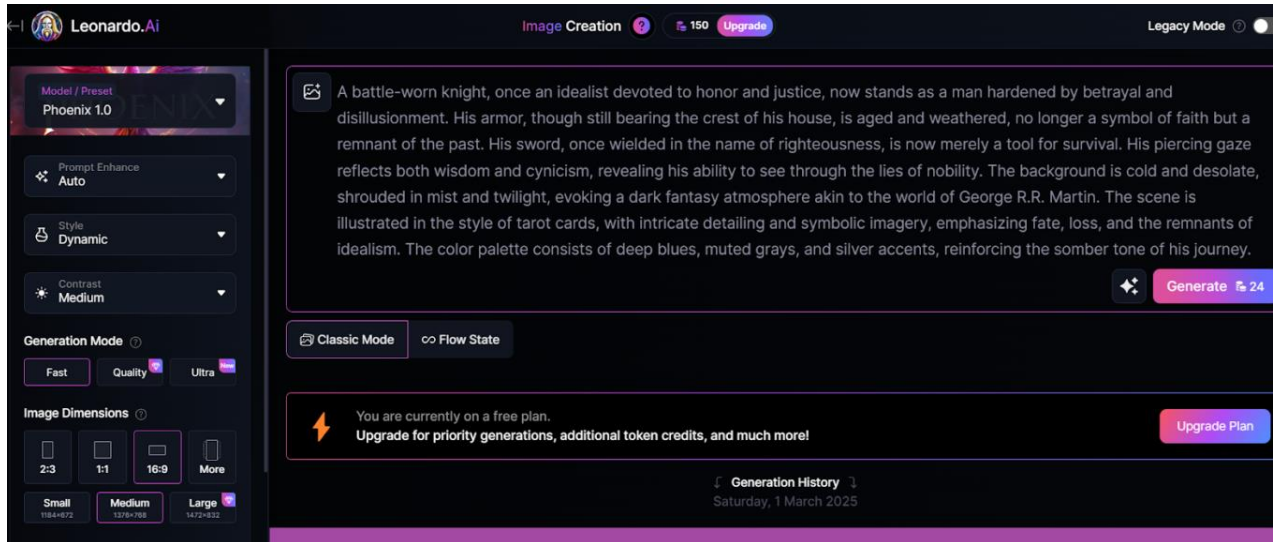


Рисунок 3.11 – Меню створення зображення та уведений готовий промпт

Про моделі ШІ (рис. 3.12) можна багато розповісти, але краще просто спробувати кожну, щоб зрозуміти, яка підходить для конкретного проєкту. Їх назви та короткий опис у підказках, що з'являються при наведенні мишкою, дають уявлення про той чи інший варіант.

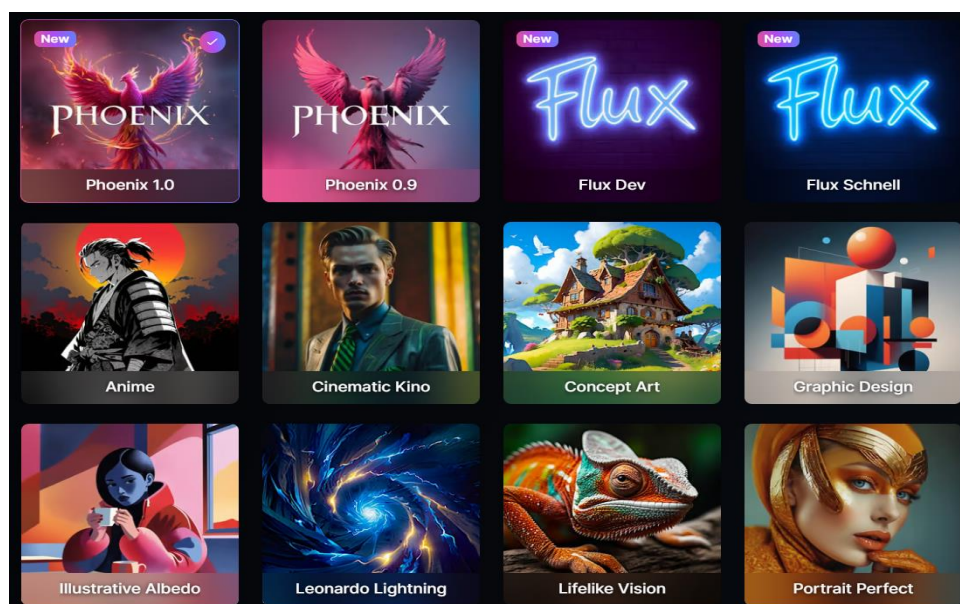


Рисунок 3.12 – Доступні моделі на Leonardo.Ai

Наступним кроком є налаштування стилю, де представлені різні варіанти, кожен з яких чітко визначає напрямок оформлення: від реалістичного до художнього, абстрактного чи навіть сюрреалістичного стилю. Назви стилів також доволі інформативні і дають зрозуміти, що саме буде створене. Крім того, важливим параметром є вибір розміру вихідного зображення, що дозволяє адаптувати створене зображення під різні потреби.

У загальному, налаштування в Leonardo.Ai дуже гнучкі та зручні, навіть у безкоштовній версії, що дозволяє без обмежень експериментувати з різними варіантами, не турбуючись про фінансові витрати. У безкоштовному плані діє система кредитів: на день дається 150 кредитів для генерації. Цього може бути недостатньо для безперервної роботи, але цей момент компенсується тим, що по якості безкоштовна та платна моделі в принципі однакові. Тобто, навіть у безкоштовній версії можна отримати дуже високоякісні результати, що робить платформу доступною для широкого кола користувачів.

Після того як були проведені налаштування моделі та стилю, які здались найбільш доречними для створення персонажа, були отримані 4 варіанти зображень (рис. 3.13). Кожен з них був унікальним, з різними акцентами, що дозволяло краще зрозуміти, як кожна деталь впливає на кінцевий вигляд героя.

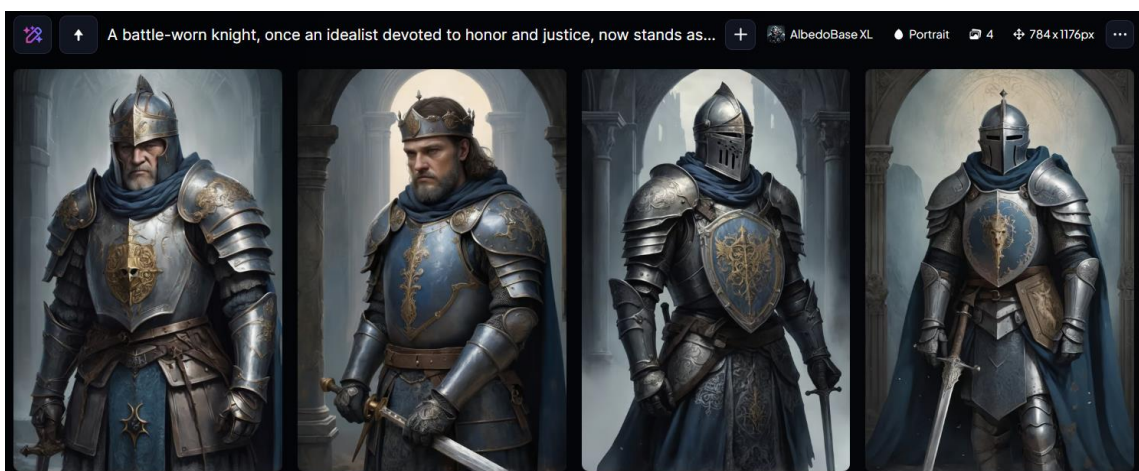


Рисунок 3.13 – Створені зображення за різними налаштуваннями

Для експерименту був змінений промпт шляхом додавання стильових деталей, а також обрані інший стиль і модель. Це дозволило перевірити, як

навіть невеликі зміни у формулюванні запиту та налаштуваннях впливають на кінцевий результат (рис. 3.14). У підсумку вийшло візуально зовсім інше зображення, хоча воно все ж зберігало певні схожості з попередніми варіантами. Такий досвід показує, що генерація зображень – це не випадковий процес, а можливість гнучко керувати результатом, змінюючи окремі деталі та поступово наближаючись до бажаного образу.

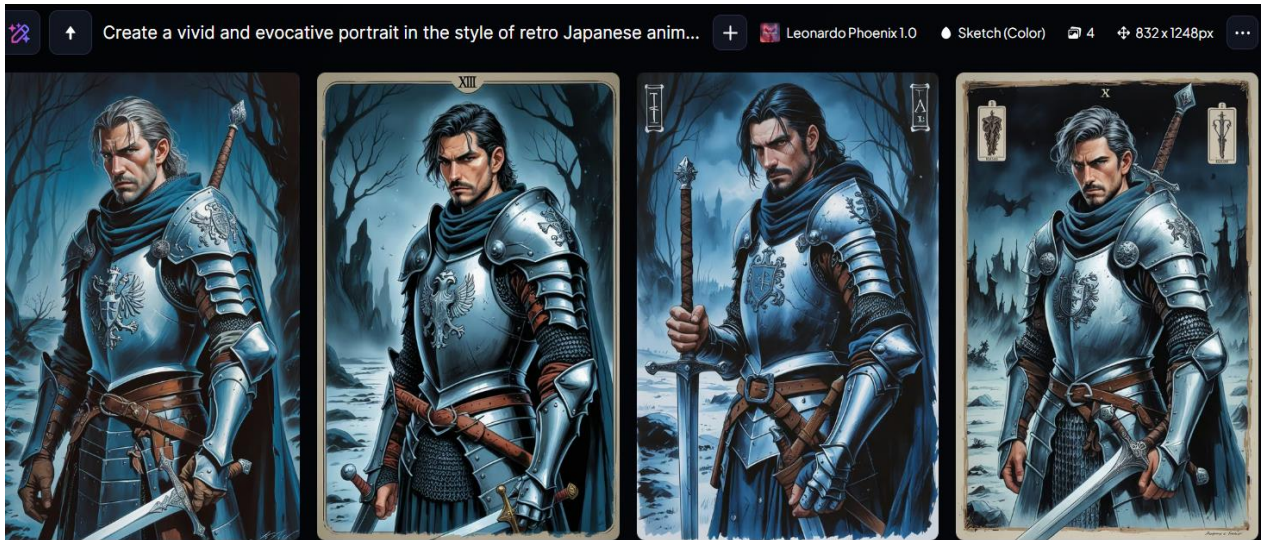


Рисунок 3.14 – Створені зображення зі зміненим промптом

У результаті роботи з Leonardo.Ai стало очевидно, що генерація зображень за допомогою ШІ – це не просто автоматичний процес, а справжній інструмент для творчості. Навіть у межах одного персонажа можна експериментувати з описами, стилями та налаштуваннями, отримуючи при цьому унікальні, але логічно пов'язані варіанти. Гнучкість платформи дозволяє досягати потрібного результату навіть у безкоштовній версії, а система налаштувань дає можливість контролювати кінцевий вигляд зображень. Зміна деталей у промпті чи вибір іншої моделі дають суттєво відмінні результати, але при цьому зберігають ключові риси задуманого персонажа. Це доводить, що ШІ не замінює творчий підхід, а навпаки – допомагає його розкрити та зробити візуалізацію ідей доступнішою. Основні ідеї застосування ШІ в програмній інженерії представлені в роботі [47].

### **3.3 Економічне оцінювання вартості розробки вебдодатку з урахуванням застосованих вебтехнологій**

У ході підготовки і виконання кваліфікаційної роботи було створено інформаційний інтерактивний вебдодаток. Такий сайт не буде приносити прямого прибутку, але очікується, що буде відвідуваним. Головною цінністю є цікавий контент на актуальну тематику. Його можна використати як портфоліо розробника або як промо-сайт приватного бізнесу. Технологія розробки дозволяє додати згодом інші функції й платні послуги.

Однак, питання вартості розробки будь-якого програмного продукту завжди актуальне, оскільки надає можливість оцінити необхідний бюджет замовнику й масштаб проекту [48]. Враховуючи, що більшість сайтів все ж створюються з метою отримання прибутку, проведемо економічний аналіз, який включатиме: витрати на дизайн, розробку та різноманітні додаткові витрати.

Економічний аналіз – це процес вивчення та оцінки економічних аспектів явищ, процесів, рішень чи проектів з метою зрозуміння їхньої ефективності, вартості, можливих вигід чи ризиків. У контексті розробки вебдодатків це включає в себе [49]:

- аналіз витрат;
- прогнозування прибутків;
- складання графіків та встановлення критеріїв;
- оцінку ринкового потенціалу;
- обґрунтування оптимальних стратегій.

Для обґрунтування доцільності розробки та впровадження проектного рішення варто провести розрахунок наступних економічних показників:

- оплата праці;
- витрати на куповані вироби;
- транспортно–заготівельні витрати;
- накладні витрати;

– інші витрати.

Розмір оплати праці визначається за різними критеріями, такими як складність та умови виконуваної роботи, професійні якості працівника, вплив його трудової діяльності та економічної діяльності підприємства. Заробітна плата формується з основної та додаткової частин оплати праці.

Базова заробітна плата нараховується, виходячи з тарифної сітки, договірних розцінок або посадових окладів, та не залежить від господарської діяльності підприємства.

Додаткова оплата праці – різноманітні матеріальні заохочення, наприклад – премія або надбавка за шкідливість. Нараховуються зазвичай по результатам запланованих та виконаних показників, кваліфікаційної підготовки, умов виробництва тощо.

Візьмемо для розрахунку мінімально необхідний склад команди розробників: дизайнер (диз), програміст (прог) та тестувальник (тест). Візьмемо середню зарплату дизайнера – 20000 грн на місяць, програміста – 30000 грн на місяць та тестувальника – 15000 грн на місяць.

Орієнтовно, розробка проекту займе 2 тижні, тобто 10 робочих днів.

Працівники будуть працювати по стандартному 40-годинному графіку (8 годин на день). Використаємо дані про погодинну оплату графічного дизайнера [50] (використаємо нижню границю) – 121 грн, програміста – 170 грн, тестувальника – 220 грн.

Формула розрахунку заробітної плати:

$$Z_{\text{п}} = (K_{\text{днів}} \times V_{\text{роб}}) \times T_{\text{г}}, \quad (3.1)$$

де  $K_{\text{днів}}$  – кількість робочих днів;

$V_{\text{роб}}$  – тривалість робочого дня в годинах;

$T_{\text{г}}$  – годинна ставка розробника.

Розрахунок за 10 робочих днів:  $Z_{\text{диз}} = (10 \times 8) \times 121 = 9680$  грн;

$Z_{\text{прог}} = (10 \times 8) \times 170 = 13600$  грн;  $Z_{\text{тест}} = (10 \times 8) \times 220 = 17600$  грн.

Витрати на придбання ( $V_{\text{пр}}$ ) в нашому випадку становитимуть: USB-накопичувач – 200 грн, папір офісний (1 упаковка, 250 аркушів) – 240 грн, безлімітний інтернет на місяць – 350 грн. Всього 790 грн.

Визначення накладних транспортних витрат. Транспортні витрати ( $V_{\text{тр}}$ ) даного типу становлять 10% від витрат на придбання товарів [51].

$$V_{\text{тр}} = V_{\text{пр}} \times 0,10, \quad (3.2)$$

де  $V_{\text{пр}}$  – витрати на придбання.

Розрахунок:  $V_{\text{тр}} = 790 \times 0,11 = 79,0$  грн.

Визначення накладних витрат. Накладні витрати ( $V_{\text{н}}$ ) проєктних організацій складають 25% витрат на оплату праці.

$$V_{\text{н}} = (Z_{\text{диз}} + Z_{\text{прог}} + Z_{\text{тест}}) \times 0,25, \quad (3.3)$$

де  $Z_{\text{диз}}$  – зарплата дизайнера;

$Z_{\text{прог}}$  – зарплата програміста;

$Z_{\text{тест}}$  – зарплата тестувальника.

Розрахунок:  $V_{\text{н}} = (9680 + 13600 + 17600) \times 0,25 = 10220$  грн.

Визначення інших витрат (електроенергія, вода тощо). Інші витрати ( $V_{\text{ін}}$ ) передбачають всі витрати, що не враховані в попередніх розрахунках. Вони становлять 6% до витрат на оплату праці:

$$V_{\text{ін}} = (Z_{\text{диз}} + Z_{\text{прог}} + Z_{\text{тест}}) \times 0,06. \quad (3.4)$$

Розрахунок:  $V_{\text{ін}} = (9120 + 13600 + 6800) \times 0,06 = 2452,8$  грн.

Загальна сума витрат обчислюється за формулою:

$$V_{\text{сум}} = (Z_{\text{диз}} + Z_{\text{прог}} + Z_{\text{тест}}) + V_{\text{пр}} + V_{\text{тр}} + V_{\text{н}} + V_{\text{ін}}. \quad (3.5)$$

Розрахунок:  $V_{\text{сум}} = (9680 + 13600 + 17600) + 790 + 79 + 10220 + 2452,8 = 54\,421,8$  грн.

Окремо варто розглядати витрати на хостинг. Не кожен хостинг наразі підтримує розміщення Node.js додатків. Гарним рішенням з підтримкою необхідного інструментарію є платформа hosting Ukraine [52].

Найдешевший варіант нам не підходить, оскільки нам потрібно мінімум 2 сайти (один для React–додатку, інший для Node.js сервера). Кращий варіант коштує 354 грн/міс. та надає 20 гігабайт пам'яті на NVMe–накопичувачі, 1 гігабайт оперативної пам'яті та можливість генерації безкоштовного SSL–сертифікату. В цілому – прийнятний варіант для нашого вебдодатку. Найдорожчий варіант за 507 гривень надає більше пам'яті та можливість створювати безмежну кількість сайтів, але в нашому випадку це непотрібно. Зупинимось на варіанті за 354 грн/міс.

Таким чином, остаточний підрахунок сумарних витрат складе 54 775,8 грн при роботі 3-х спеціалістів протягом 10 днів. Дані розрахунки носять наближений характер, звичайно, кожна компанія економить ресурси. Тому, наприклад, за рахунок зменшення накладних витрат розробка може бути дешевшою. Загалом, проведення розрахунку пояснює джерела походження цін на виконання аналогічних розробок, які здебільшого носять комерційний характер і мають період окупності.

### **Висновки до розділу 3**

В практично-орієнтованому розділі кваліфікаційної роботи представлено результати застосування обраних технологій на завершальній стадії розроблення вебдодатку у вигляді попередньо-визначеного формату квест-тесту. Повний програмний проєкт складається з набору файлів типу .html, .css, .json, .js, а також графічних файлів та ін. Показано результат розробки дизайну та функціоналу окремих сторінок, скан-коди фрагментів найбільш функціональних частин додатку, методологія взаємодії між частинами коду, описаних на різних мовах та фреймворках.

Значну технічну частину проєкту займає опис функцій обробки даних тесту, навігація між сторінками в залежності від обраних варіантів відповідей, дублювання функцій на різних мовах. Вибір і використання технології JSON було виправданим та показало відповідний очікуваний ефект. Для збереження пройдених питань у браузері використано localStorage, що дозволяє відслідковувати прогрес тестування, повернутися назад. Для відправлення даних на сервер використано метод POST.

Окремий підрозділ відведено огляду результатів застосування генеративних моделей штучного інтелекту для розробки серії специфічних зображень, які використані в дизайні вебдодатку ілюстративно. Серед інших протестованих обрана модель Leonardo.Ai та створення спеціального промпту в ChatGPT.

Розрахунок економічної вартості веброзробки проведено шляхом оцінки часу, необхідного для веброзробки, оплати праці відповідних фахівців сумарно по видам робіт, а також враховував всі типи витрат (постійні, накладні) з розшифровкою та усередненими даними. Загальна вартість подібного проєкту складає орієнтовно 54 775,8 грн разом із хостингом без урахування витрат на адміністрування та подальше обслуговування вебдодатку.

## ВИСНОВКИ

У ході виконання кваліфікаційної роботи було систематизовано та проаналізовано значну кількість сучасного інструментарію веброзробника. Вимоги до вебдодатків наростають надзвичайно швидко. При цьому для кожної задачі потрібен свій набір інструментів, технологій та підхід – універсальних рішень не існує.

В роботі описані та порівняні frontend-рішення, серед яких HTML&CSS, особлива увага була приділена JavaScript-фреймворкам. Були порівняні популярні фреймворки React, Angular, Vue, Svelte, досліджені особливості використання кожного з них. Універсальним рішенням виявився React. Окремо розглядалися системи керування контентом (CMS): досліджено класифікацію, особливості застосування та галузі використання, для яких вони призначені.

Додатково проведений аналіз допоміжного програмного забезпечення: системи контролю версій та хмарні репозитарії, інтегровані середовища розробки, пакетні менеджери та контейнеризація.

У прикладній частині роботи був спроектований та розроблений інтерактивний вебдодаток у вигляді квест-тесту із можливістю переключення мовного інтерфейсу на кілька мов, обробки відповідей користувача при проходженні всіх кроків тесту, передавання відповідей на серверну частину, збереження відповідей. Запорукою успішного завершення роботи і виконання завдань було планування й дотримання наступних вимог:

1. Формування обґрунтованого стеку сучасних технологій з урахуванням ретельного аналізу переваг та недоліків різних продуктів. В результаті були обрані найоптимальніші інструменти: окрім традиційних frontend-технологій HTML&CSS, JavaScript, були використані можливості фреймворку React, Visual Studio Code – редактор коду, обробка сцен в JASON, технологія збереження даних в LocalStorage, backend-фреймворк Node.js, Git як система контролю версій, GitHub – хмарний репозиторій.

2. Застосування методу попереднього прототипування вебдодатку з огляду на специфіку його призначення і використання дозволила провести перевірку функцій на реалістичній моделі і зрозуміти алгоритм обробки даних та шляхів масштабування подібних сторінок у розгалуженій структурі.

3. Використання системи штучного інтелекту Leonardo.Ai замість графічних редакторів дозволило згенерувати серію необхідних графічних елементів, які виглядали професійно й привабливо. Для отримання запланованого результату основну увагу потрібно приділити генеруванню відповідного запиту (промпту), який також виконувався за допомогою іншої моделі штучного інтелекту – ChatGPT.

4. Додавання системи обробки і збереження даних з нескладним бекендом підвищило функціональність додатку.

5. Здійснено апробацію результатів шляхом публікації тез доповідей на міжнародних конференціях (додаток Л).

Проведено економічний аналіз. Загальна вартість подібного проєкту складає орієнтовно 54 775,8 грн при роботі 3-х спеціалістів протягом 10 днів разом із хостингом без урахування витрат на адміністрування та подальше обслуговування вебдодатку.

Подальше вдосконалення роботи можливо в напрямку розвитку підтримки SEO, оскільки React не досить добре з ним взаємодіє. Також можливе розширення функціоналу вебдодатку, додавання мовних профілів.

Подібний додаток може бути частиною більш масштабного проєкту будь-якої системи забезпечення послуг, розвинутий у більш соціально-значимий корисний продукт, запит на які в суспільстві стабільно зростає. Наприклад, психологічна служба, юридична компанія або є вибір різних опцій дизайну, будівництва тощо. В такому сенсі розробка може отримати практичне застосування.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Гнатишин М. Аналіз сучасних тенденцій розвитку технологій веброзробки. *Природничі та гуманітарні науки. Актуальні питання*: Матер. VI Міжнародної студентської науково-технічної конференції, м. Тернопіль, 27-28 квітня 2023 року. Тернопіль: ТНТУ ім. І. Пулюя, 2023. С. 132-133.
2. Шкарупа А. О., Стрельцов О. А. Аналіз особливостей сучасних вебдодатків. «Automation and Development of Electronic Devices» ADED–2022: Collection of Students' Scientific Paper, 2022. Part 2. P. 102–107.
3. A. K. Goel, R. Bakshi, K. K. Agrawal. Web 3.0 and Decentralized Applications. *Proc. 2022 the 2nd Int. Conf. on Innovative Research in Renewable Energy Technologies*. 2022. P. 1–8.
4. Клушин Ю.С., Захарчин Ю.Б. Підвищення швидкості роботи веб-додатків. *Комп'ютерні системи та мережі*. 2020. Том 2. № 1. С. 33–43.
5. K. E. Ogundeyi, C. Yinka–Banjo. WebSocket in real time application. *Nigerian Journal of Technology*. 2019. Vol. 38, №. 4. P. 1010 –1020.
6. More than 5 billion people. *We are social*. URL: <https://cutt.ly/awT5rDs2> (дата звернення: 20.10.2025).
7. Jason Beaird, Alex Walker, James George. *The Principles of Beautiful Web Design*. SitePoint, 2020. 282 p.
8. Desktop vs Mobile vs Tablet Market Share Worldwide - September 2025. *StatCounter*. URL: <https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet/worldwide/#monthly-202409-202509-bar> (дата звернення: 20.10.2025).
9. Cross–Platform Mobile App Development with Flutter–Xamarin–React Native. *Medium*. URL: <https://cutt.ly/iwT5ttuc> (дата звернення: 20.04.2025)..
10. Chuck Musciano, Bill Kennedy, *HTML & XHTML: The Defenitive Edition*. O'Reilly, 2006. 680 p.
11. What is HTML Preprocessor? *GeeksforGeek*. URL: <https://cutt.ly/BwT5iubA> (дата звернення: 20.10.2025).

12. CSS: Cascading Style Sheets. URL: <https://developer.mozilla.org/en-US/docs/Web/CSS> (дата звернення: 20.10.2025).
13. Majida Laaziri, Khaoula Benmoussa, Samira Khoulji, Kerkeb Mohamed Larbi, Abir El Yamami. Analyzing bootstrap and foundation front-end frameworks: a comparative study. *International Journal of Electrical and Computer Engineering (IJECE)*. 2019. Vol. 9, №. 1. P. 713–722
14. Marjin Haverbeke. *Eloquent JavaScript*. 3-rd edition. No Starch Press, 2018. 472 p.
15. Stack Overflow Trends. *Stack Overflow*. URL: <https://trends.stackoverflow.com/?tags=jquery%2Cangularjs%2Cangular%2Creactjs> (дата звернення: 24.04.2025).
16. Канцібер Д.С. Використання мови TypeScript при розробці вебсайтів. *Стратегічний менеджмент агропродовольчої сфери в умовах глобалізації економіки: безпека, інновації, лідерство*: Матер. І міжн. науково-практичної конференції, м. Полтава, 28 вересня 2023 року. Полтава: ПДАУ, 2023. С. 389–392.
17. David Gourley, Brian Totty, Marjorie Sayer, Anshu Aggarwal, Sailu Reddy. *HTTP: The Definitive Guide*. O`Reilly Media Inc., 2002. 976 p.
18. Zuki Pristianoro Putro, Riza Adrianti Supono. Comparison Analysis of Apache Nginx Webserver Load Balancing on Proxmox VE in Supporting Server Performance. *International Research Journal of Advanced Engineering and Science*. 2022. Volume 7. Issue 3. P. 144–151.
19. Most Popular Backend Frameworks – 2012/2023. *Statistics & Data*. URL: <https://cutt.ly/EwT5vJ0b> (дата звернення: 20.10.2025)..
20. М. А. Демиденко. Введення в сучасні бази даних: навчальний посібник. Дніпро: НТУ «Дніпровська політехніка», 2020. 38 с.
21. Dr. Mukesh Negi. *Fundamental of Database Management System*. BPB Publications, 2019. 175 p.
22. What is a Content Management System (CMS)? *Oracle*. URL: <https://cutt.ly/YwT51UfV> (дата звернення: 20.10.2025).

23. Сучкова В. Що таке CMS?: стаття 24.01.2024. *Shelfy*. URL: <https://shelfy.com.ua/newsroom/shho-take-cms/> (дата звернення: 20.10.2025).
24. CMS Market Share Analysis. *JoostdeWalk*. URL: <https://joost.blog/cms-market-share/> (дата звернення: 20.10.2025).
25. Знайдіть психотерапевта, який вас зрозуміє. *Rozmova*. URL: <https://www.rozmova.me/tests> (дата звернення: 20.10.2025).
26. Як ваше психічне здоров'я? URL: <https://www.mh4u.in.ua/test/yak-vashe-psyhichne-zdorovu/> (дата звернення: 20.10.2025).
27. Доказова психологія для допомоги Українському суспільству! URL: <https://ua-test.com/> (дата звернення: 20.10.2025).
28. Кращі бекенд фреймворки URL: <https://merehead.com/ua/blog/best-backend-frameworks-2024/> (дата звернення: 20.09.2025).
29. Колесніков Д. React: Що таке React? Як почати вивчати React? Основні навички. *Brainlab*. URL: <https://brainlab.com.ua/uk/blog-uk/shho-take-react-js-yak-pochaty-vyvchaty-reakt> (дата звернення: 16.09.2025).
30. Павлов М. Що таке прототипування?: стаття. *Onpage*. URL: <https://onpage.school/prototyping/> (дата звернення: 16.09.2025).
31. i18next documentation. URL: <https://www.i18next.com/> (дата звернення: 20.04.2025).
32. Що таке NPM. *Tutorial.in.Ua*: вебсайт. URL: <https://nodejs.tutorial.in.ua/what-is-npm/> (дата звернення: 04.07.2025).
33. Yarn vs NPM: Which One is Best to Choose? *Knowledgehut*. URL: <https://cutt.ly/BwT5Ob1f> (дата звернення: 04.07.2025).
34. Git – Про систему контролю версій. *Git*. URL: <https://cutt.ly/7wT5OXqs> (дата звернення: 20.10.2025).
35. Beyond Git: The other version control systems developers use. *Stack Overflow Blog*. URL: <https://cutt.ly/WwT5PgF8> (дата звернення: 20.04.2025).
36. Редактори коду. *Введення у веброзробку*. URL: <https://cutt.ly/CwT5AeCy> (дата звернення: 20.10.2025).

37. Documentation for Visual Studio Code. *Visual Studio Code*. URL: <https://code.visualstudio.com/docs> (дата звернення: 20.10.2025)..
38. Google шрифти: Як використовувати Google шрифти. *W3schoolsua*. URL: [https://w3schoolsua.github.io/css/css\\_font\\_google.html](https://w3schoolsua.github.io/css/css_font_google.html) (дата звернення: 20.10.2025).
39. CSS 2D Трансформації. *W3schoolsua*. URL: [https://w3schoolsua.github.io/css/css3\\_2dtransforms.html#gsc.tab=0](https://w3schoolsua.github.io/css/css3_2dtransforms.html#gsc.tab=0) (дата звернення: 20.10.2025).
40. JSON: що це за формат і як з ним працювати. *GOIT*. URL: <https://goit.global/ua/articles/json-shcho-tse-za-format-i-ia-k-z-nym-pratsiuvaty/> (дата звернення: 20.10.2025)..
41. Dall-E Free Your Gateway to Creative Imagining. *DALL-E AI*: вебсайт. URL: <https://www.dall-efree.com/about-us> (дата звернення: 20.10.2025)..
42. Midjourney. URL: [https://www.midjourney.com/explore?tab=top\\_month](https://www.midjourney.com/explore?tab=top_month) (дата звернення: 20.10.2025).
43. Stable Diffusion. URL: [https://stablediffusion.com/#google\\_vignette](https://stablediffusion.com/#google_vignette) (дата звернення: 20.09.2025).
44. Leonardo.Ai. URL: <https://leonardo.ai/> (дата звернення: 20.09.2025).
45. Копішинська О. П., Дузенко О.О., Кіріченко С.Р. Тенденції використання штучного інтелекту в області програмної інженерії. *Modern technologies in education, work and science : The VIII International scientific and practical conference, October 20-22, 2025, Krakow, Poland*. P.179-181. URL: <https://eu-conf.com/wp-content/uploads/2025/08/modern-technologies-in-education-work-and-science.pdf> (дата звернення: 20.10.2025).
46. ChatGPT URL: <https://chatgpt.com/> (дата звернення: 20.09.2025).
47. Дузенко О. Особливості інструментів backend-розробки для забезпечення функцій вебдодатків. *Студентські роботи за науковою тематикою кафедри інформаційних систем та технологій ННІ ЕУП та ІТ*

ПДАУ: XXII щорічний міждисциплінарний семінар, 25 листопада 2025 р., м. Полтава. Полтава: ПДАУ, 2025. С.

48. Посібник користувача калькулятора вартості веб-розробки Stfalcon. URL: <https://stfalcon.com/uk/blog/post/guide-web-development-cost-calculator> (дата звернення: 20.11.2025).

49. Лісовська Л. Методи і моделі оцінювання вартості результатів науково-технічних робіт: переваги та недоліки. Економічний форум. 2023. Том 13. №4. С. 109-117. <https://doi.org/10.36910/6775-2308-8559-2023-4-14>.

50. Графічний дизайнер: середня зарплата в Україні. *Jooble: вебсайт*. URL: <https://surl.li/sbaory> (дата звернення: 20.11.2025).

51. Способи обліку ТЗВ. *I.Factor.UA*. URL: <https://surl.lt/vogdne> (дата звернення: 20.11.2025).

52. Хостинг і реєстрація доменів. URL: <https://www.ukraine.com.ua/> (дата звернення: 22.11.2025).