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 бакалавр

на тему: **«Розроблення інтерактивного вебдодатку на основі сучасних вебтехногій і графічного дизайну в стилі фентезі»**

Виконав: здобувач вищої освіти
за освітньою програмою
Інформаційні управляючі системи
спеціальності 126 Інформаційні
системи та технології
ступеня вищої освіти бакалавр
групи 126ІСТ_бд_2021
Рибка Анастасія Олександрівна
Керівник: Копішинська Олена
Петрівна
Рецензент: Муравльов Володимир
В'ячеславович

Полтава – 2025 року

ВСТУП

Актуальність теми кваліфікаційної роботи полягає в систематизації знань про сучасні вебтехнології та виробленні обґрунтованих критеріїв вибору відповідних інструментів при створенні вебдодатків будь-якого рівня складності та призначення. Технології розвитку веб швидко видозмінюються, трансформуються прагнуть раціоналізації на тлі удосконалення, розподілу функцій, появи нових мов програмування, фреймворків, бібліотек, зрештою, залучення штучного інтелекту. Розуміння та впровадження в роботу новітніх технологій стає критично важливим для досягнення конкурентних переваг. Як відомо, вебдодатки використовуються в абсолютно всіх сферах, де є процес отримання і використання, переробки інформації. Таким чином, постійний пошук шляхів вдосконалення функціональності та зовнішньої привабливості, легкості використання вебдодатків у всіх соціальних, виробничих, наукових та бізнес аспектах, розвинення технологічних навичок веброзробника є об'єктивною необхідністю.

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

Завданнями кваліфікаційної роботи є:

– 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;

– реалізувати технологію вибору мовного інтерфейсу;

– розроблення сучасного інтерактивного вебдодатку для цільової аудиторії та оцінювання економічності обраних методів, технологій.

Об'єктом дослідження кваліфікаційної роботи є вибір та комбінування спеціалізованих вебтехнологій і програмних середовищ для розробки інформаційного 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, порівняльні методи оцінювання, використання штучного інтелекту для синтезу 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 літературні дослідження з досліджуваної проблеми.

Практична значущість роботи полягає в досягненні оригінального поєднання відомих технологій розробки вебдодатків із демонстрацією ефективності особливих розширень, зокрема функцій CSS обробки подій, раціональних комбінацій дизайну користувача, вміле застосування штучного інтелекту.

Апробація результатів дослідження відбувалася шляхом оприлюднення тез доповіді на студентській науково-практичній конференції за підсумками проходження виробничих практик здобувачів вищої освіти спеціальності 126 Інформаційні системи та технології, м. Полтава, ПДАУ, 10 жовтня 2024 р.

Структура та обсяг кваліфікаційної роботи. Пояснювальна записка кваліфікаційної роботи складається зі вступу, 3 розділів, висновків, списку використаних джерел (36 найменувань), додатків. Кваліфікаційна робота містить 7 таблиць, 26 рисунків, викладена на 57 сторінках.

РОЗДІЛ 1
ТЕОРЕТИЧНИЙ ОГЛЯД ТЕХНОЛОГІЙ ДИЗАЙНУ ТА РОЗРОБЛЕННЯ ВЕБДОДАТКІВ

1.1 Стан сучасних вебтехнологій та прикладні аспекти їх застосування

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 [1].

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 [2]. 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 [3].

Потрібно враховувати, що технології 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 є доволі низькою. 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 веброзробці дала змогу створювати елементи, які можна повторно використовувати в різних частинах коду.

Окрім цього, стали з'являтися 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 коннекти, як при використані 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 всеможливий 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: вид діяльності, характеристика, перелік типових програмних рішень. 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 категорій вебсайтів

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.
Інформаційні сайти	Спрямовані на надання користувачам інформації різного характеру: новинні портали, сайти компаній, освітні ресурси ті інші.	Розгалуджена 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 управліннями 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 найкраще використовувати потужні фреймворки.

Окремо важливим моментом є адаптивність. Зараз популярним є підхід «Mobile First». 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. Це важливо, оскільки якщо подивитись на статистику інтернет–трафіку за 2015-2024 рр., то можна побачити, що трохи більше ніж 60% трафіку складають користувачі мобільних пристроїв (рис. 1.1).

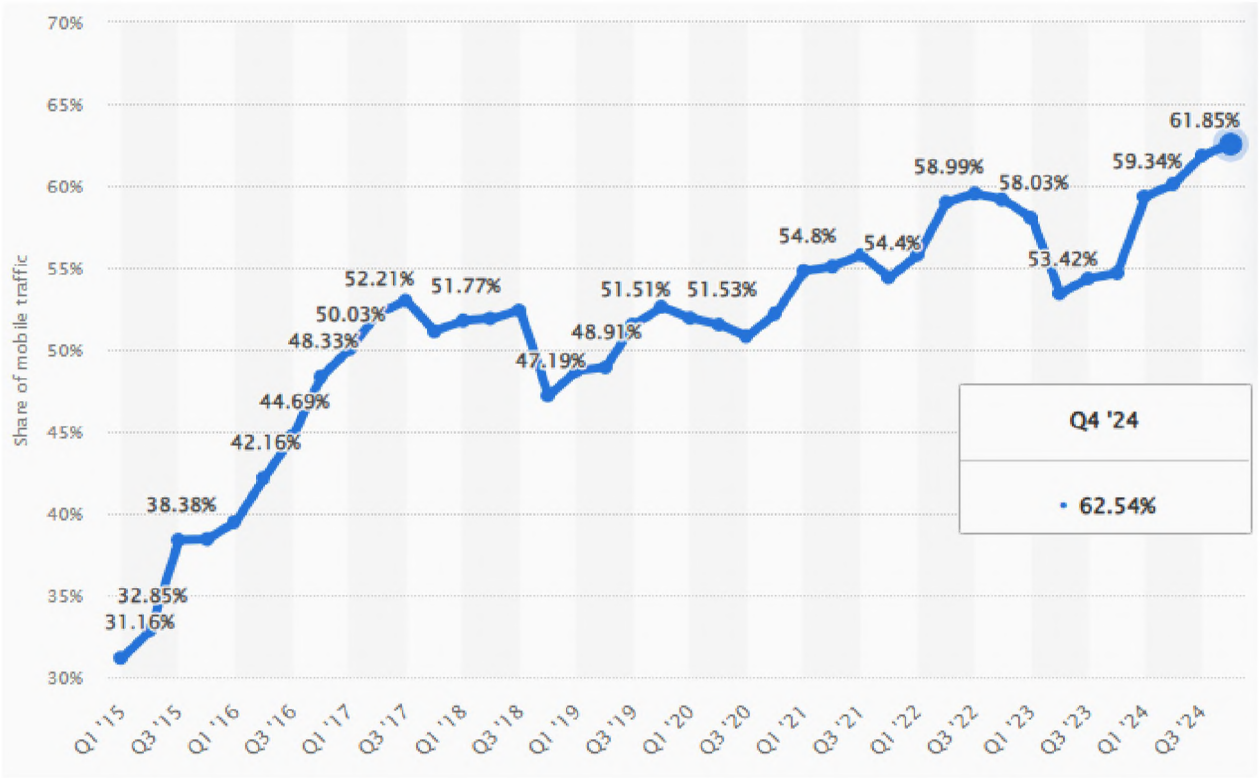


Рисунок 1.1 – Графік глобального використання інтернет-трафіку користувачами мобільних пристроїв за 2015-2024 рр. [8]

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 Склад і характеристика інструментів Frontend- та Backend-розробки для створення 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 фронтенд–частини сайту є лише 1 засіб – HTML. Все інше, включаючи CSS, JavaScript та різні бібліотеки, фреймворки й препроцесори не є обов'язковим з точки зору браузеру, але в сучасному світі без подібних інструментів не створюється жоден (окрім самих простих, коли розробник тільки навчається) 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 HTML–препроцесорів 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].

Оскільки HTML дозволяє лише розміщувати елементи на сторінці, постає необхідність в інструменті, який буде вказувати браузеру як ці об'єкти виглядають. Рішенням цієї проблеми є CSS (Cascading Style Sheets). За допомогою таблиці стилів можна задавати, розмір, колір, положення одних елементів відносно других, поведінку курсора при наведенні та багато інших.

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 [12].

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 [13]. У JavaScript фактично немає конкурентів: 98% вебсайтів використовують саме цю мову. Але альтернативи все ж є: Dart, CoffeeScript, Haxe, 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.

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 проєктах з великою кількістю класів, функцій та змінних. Розробнику, який використовує TypeScript не треба буде витрачати 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) [14].

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 рисунку 1.2.

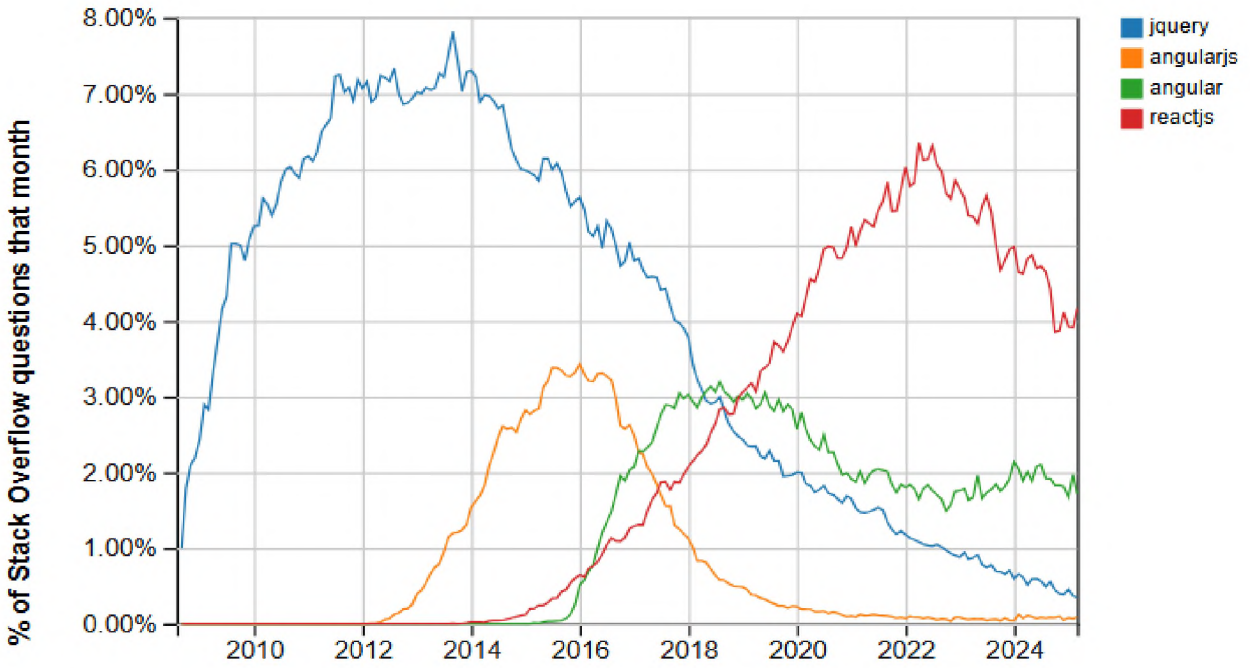


Рисунок 1.2 – Графік популярності фреймворків на Stack Overflow станом на кінець 2024 р. [15]

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.

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 необхідністю додаткова корисна інформація – payload. Графічне зображення класичної взаємодії клієнту (frontend) та серверу (backend) наведено на рис. 1.3.

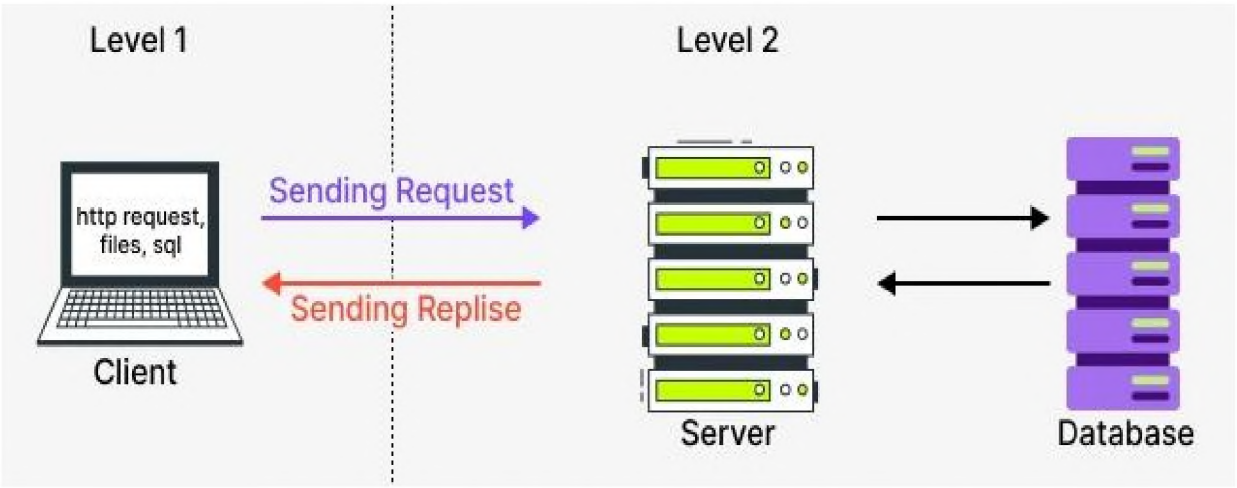


Рисунок 1.3 – Взаємодія клієнта та сервера

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 більшої кількості користувачів [16].

Прикладами таких програмних засобів є Apache та Nginx. Результати дослідження [17] 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.io. 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.

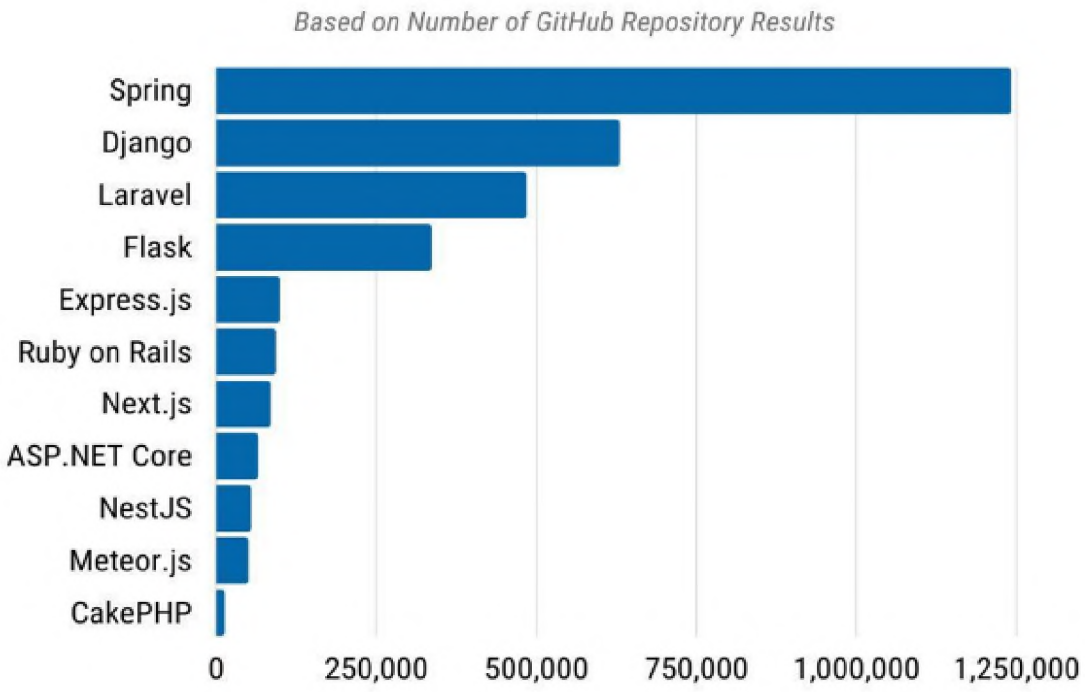


Рисунок 1.4 – Діаграма популярності backend-фреймворків [18]

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», документоорінєтоване 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 [19].

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 [20]. 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 треба розбиратися в програмному коді [21]. 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 (додаток Б).

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 електроного магазину, або система онлайн–оплати покупок.

За типом CMS можна розділити наступним 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 (ECM) – 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.

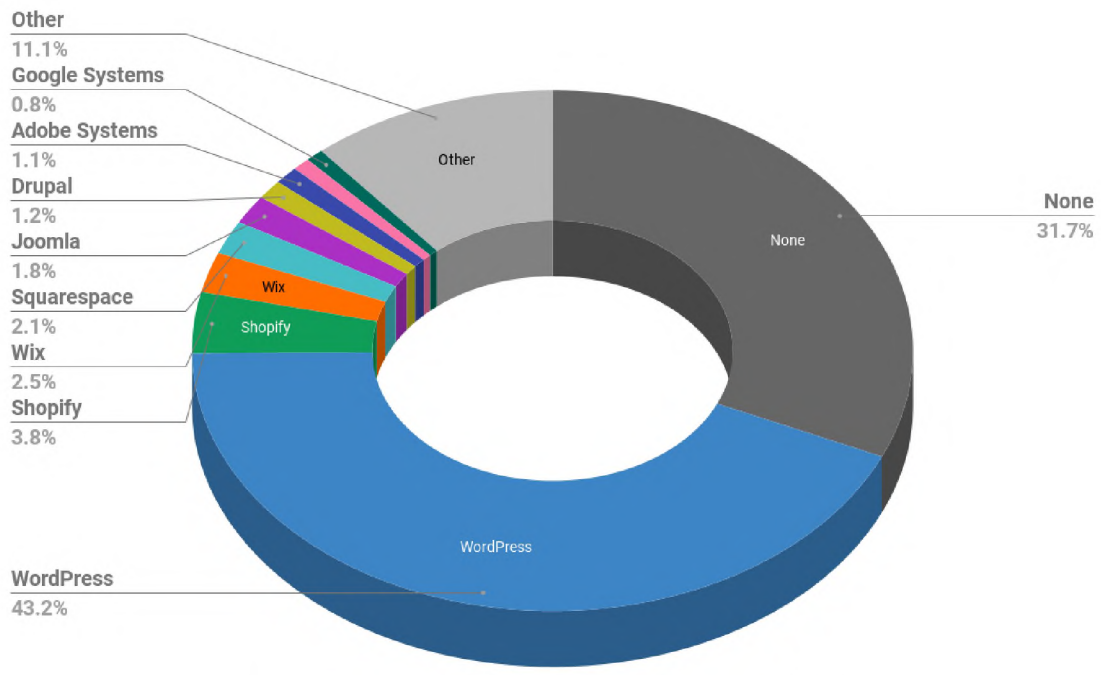


Рисунок 1.5 – Рейтинг популярності CMS [22]

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 розширені. 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 – Порівняльний аналіз Content Management Systems

Переваги	Недоліки
– великий вибір різноманітних CMS; – висока швидкість розробки; – дешевизна; – не потрібно вміти писати код; – простий і зрозумілий інтерфейс; – підтримка одночасної роботи багатьох користувачів; – велика кількість безкоштовних готових шаблонів та рішень; – можливість кастомізації; – гарне тимчасове рішеня, поки повноцінний сайт в розробці; – більшість CMS мають модульну архітектуру, а їх функціональність легко розширюється за допомогою плагинів; – навіть безкоштовні CMS мають «технічну підтримку» у вигляді спільноти користувачів.	– низька адаптивність (flexablilty); – чим специфічніше завдання, тим менше шанс, що CMS підійде для цього; – потрібен час на засвоєння конкретної CMS; – для простих сайтів функціональність CMS, як правило, виявляється надмірною; – може зламатися, якщо адміністратор неправильного його налаштував; – у багатьох CMS є проблеми з безпекою.

Отже, можна зробити висновок, що CMS є чудовими інструментами, якщо їх використовувати в правильному руслі.

РОЗДІЛ 2
АНАЛІЗ ТЕХНОЛОГІЙ ПРОЄКТУВАННЯ ТА КРИТЕРІЇ ВИБОРУ ДЛЯ РОЗРОБКИ ІНФОРМАЦІЙНИХ ВЕБСАЙТІВ

2.1 Аналіз вимог до функцій та структури інтерактивного інформаційного вебдодатку

Практичною частиною кваліфікаційної роботи є проєктування та розробка інтерактивного вебдодатку для взаємодії користувача зі спеціально підібраним контентом, обробкою даних та відповідним дизайном. Основна ідея – розробити вебсайт з соціально-психологічним контентом. Користувач послідовно відповідає на питання, потрапляє на нові сторінки в залежності від наданої 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.

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 структуру майбутнього вебдодатку;

– розробити прототип та протестувати для виявлення складних унктів реалізації;

– підібрати або створити власний сучасний графічний матеріал за тематикою розділів вебдодатку;

– реалізувати вебдодаток із використанням сучасних технологій, використовуючи сучасні середовища розробки та контролю версій, провести тестування.

Дизайн такого сайту повинен бути ергономічний 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. Розраховуємо, що вебдодаток буде багатосторінковим, оскільки планується пройти тест-квест за більше, ніж 5 рівнями.

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 планується розвинути й реалізувати:

– прогрес-бар для показу, скільки вже 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:

– вибір гендеру (впливає або на адаптацію сцен, або просто як частина профілю).

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 Вибір інструментарію проєктування вебдодатку на основі порівняльних характеристик

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, технології backend-розробки.

При виборі фреймворків були використані основні правила, які були вибрані шляхом вивчення досвіду практикуючих вебдизайнерів, зокрема [23]. 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. 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.

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

6. 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.

Розібравши всі тенденції та найкращі фреймворки на 2024 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, будо прийнято рішення використати такі технології, як React та JSON. Вибір має чітке технічне і практичне обґрунтування (табл. 2.1), особливо для інтерактивного психологічного вебтесту.

Таблиця 2.1 – Обґрунтування вибору базових технологій розробки

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, створена 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. VirtualDOM 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 rerendering лише тієї частини DOM–дерева, яка зазнала змін. 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: малоймовірно, 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 розроблено макет ідеї прототипу (рис. 2.1), з якого видно логіку побудови першої та наступних сцен, і перемикач мов інтерфейсу.

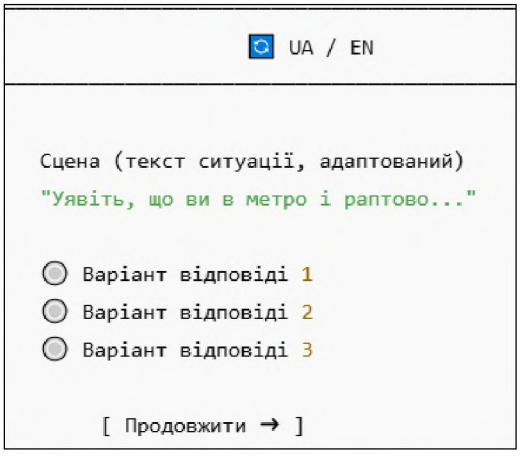


Рисунок 2.1 – Схема прототипу основної ідеї вебдодатку для однієї сцени

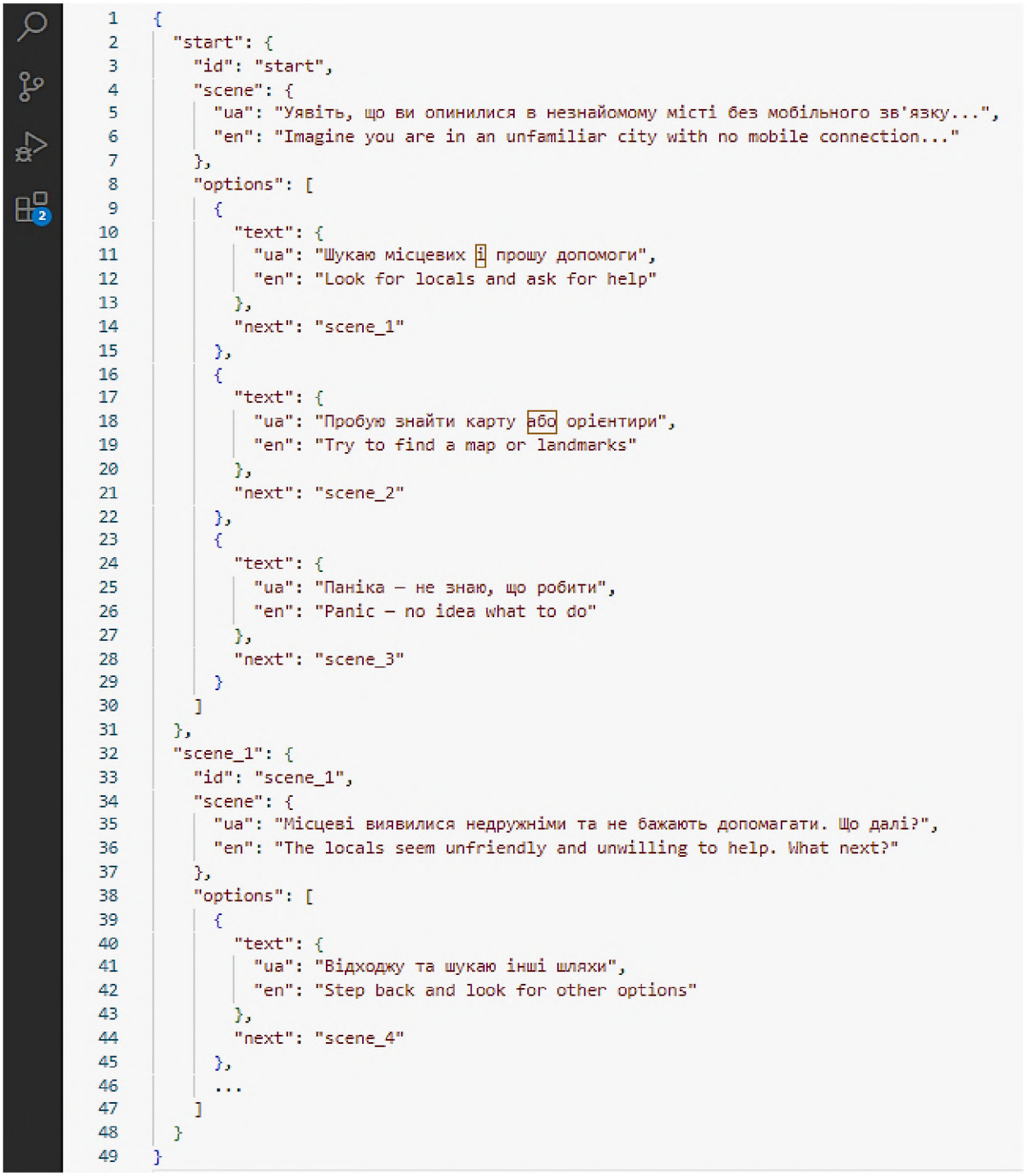
Згідно макету прототипу було необхідно розробити кілька скорочених прикладів з метою перевірки зручності та функціональності інструментарію:

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

– JSON-структуру для логіки питань;

– прототип дизайну в стилі «психологічного тесту».

На рис. 2.2 наведено код прикладу JSON-структури поведінки логіки питань при попередній розробці прототипу окремих частин вебдодатку.



```
{
  "start": {
    "id": "start",
    "scene": {
      "ua": "Уявіть, що ви опинилися в незнайомому місті без мобільного зв'язку...",
      "en": "Imagine you are in an unfamiliar city with no mobile connection..."
    },
    "options": [
      {
        "text": {
          "ua": "Шукаю місцевих і прошу допомоги",
          "en": "Look for locals and ask for help"
        },
        "next": "scene_1"
      },
      {
        "text": {
          "ua": "Пробую знайти карту або орієнтири",
          "en": "Try to find a map or landmarks"
        },
        "next": "scene_2"
      },
      {
        "text": {
          "ua": "Паніка – не знаю, що робити",
          "en": "Panic – no idea what to do"
        },
        "next": "scene_3"
      }
    ]
  },
  "scene_1": {
    "id": "scene_1",
    "scene": {
      "ua": "Місцеві виявилися недружніми та не бажають допомагати. Що далі?",
      "en": "The locals seem unfriendly and unwilling to help. What next?"
    },
    "options": [
      {
        "text": {
          "ua": "Відходжу та шукаю інші шляхи",
          "en": "Step back and look for other options"
        },
        "next": "scene_4"
      },
      ...
    ]
  }
}
```

Рисунок 2.2 – Приклад JSON-структури логіки питань

Ця структура дозволяє описати кожну окрему сцену, її текст можливо подавати двома мовами. Показано варіанти відповідей і до яких сцен вони ведуть. Наступним кроком було створення React-прототипу цієї сцени на основі цього JSON (див. рис. 2.2) – з мовною підтримкою, кнопками, переходами й навіть прогресом.

Початковий код для психологічного тесту на React представлено в додатку А. Інтерактивна візуалізація моделі тестування на різних мовах показана на рис. 2.3-2.4.

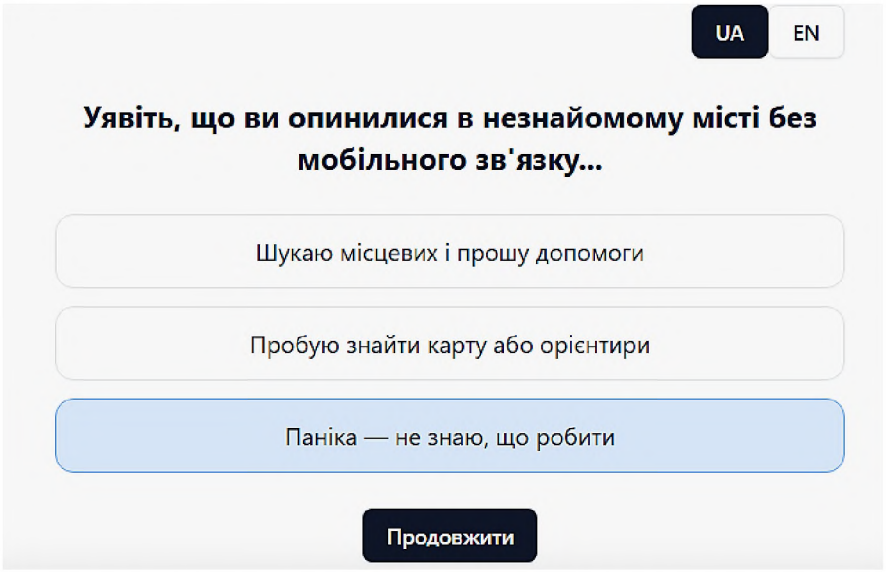


Рисунок 2.3 – Візуалізація сцени з вибором відповіді українською мовою

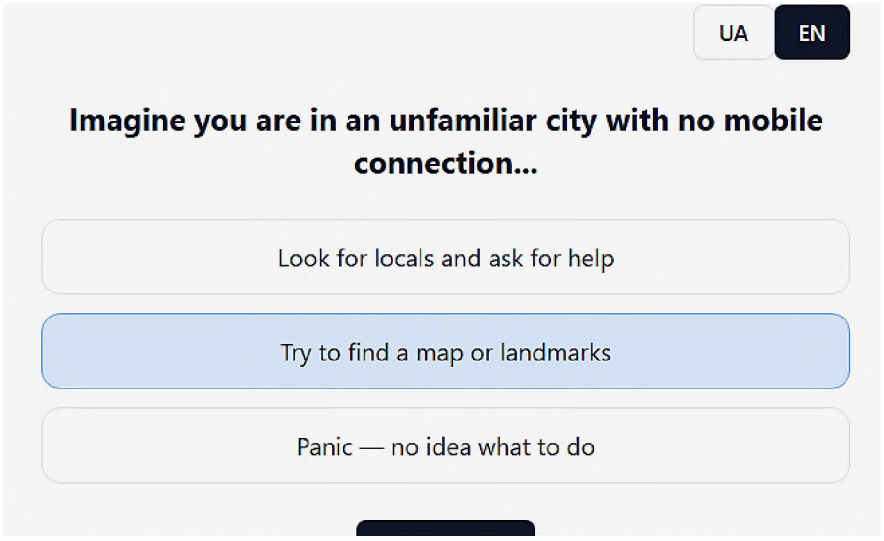


Рисунок 2.4 – Візуалізація прототипу сцени з вибором варіанту англійською мовою

У прикладі прототипу інтерфейсу вебдодатку реалізовано:

– вибір мови (UA/EN);
– відображення сцени та варіантів відповідей;
– логіка переходу між питаннями на основі вибору;
– приклад із однією гілкою переходу;

Обраний стек технологій дозволяє не лише створити інтерактивний приклад на його основі, а також додати більше сцен, прогрес-бар, фінальну оцінку (портрет) на основі виборів. Окрім React, на frontend'і нам знадобляться ще деякі 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 l10n/i18n, вона підтримує стандартні функції бібліотеки i18n, такі як інтерполяція та множина, і добре працює з асинхронними запитами [24]. При правильному налаштуванні вона визначає параметри мови браузера для автоматичного завантаження даних, специфічних для мовних стандартів.

Як і інші бібліотеки JavaScript l10n/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 переконайтеся, що всі змінні збережені. Далі слідує кілька правил.

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 регістру keyname_one та keyname_other.

2.3 Порівняльний аналіз середовищ розробки та контролю версій

Окрім основного інструментарію, 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. Розберемся більш детально з кожним із видів інструментів.

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 [25]. 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 запускі проєкту, створюється ще один файл «lock», в якому знаходиться перелік всіх пакетів, які потрібно встановити [25]. 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 менеджеру. В результаті буде встановлено всі пакети відповідних версій.

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

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 наступні категорії:

– локальні системи контролю версій. Якщо над проєктом працюєте 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 [26];

– 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 виході з ладу жорсткого диску, ви втрачаєте весь проєкт, якщо у вас не було створено резервних копій [26];

– розподілені (децентралізовані) системи контролю версій. Найсучасніший підхід до контролю версій проєкту. При такому підході, учасники отримують не лише останній знімок файлів репозитарія: натомість вони отримують повну копію та всю 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 [26].

Враховуючи особливості різних видів систем, найкращим варіантом на сучасному етапі є саме розподілені системи контролю версій. На рис. 2.5 наведено рейтинг популярності їх використання. Видно, що лідером з величезним відривом є Git.

Рисунок 2.5 – Популярність систем контролю версій [27]

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 [28].

Зазвичай, у кожної мови програмування є власні, спеціалізовані саме під неї IDE. Наприклад для Java це IntelijIDEA 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, томуйого можна зробити повноцінним IDE для будь-якої мови програмування. Але деякі моменти будуть гірші порівняно з IDE. Наприклад, особливо слабким місцем VSC є відстуність 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 для розробки

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 [29]. Наприклад, для виконання завдань знадобляться наступні плагіни:

– ES7+ React/Redux/React Native – плагін, що спрощує роботу з React шляхом надання сніппетів 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 проєктів. До особливостей Git можна віднести:

– збереження абсолютно всієї історію проєкту у вигляді 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.

Таким чином, сформульовано та поділено на логічні кроки задачу практичної частини кваліфікаційної роботи – розроблення сучасного вебдодатку. Поставлено вимоги до дизайну, розроблено майбутню структуру та вимоги до функціоналу вебдодатку.

РОЗДІЛ 3
ОПИС ПРАКТИЧНОЇ ЧАСТИНИ РЕАЛІЗАЦІЇ ВЕБДОДАТКУ

3.1 Програмна реалізація інтерфейсу користувача інтерактивного вебдодатку

Успішне прототипуваня майбутнього вебдодатку і 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 майбутнього вебдодатку. Макет головної сторінки наведено в додатку Б, коди – в додатку В. Дизайн сайту виконаний в мінімалістичному стилі, з двома домінуючими кольорами – фіолетовий (з відттінками) і охра (темно-жовтий), які поєднуються з нейтральними. Лістинги стилів сторінок представлені в додатках Г, Д, Е. 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.

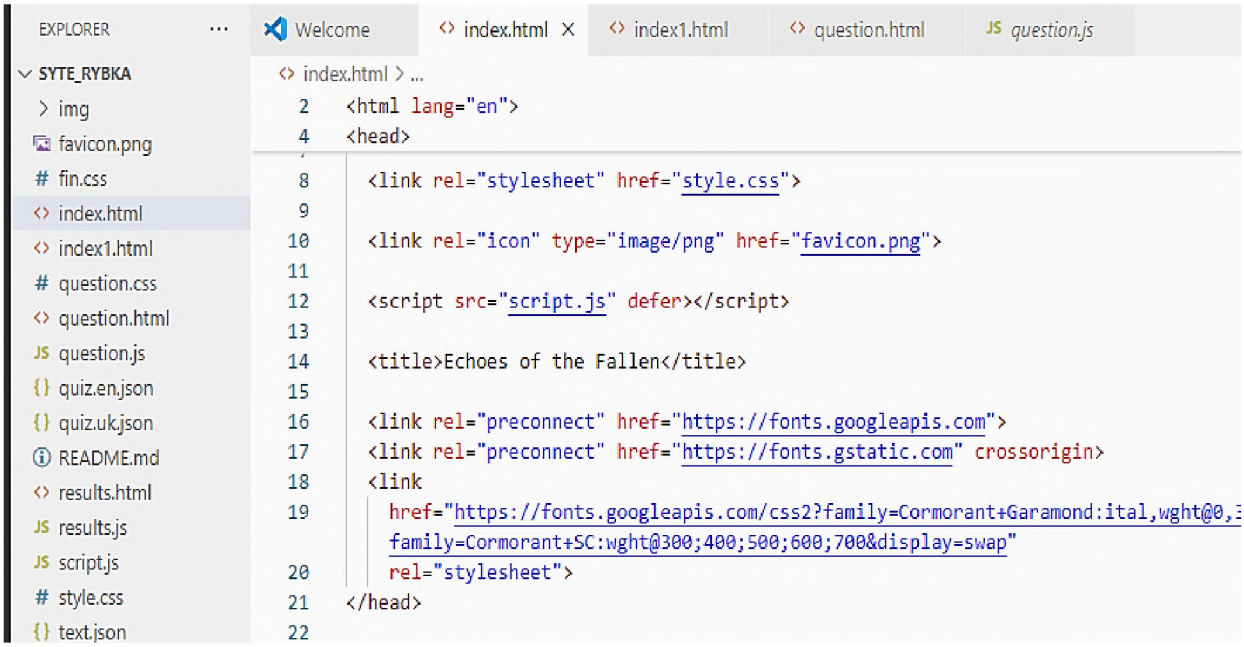


Рисунок 3.1 – Структура всього проєкту та частини `<head> …</head>`

В частині <head> представлено посилання на fonts.googleapis для використання спеціальних вебшрифтів. Основний шлях користувач проходить, послідовно натискаючи на варіанти відповіді, розміщені на кнопках або блокових елементах. Після натискання «Старт» відбувається перехід на першу тематичну сторінку й вибір статі – кнопки «Чоловік» або «Жінка» (рис. 3.2). Наступне вікно – перша сцена і «сюжетний» текст із варіантами відповіді, які представлені в трьох блоках на вибір (рис. 3.3).

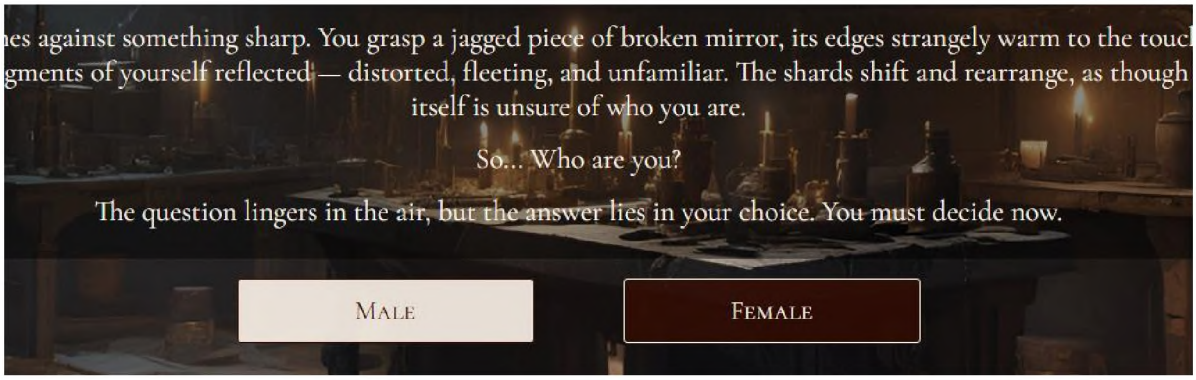


Рисунок 3.2 – Дизайн сторінки з вибором власної гілки згідно гендерної приналежності

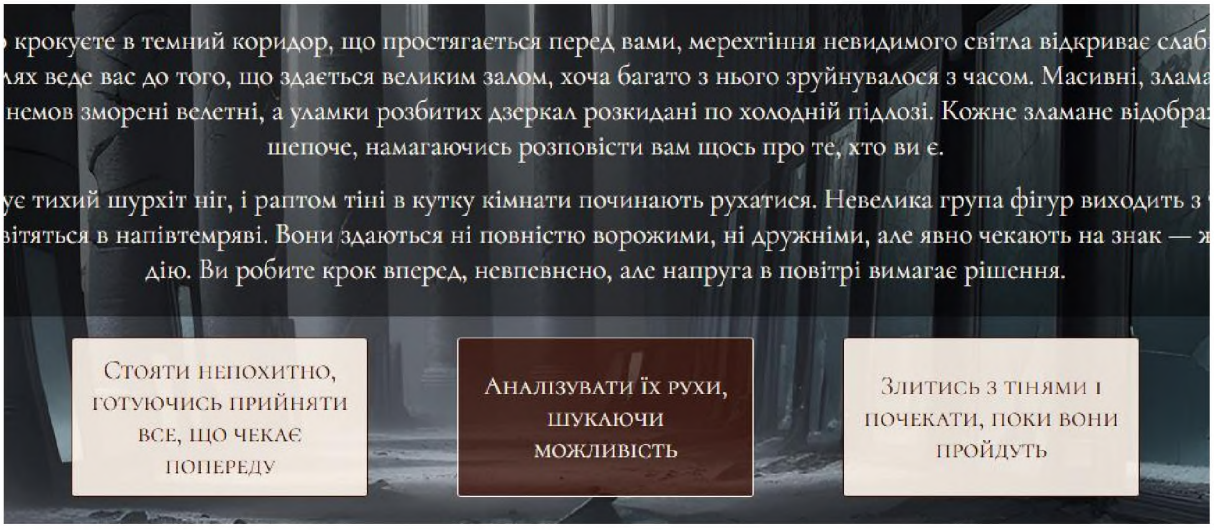


Рисунок 3.3 – Приклад оформлення фрагменту кожної сцени в тест-квесті на вебдодатку з трьома варіантами вибору подальшої власної траєкторії

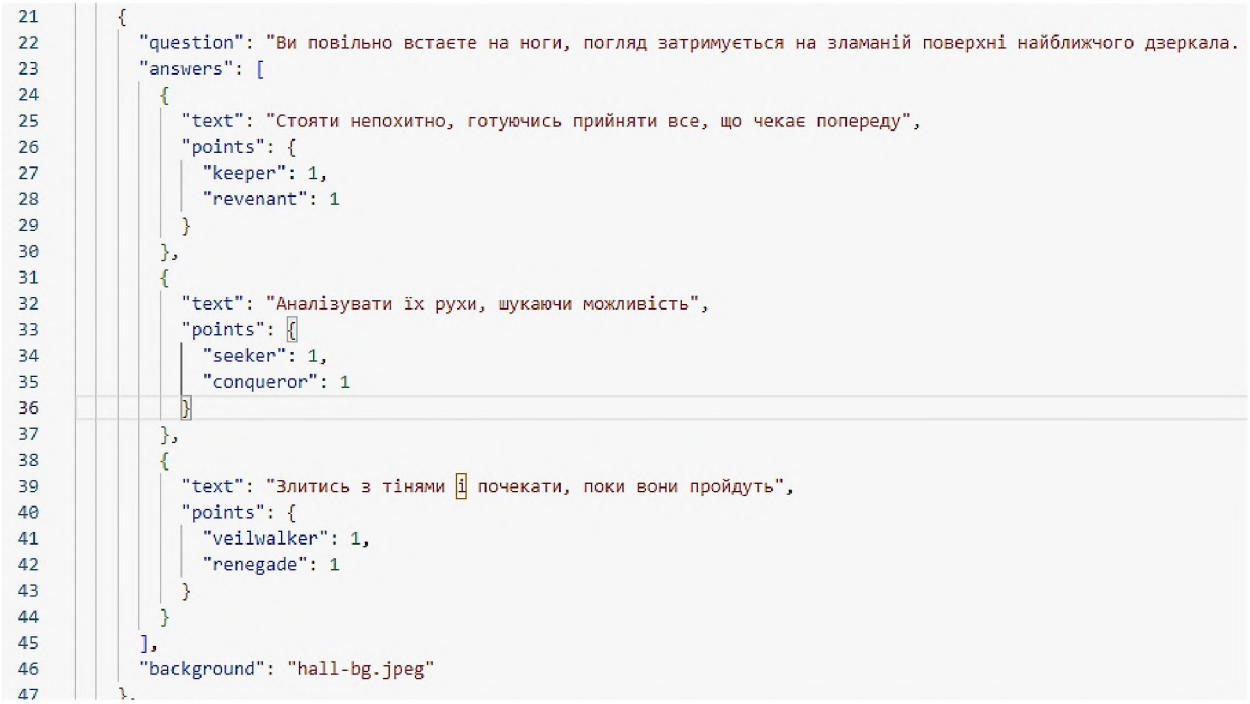
Реалізація сцен і забезпечення вибору користувача були основними технічними задачами, які було вирішено шляхом комплексного застосування

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 Д) для вибору і фіксації відповіді.

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 описані в класах `&:hover` та `&:active.`

Для опису сцен із питаннями і вибором відповідей створені відповідні JASON-файли: варіант українською мовою, аналогічний файл для варіанту англійською мовою. Кожен файл містить 350 рядків коду. Фрагмент опису однієї сцени з одним питанням та врахуванням однієї з трьох відповідей українською мовою показано на рис. 3.5. Таких питань створено 13 і далі слідує перехід до опису результатів – певного психологічного портрету, визначеного шляхом тестування.



```
{
  "question": "Ви повільно встаєте на ноги, погляд затримується на зламаній поверхні найближчого дзеркала.
  "answers": [
    {
      "text": "Стояти непохитно, готуючись прийняти все, що чекає попереду",
      "points": {
        "keeper": 1,
        "revenant": 1
      }
    },
    {
      "text": "Аналізувати їх рухи, шукаючи можливість",
      "points": {
        "seeker": 1,
        "conqueror": 1
      }
    },
    {
      "text": "Злитись з тінями і почекати, поки вони пройдуть",
      "points": {
        "veilwalker": 1,
        "renegade": 1
      }
    }
  ],
  "background": "hall-bg.jpeg"
},
```

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

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

```
btnContainer.onclick = handleClick;

continueStartedQuiz();

getQuestions().then(startQuiz);

function continueStartedQuiz() {
  //перевірка, чи існує індекс поточного питання та дані про відповіді
  const index = localStorage.getItem('index');
  const storedAnswers = localStorage.getItem('answers');

  //якщо є збережений індекс, оновлюємо поточне питання
  if (index) {
    currentQuestion = +index;

    if (currentQuestion > 12) {
      location.href = "results.html";
    }
  }
  if (storedAnswers) {
    answers = JSON.parse(storedAnswers);
  }
```

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.

```
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

Варіанти відповідей розглядаються як масиви, кожний ключ обробляється в циклі, визначаємо індекс поточної відповіді (рядок 81), нараховуємо бали для кожної відповіді (рядок 82, рис. 3.7).

Далі оновлюємо відповіді для кожного ключа (рядки 85-86) і зберігаємо прогрес (рядок 90).

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 оисаної 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 (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 штучного інтелекту в розробленні графічних об'єктів 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) [30] 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 [31]: 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 [32]	+ 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 [33]: підходить для: тих, хто хоче зробити фентезі графіку для ігор, книг, UI, без технічного клопоту	+ Онлайнова 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. 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 [34], 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.

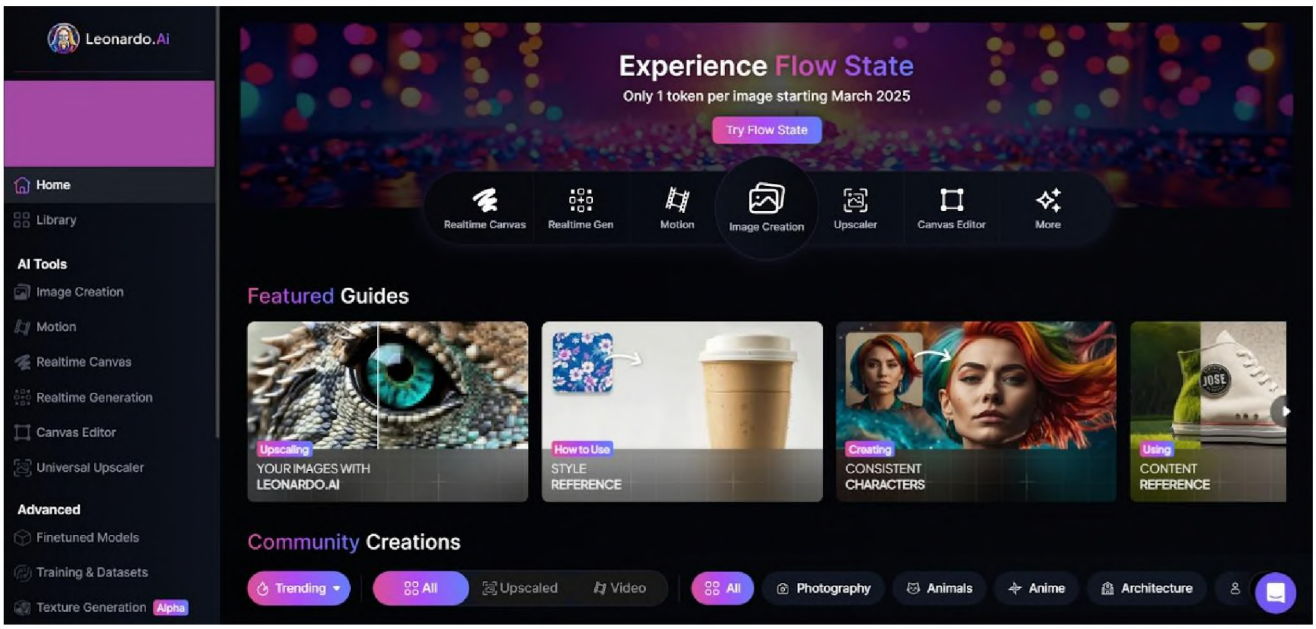


Рис. 3.10 – Головна сторінка Leonardo.Ai

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 сформувати 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.

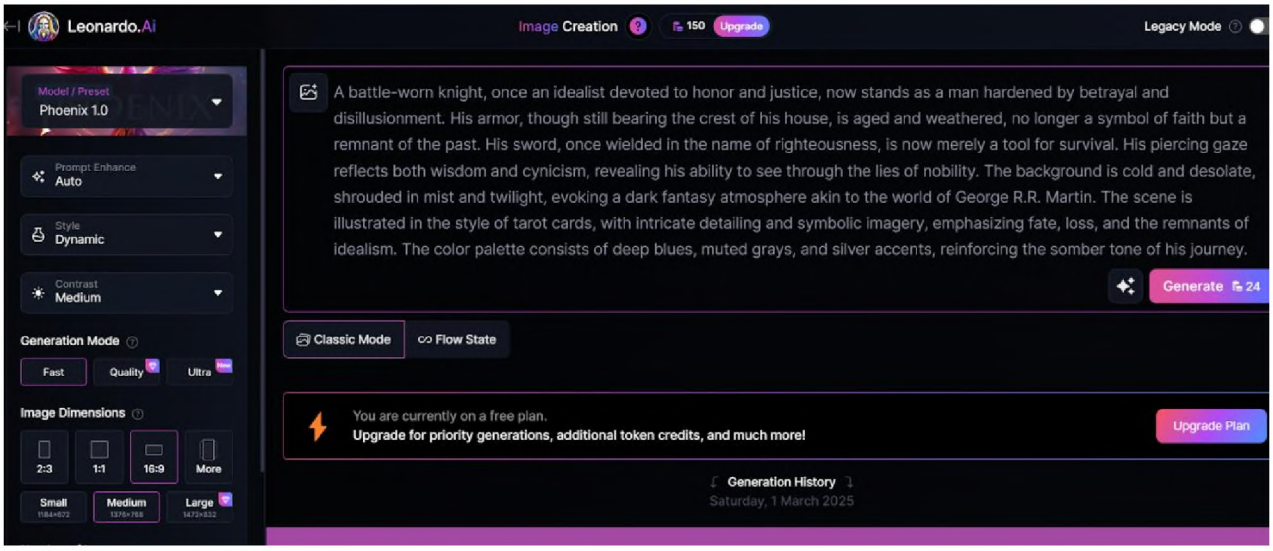


Рисунок 3.11 – Меню створення зображення та уведений готовий промпт

Про моделі (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.

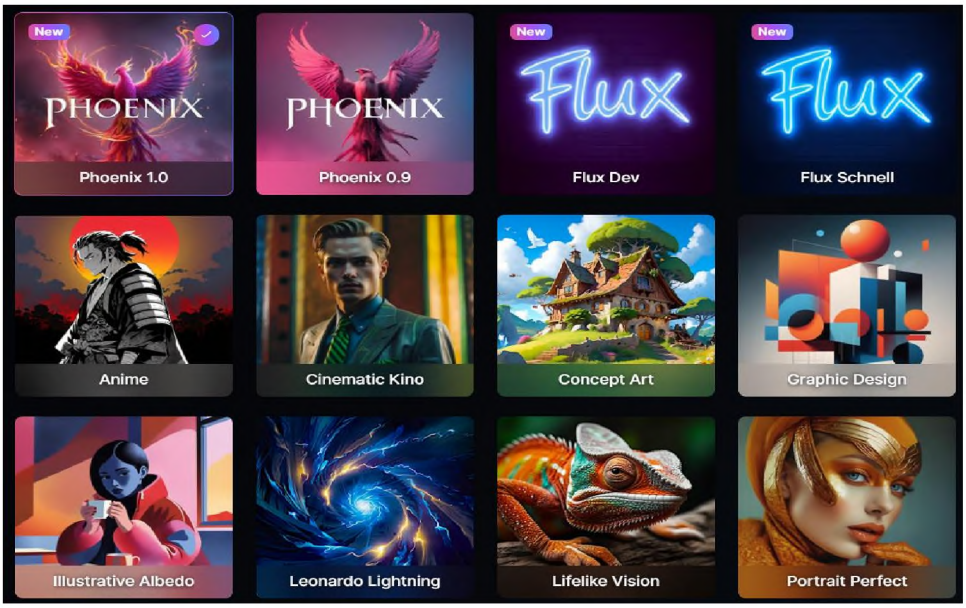


Рисунок 3.12 – Доступні моделі на Leonardo.Ai

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.

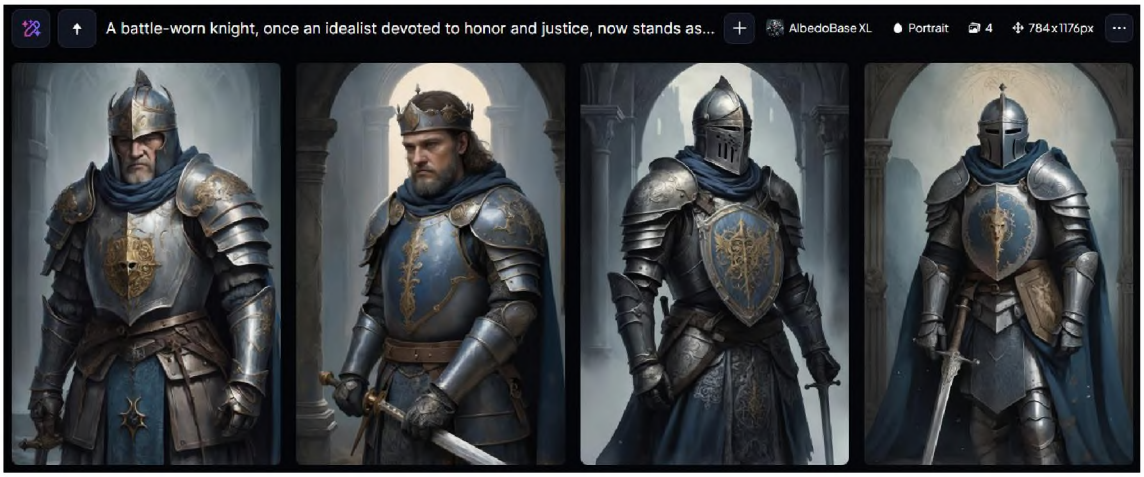


Рисунок 3.13 – Створені зображення за різними налаштуваннями

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.

3.3 Економічне оцінювання вартості розробки сучасного вебдодатку

У ході кваліфікаційної роботи було створено інформаційний інтерактивний вебдодаток. Такий сайт не буде приносити прямого прибутку, але очікується, що буде відвідуваним. Головною цінністю є цікавий корнтент на актуальну тематику. Його можна використати як портфоліо роообника або як промо-сайт приватного бізнесу.

Враховуючи, що більшість сайтів все ж створюються з метою отримання прибутку, проведемо економічний аналіз, який включатиме: витрати на дизайн, розробку та різноманітні побічні витрати.

Економічний аналіз – це процес вивчення та оцінки економічних аспектів явищ, процесів, рішень чи проєктів з метою зрозуміння їхньої ефективності, вартості, можливих вигід чи ризиків. У контексті розробки вебдодатків це включає в себе [35]:

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

Для обґрунтування доцільності розробки та впровадження проєктного 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 проєкту займе 2 тижня, тобто 10 робочих днів.

Працівники будуть працювати по стандартному 40-годиному графіку (8 годин на день). Використаємо дані про погодинну оплату графічного дизайнера [36] (використаємо нижню границю) – 121 грн, програміста – 170 грн, тестувальника – 220 грн.

Формула розрахунку заробітньої плати:

$$З_п = (К_{днів} \times В_{роб}) \times Т_г, \quad (3.1)$$

де $К_{днів}$ – кількість робочих днів;

$В_{роб}$ – тривалість робочого дня в годинах;

$Т_г$ – годинна ставка розробника.

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

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

Витрати на придбання ($В_{пр}$) в нашому випадку становитимуть: USB–накопичувач – 200 грн, папір офісний (1 упаковка, 250 аркушів) – 190 грн, безлімітний інтернет на місяць – 250 грн. Всього 640 грн.

Визначення накладних транспортних. Транспортні витрати ($В_{тр}$) для розрахунків даного типу становлять 11% від суми витрат на придбання товарів.

$$В_{тр} = В_{пр} \times 0{,}11, \quad (3.2)$$

де $В_{пр}$ – витрати на придбання.

Розрахунок: $В_{тр} = 640 \times 0,11 = 70,4$ грн.

Визначення накладних витрат. Накладні витрати ($В_н$) проєктних організацій складають 25% витрат на оплату праці.

$$В_н = (З_{диз} + З_{прог} + З_{тест}) \times 0,25, \quad (3.3)$$

де $З_{диз}$ – зарплата дизайнера;

$З_{прог}$ – зарплата програміста;

$З_{тест}$ – зарплата тестувальника.

Розрахунок: $В_н = (9680 + 13600 + 17600) \times 0,25 = 10220$ грн.

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

$$В_{ін} = (З_{диз} + З_{прог} + З_{тест}) \times 0,06. \quad (3.4)$$

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

Загальна сума витрат становить:

$$В_{сум} = (З_{диз} + З_{прог} + З_{тест}) + В_{тр} + В_н + В_{ін}. \quad (3.5)$$

Розрахунок: $В_{сум} = (9680 + 13600 + 17600) + 70,40 + 10220 + 2452,8 =$
$= 53623,2$ грн.

Окремо варто розглядати витрати на хостинг. Не кожен хостинг наразі підтримує розміщення Node.js додатків. Гарним рішенням з підтримкою необхідного інструментарію є платформа ukraine.com.ua.

Найдешевший варіант нам не підходить, оскільки нам потрібно мінімум 2 сайти (один для React–додатку, інший для Node.js сервера). Кращий варіант коштує 285 гривень на місяць 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 407 гривень надає більше пам'яті та можливість створювати безмежну кількість сайтів, але в нашому випадку це непотрібно. Зупинимось на варіанті за 285 гривень на місяць.

Таким чином, остаточний підрахунок сумарних витрат складе 54030,2 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

У ході виконання кваліфікаційної роботи було проаналізовано велику 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 розглядались системи керування контентом (CMS): досліджено класифікацію, особливості застосування та галузі використання, для яких вони призначені.

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

У прикладній частині роботи був спроєктований та розроблений інтерактивний вебдодаток із можливістю переключення мовного інтерфейсу, обробки відповідей користувача при проходженні спеціального тесту, передавання відповідей на серверну частину, збереження відповідей. Запорукою успішного завершення роботи і виконання завдань було планування й дотримання наступних вимог:

1. Обрання робочого стеку технологій. Для вибору найоптимальніших інструментів проведено ретельний аналіз переваг та недоліків різних продуктів. В результаті було сформовано наступний стек: окрім традиційних frontend–технологій HTML&CSS, JavaScript, були використані можливості фреймворку React, Visual Studio Code – редактор коду, обробка сцен в JASON, технологію збереження даних в LocalStorage, backend–фреймворк Node.js, Git – система контролю версій, GitHub – хмарний репозиторій.

2. Застосування методу попереднього прототипування вебдодатку з огляду на специфіку його призначення і використання. Це дало можливість 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.

Проведено економічний аналіз. Орієнтовна вартість розробки та хостингу такого проєкту разом із накладними витратами становить 54030,2 грн при роботі 3-х спеціалістів протягом 10 днів. Досліджено варіанти розміщення вебсайту на хостингу. Зроблено вибір на користь платформи ukraine.com.ua та тарифу «Кращий» за 407 грн. на місяць.

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

Подібний додаток може бути частиною більш масштабного проєкту будь-якої системи забезпечення послуг, де є проходження поетапного опитування й виходу на індивідуальний результат. Наприклад, психологічна служба, юридична компанія або є вибір різних опцій дизайну, будування тощо. В такому сенсі розробка може отримати практичне застосування.