МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«ДОСЛІДЖЕННЯ ВИКОРИСТАННЯ ШТУЧНОГО ІНТЕЛЕКТУ ПРИ РОЗРОБЦІ КЛІЄНТСЬКОГО ІНТЕРФЕЙСУ ВЕБ-ДОДАТКІВ»**

Виконав: студент 2 курсу, групи 8.1229
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)
освітньої програми Комп'ютерні науки
(назва освітньої програми)
А.А. Каковкін
(ініціали та прізвище)
Керівник доцент кафедри комп'ютерних наук, доцент, к.т.н. Решевська К.С.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент завідувач кафедри програмної інженерії, доцент, к.ф.-м.н. Лісняк А.О
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра комп'ютерних наук

Рівень вищої освіти магістр

Спеціальність 122 Комп'ютерні науки
(шифр і назва)

Освітня програма Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук,
д.т.н., професор

_____ Шило Г.М.
(підпис)

" 5 " березня 2024 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Каковкіну Артему Андрійовичу
(прізвище, ім'я та по-батькові)

1. Тема роботи Дослідження використання штучного інтелекту при розробці клієнтського інтерфейсу веб-додатків

керівник роботи Решевська Катерина Сергіївна, доцент, к.т.н.
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 01 » березня 2024 року № 385-с

2. Строк подання студентом роботи 28.11.2024

3. Вихідні дані до роботи 1. Постановка задачі.
2. Аналіз предметної області
3. Реалізація проектів

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Аналіз предметної області.
2. Основні теоретичні відомості.
3. Реалізація веб-додатків для тестування
4. Експериментальне порівняння та аналіз

5. Перелік графічного матеріалу
Презентація

. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1			
2			
3			

7. Дата видачі завдання 04.03.2024

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	10.09.2024	
2.	Збір вихідних даних.	10.09.2024	
3.	Обробка методичних та теоретичних	05.10.2024	
	джерел.		
	Розробка веб-сайту	7.10.2024	
4.	Розробка першого та другого розділу.	10.11.2024	
5.	Розробка третього розділу.	15.11.2024	
6.	Розробка четвертого розділу	16.11.2024	
7.	Оформлення та нормоконтроль	20.11.2024	
	кваліфікаційної роботи бакалавра.		
8.	Захист кваліфікаційної роботи.	12.12.2024	

Студент _____ А.А. Каковкін
(підпис) (ініціали та прізвище)

Керівник роботи _____ К.С. Решевська
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.Г.Спиця
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Дослідження використання штучного інтелекту при розробці клієнтського інтерфейсу веб-додатків»: 77 с., 9 рис., 1 табл., 9 джерел, 1 додатків.

АВТОМАТИЗАЦІЯ, АДАПТИВНІСТЬ, ВЕБ-ДОДАТОК, ВЗАЄМОДІЯ, ДИЗАЙН, ЕФЕКТИВНІСТЬ, ЗРУЧНІСТЬ, МАШИННЕ НАВЧАННЯ, ПЕРСОНАЛІЗАЦІЯ, ТЕСТУВАННЯ, ШТУЧНИЙ ІНТЕЛЕКТ

Об’єкт дослідження – використання штучного інтелекту при розробці клієнтського інтерфейсу веб-додатків.

Мета роботи: дослідити використання ШІ при розробці клієнтського інтерфейсу веб-додатків.

Метод дослідження – порівняння.

У межах дослідження було розроблено чотири веб-додатки: два з використанням традиційних підходів та два з інтеграцією штучного інтелекту. Проекти орієнтовані на різні завдання, такі як надання мотиваційних порад і рекомендацію книг, що дозволило оцінити ефективність різних підходів до реалізації клієнтських інтерфейсів. Для аналізу було інтегровано функцію вимірювання часу виконання запитів, що дозволило оцінити швидкість обробки даних у традиційних додатках та додатках з інтеграцією ШІ.

Традиційні додатки продемонстрували швидку реакцію системи завдяки використанню заздалегідь визначених даних і мінімальній обчислювальній складності. Натомість додатки з ШІ забезпечили адаптивність і персоналізацію, використовуючи моделі ШІ для аналізу і формування відповідей, але потребували більше ресурсів і часу для обробки запитів.

Таким чином, проведене дослідження стало важливим етапом у вивченні можливостей ШІ у веб-розробці та показало потенціал інноваційних підходів.

SUMMARY

Master's thesis “Study of the use of artificial intelligence in the development of the client interface of web applications”: 77 p., 9 figs., 1 table, 9 sources, 1 appendices.

AUTOMATION, ADAPTABILITY, WEB APPLICATION, INTERACTION, DESIGN, EFFICIENCY, USABILITY, MACHINE LEARNING, PERSONALIZATION, TESTING, ARTIFICIAL INTELLIGENCE

Object of study – The use of artificial intelligence in the development of the client-side interface of web applications.

Purpose: to investigate the use of AI in the development of the client interface of web applications.

Research method: comparison

As part of the study, four web applications were developed: two using traditional approaches and two with the integration of artificial intelligence. The projects are focused on different tasks, such as providing motivational advice and recommending books, which made it possible to evaluate the effectiveness of different approaches to implementing client interfaces. For the analysis, the function of measuring the query execution time was integrated, which allowed us to evaluate the data processing speed in traditional applications and applications with AI integration.

Traditional applications demonstrated fast system response due to the use of predefined data and minimal computational complexity. Instead, AI applications provided adaptability and personalization by using AI models to analyze and generate responses, but required more resources and time to process requests.

Thus, the study was an important step in exploring the possibilities of AI in web development and showed the potential of innovative approaches.

ЗМІСТ

Завдання на кваліфікаційну роботу 2
Реферат 4
Summary 5
Вступ 8
1 Аналіз предметної області 10
1.1 Аналіз використання штучного інтелекту в розробці клієнтських інтерфейсів веб-додатків 10
1.2 Огляд існуючих рішень та інструментів для автоматизації UI/UX дизайну 12
1.3 Вибір методів порівняння ефективності AI у створенні інтерфейсів 14
1.4 Висновки аналізу предметної області 16
2 Розробка методології дослідження 18
2.1 Визначення критеріїв оцінки ефективності клієнтських інтерфейсів 18
2.2 Побудова архітектури веб-додатків для тестування AI-підходів 21
2.3 Обґрунтування використання методів порівняння 24
2.4 Висновки з 2 розділу 27
3 Реалізація веб-додатків для тестування 29
3.1 Розробка традиційного клієнтського інтерфейсу 29
3.1.1 Project Book 29
3.1.2 Tipsday 35
3.2 Інтеграція штучного інтелекту в процес розробки веб-додатку 39
3.2.1 Project_AI 39
3.2.2 TipsAI 42
3.3Налаштування інструментів для порівняння проектів 45
4.Експериментальне порівняння та аналіз 46
4.1 Проведення тестування часу обробки запитів веб-додатків 46
4.2 Порівняння отриманих результатів за визначеними критеріям 47

Висновки 49
Перелик посилань..... 51
Додаток А Project Book..... 52
Додаток Б Project_AI..... 66

ВСТУП

Інтернет та веб-додатки стали неодмінною частиною сучасного життя, що породжує постійне зростання очікувань користувачів стосовно зручності та ефективності взаємодії з цими додатками. Щоб задовольнити ці вимоги, розробники веб-додатків повинні постійно шукати нові підходи до проектування клієнтських інтерфейсів. Одним з найбільш перспективних напрямків у цьому відношенні є використання штучного інтелекту (ШІ), що дає можливість створювати інтерфейси, адаптовані до індивідуальних потреб користувачів та забезпечувати високий рівень персоналізації.

Деякі визначення штучного інтелекту та їх автори:

"Штучний інтелект – це наука та інженерія створення інтелектуальних машин, зокрема інтелектуальних комп'ютерних програм", – Джон Маккарті, один із засновників галузі штучного інтелекту.

"Штучний інтелект – це спроба надати машинах здатність розв'язувати проблеми, що зазвичай вимагають інтелекту людини", – Алан Тюрінг, математик та піонер в області комп'ютерів та штучного інтелекту.

"Штучний інтелект – це створення програм, що роблять так, ніби вони розуміють, навіть якщо це не так", – Роджер Шенкс, дослідник та практик у галузі штучного інтелекту.

"Штучний інтелект – це наука про роботу програм, які, за всіма звичайними умовами, вважаються розумовою, якщо люди роблять їх такими", – Марвін Мінскі, один із основоположників галузі штучного інтелекту.

Ці визначення демонструють різноманітні підходи та погляди на сутність штучного інтелекту від різних вчених та фахівців[1].

Традиційні методи проектування інтерфейсів не завжди забезпечують відповідь на сучасні вимоги користувачів. Недостатня персоналізація, та неадаптованість до їхніх потреб – основні проблеми, з якими стикаються розробники веб-додатків. У багатьох веб-додатках інтерфейс пропонує

однаковий функціонал та контент для всіх користувачів, незалежно від їхніх індивідуальних потреб та вподобань. Це призводить до недооцінки індивідуальних потреб користувачів і може знизити їхню задоволеність від використання додатка. Іншою проблемою є складнощі у взаємодії з інтерфейсом. Багато веб-додатків мають складні навігаційні системи та запутані структури, що ускладнює процес пошуку необхідної інформації для користувачів і може призвести до їхнього розчарування та втрати інтересу до додатка. Крім того, неефективне використання ресурсів є ще однією серйозною проблемою. Деякі веб-додатки можуть вимагати від користувачів великої кількості часу та зусиль для досягнення своїх цілей. Це може призвести до втрати інтересу та відсутності взаємодії з додатком, що негативно впливає на його успішність та ефективність.

За останні роки штучний інтелект досить суттєво змінився, розвинувши нові методи та алгоритми, які виявляються дуже ефективними у різних сферах життя, включаючи розробку веб-додатків. Використання методів машинного навчання, обробки природної мови та комп'ютерного зору дозволяє створювати інтерфейси, що адаптуються до індивідуальних потреб користувачів та реагують на їхні дії миттєво. Нинішня технологія не призначена для написання цілого проекту з нуля – вона покликана доповнити програмістів і трохи полегшити їхню роботу [2].

Мета цієї роботи – дослідити використання ШІ при розробці клієнтського інтерфейсу веб-додатків. Для її досягнення було поставлено такі завдання:

– розробити веб-додатки із використанням традиційних підходів та ШІ;

– провести аналіз ефективності інтерфейсів обох типів;

– впровадити інструменти для оцінки часу виконання запитів;

– зробити висновки щодо перспективності інтеграції ШІ у розробку клієнтських інтерфейсів.

1 АНАЛІЗ ПЕРДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз використання штучного інтелекту в розробці клієнтських інтерфейсів веб-додатків

Штучний інтелект (ШІ) відіграє дедалі важливішу роль у процесі створення клієнтських інтерфейсів для веб-додатків. Традиційний процес розробки інтерфейсів зазвичай передбачає значні витрати часу та людських ресурсів, оскільки він включає проектування, тестування, збір відгуків від користувачів та численні ітерації. Використання ШІ дозволяє автоматизувати трудомісткі процеси дизайну, підвищувати адаптивність інтерфейсу до потреб користувачів та забезпечувати персоналізоване взаємодію.

Сфери застосування штучного інтелекту в UI/UX

Одним із найважливіших аспектів використання ШІ є автоматичне створення дизайну. Сучасні інструменти, такі як Wix ADI (Artificial Design Intelligence), здатні генерувати макети веб-додатків на основі початкових даних. Цей підхід дозволяє швидко створювати функціональні та естетично привабливі рішення, що відповідають запитам клієнтів. ШІ аналізує вимоги користувача, такі як тематика веб-додатка чи бажаний стиль, і пропонує оптимальну структуру сторінок та оформлення.

Ще одним напрямком є персоналізація інтерфейсів. Алгоритми ШІ, які навчаються на великих обсягах даних, дозволяють адаптувати веб-додатки до індивідуальних потреб користувачів. Наприклад, аналізуючи поведінку користувача (відвідувані сторінки, частоту використання певних функцій, навіть емоційні реакції), системи можуть автоматично змінювати структуру або контент сторінок. Це підвищує зручність використання та створює позитивний користувацький досвід.

Важливим аспектом є аналіз поведінки користувачів. Інструменти на базі ШІ можуть виявляти слабкі місця інтерфейсу, використовуючи дані про

взаємодію з додатком. Наприклад, теплові карти кліків або трекінг руху миші дозволяють зрозуміти, які елементи дизайну привертають увагу, а які залишаються непоміченими. Це дає змогу швидко вносити зміни, які оптимізують інтерфейс.

Окрім цього, ШІ активно використовується для автоматизації тестування. Наприклад, моделі прогнозування уваги дозволяють оцінювати зони екрану, які потенційно будуть привертати увагу користувача. Це скорочує час на ручне тестування та допомагає визначати ефективність дизайну на початкових етапах його розробки.

Переваги використання ШІ

Застосування ШІ у розробці інтерфейсів надає низку переваг. По-перше, значно скорочується час розробки. Процеси, які раніше виконували вручну, тепер автоматизуються, дозволяючи дизайнерам зосередитися на творчих аспектах проекту. По-друге, зростає рівень адаптивності інтерфейсів, що робить їх більш гнучкими до потреб різних аудиторій. Нарешті, ШІ сприяє впровадженню інновацій, таких як автоматичне оновлення контенту або динамічна зміна дизайну залежно від контексту використання.

Виклики впровадження

Водночас використання ШІ пов’язане з певними труднощами. Основною з них є висока вартість впровадження, оскільки навчання моделей і підтримка інфраструктури вимагають значних фінансових ресурсів. Іншою проблемою є конфіденційність даних, оскільки для роботи ШІ необхідно збирати та аналізувати інформацію про користувачів. Також результативність ШІ значною мірою залежить від якості даних, які використовуються для навчання, що створює додаткові виклики для розробників.

1.2 Огляд існуючих рішень та інструментів для автоматизації UI/UX дизайну

Сучасні технології значно полегшують процес створення інтерфейсів користувача (UI/UX) завдяки впровадженню штучного інтелекту (ШІ). На ринку вже існує низка інструментів, що використовують ШІ для автоматизації окремих етапів дизайну або створення повноцінних веб-додатків. Аналіз цих інструментів дозволяє оцінити їх переваги та обмеження, а також зрозуміти їхню роль у сучасному процесі розробки інтерфейсів.

Основні інструменти для автоматизації UI/UX

Одним із популярних рішень є Wix ADI (Artificial Design Intelligence). Цей інструмент автоматично створює веб-сайти на основі відповідей користувача на низку питань. Wix ADI аналізує вхідні дані, такі як сфера діяльності, бажаний стиль та функціональність, і генерує сайт із готовим дизайном і базовими функціями. Цей підхід значно скорочує час на розробку, але обмежує можливості тонкого налаштування для досвідчених дизайнерів.

Ще одним прикладом є Uizard, платформа для швидкого створення прототипів. Uizard дозволяє перетворювати прості скетчі або малюнки інтерфейсів у інтерактивні макети, використовуючи алгоритми ШІ. Це корисно для дизайнерів, які працюють із концептуальними ідеями та хочуть швидко візуалізувати свої рішення.

Серед вузькоспеціалізованих інструментів варто відзначити Colormind – сервіс для генерації гармонійних кольорових схем. Використовуючи машинне навчання, Colormind пропонує унікальні кольорові комбінації на основі сучасних дизайнерських трендів або зразків, завантажених користувачем.

Для розробників, які інтегрують UI/UX у процес створення функціональних додатків, існують інструменти на кшталт ChatGPT API та Codex. Ці рішення здатні автоматизувати генерацію HTML-коду, CSS-стилів та навіть JavaScript-функцій для базових елементів інтерфейсу. Використання

таких сервісів дозволяє скоротити час на рутинні завдання, зосереджуючись на складніших аспектах розробки.

Переваги існуючих інструментів

Швидкість розробки. Інструменти на основі ШІ скорочують час на проєктування та тестування інтерфейсів, що особливо важливо для стартапів та невеликих проєктів із обмеженими ресурсами.

Автоматизація рутинних процесів. Замість ручного налаштування, дизайнери можуть довірити створення кольорових схем, шрифтів або навіть структур сторінок інструментам ШІ.

Доступність для новачків. Навіть користувачі без глибоких знань у сфері дизайну можуть створювати привабливі веб-додатки за допомогою таких інструментів, як Wix ADI чи Uizard.

Обмеження та виклики використання

Незважаючи на значні переваги, використання таких інструментів має й певні недоліки. По-перше, більшість рішень орієнтовані на створення стандартних інтерфейсів, що може обмежувати творчий підхід досвідчених дизайнерів. По-друге, автоматизовані рішення не завжди можуть врахувати унікальні вимоги проєкту, що потребує додаткових доопрацювань. Крім того, використання інструментів ШІ може бути викликом у контексті конфіденційності даних користувачів, які аналізуються для створення персоналізованих інтерфейсів.

Існуючі інструменти для автоматизації UI/UX дизайну вже сьогодні дозволяють значно спростити процес розробки веб-додатків. Вони забезпечують ефективність, швидкість і доступність, але потребують уважного підходу до інтеграції, враховуючи обмеження у гнучкості та питання безпеки даних. У подальшому розвиток таких рішень може ще більше змінити підхід до створення інтерфейсів, зосереджуючи увагу дизайнерів на стратегічних та інноваційних завданнях.

1.3 Вибір методів порівняння ефективності AI у створенні інтерфейсів

У дослідженні використання штучного інтелекту (ШІ) для розробки клієнтських інтерфейсів важливим етапом є вибір методів порівняння ефективності різних підходів і технологій. Оскільки існує безліч інструментів, які застосовують ШІ для автоматизації процесів дизайну, від вибору кольорових схем до персоналізації інтерфейсів, важливо правильно оцінити їх ефективність з точки зору кількох критеріїв. Це дозволяє зрозуміти, який інструмент чи метод є найбільш оптимальним для конкретного проєкту.

Основні критерії порівняння ефективності:

Час розробки: Один із основних критеріїв порівняння — це швидкість, з якою інструменти на базі ШІ дозволяють створювати функціональні прототипи або готові інтерфейси. Інструменти, що автоматизують процес проектування, можуть значно скоротити час, необхідний для створення веб-додатка. У цьому випадку порівняння методів буде включати вимірювання часу на різні етапи розробки інтерфейсу.

Якість дизайну: Якість інтерфейсу — ще один важливий критерій. Оцінюється не лише зовнішній вигляд інтерфейсу, а й його зручність та функціональність. Визначення того, як ШІ може покращити або автоматизувати складні дизайнерські рішення, є суттєвим при виборі методу. Для порівняння цього показника можуть використовуватися як суб'єктивні методи (відгуки користувачів, тестування UX), так і об'єктивні методи, що включають аналіз дотримання дизайнерських принципів та стандартів.

Адаптивність: Адаптивність інтерфейсу означає здатність системи змінюватися залежно від пристрою користувача, його переваг або інших контекстуальних факторів. Оцінка ефективності різних методів ШІ у створенні адаптивних інтерфейсів включає порівняння здатності інструментів автоматично підлаштовувати дизайн під різні екрани, роздільну здатність і навіть поведінку користувачів.

Персоналізація: Персоналізація є важливою складовою сучасних інтерфейсів. Методи порівняння можуть включати оцінку того, як інструменти ШІ можуть інтегрувати дані користувача для створення інтерфейсів, що враховують індивідуальні вподобання, історію взаємодії з веб-додатком або навіть емоційний стан.

Ефективність взаємодії користувача з інтерфейсом: Інтерфейси, розроблені за допомогою ШІ, повинні не тільки відповідати вимогам користувачів, але й забезпечувати безперешкодну та інтуїтивно зрозумілу взаємодію. Оцінка ефективності включає в себе кількість кроків, які потрібно виконати для досягнення мети користувача, а також рівень задоволеності користувачів.

Методи порівняння

Експертні оцінки: Один з найбільш поширених методів порівняння ефективності інструментів ШІ — це оцінка результатів за допомогою експертів у галузі дизайну та розробки. Дизайнери та розробники можуть оцінити інтерфейси, створені за допомогою різних ШІ-інструментів, на основі зазначених критеріїв. Цей метод є суб'єктивним, але дає точну картину того, як система працює в реальних умовах.

Кількісне тестування: Кількісні методи включають вимірювання таких показників, як час виконання завдань, кількість помилок, продуктивність інтерфейсу та інші. Цей метод дозволяє отримати об'єктивні дані про ефективність інструментів ШІ.

A/B тестування: Цей метод полягає в порівнянні двох версій інтерфейсу, однією з яких є стандартний дизайн, а іншою — інтерфейс, створений за допомогою ШІ. A/B тестування дозволяє перевірити, яка версія інтерфейсу має кращі показники за різними критеріями, такими як задоволеність користувачів, час на виконання завдань, кількість помилок тощо.

Тестування користувачами: Порівняння методів ШІ може включати також тестування з реальними користувачами, які виконують певні завдання за допомогою інтерфейсів, створених різними методами. Це дозволяє не тільки

оцінити ефективність, але й виявити можливі проблеми в інтерфейсі, які могли бути не враховані під час автоматизованої розробки.

Для ефективного порівняння результатів використання ШІ в створенні інтерфейсів важливо застосовувати комплексний підхід, який враховує як кількісні, так і якісні аспекти. Оцінка часу розробки, якості дизайну, адаптивності, персоналізації та ефективності взаємодії з інтерфейсом є ключовими елементами, що дозволяють визначити переваги і недоліки різних методів. Використання таких методів, як експертні оцінки, кількісне тестування та A/B тестування, дозволяє отримати всебічну картину ефективності ШІ у розробці веб-додатків.

1.4 Висновки аналізу предметної області

Аналіз предметної області допоміг виявити ключові проблеми та перспективі у розробці клієнтських інтерфейсів для веб-додатків, особливо в контексті інтеграції технологій штучного інтелекту.

Традиційні підходи до створення інтерфейсів виявляють свої недоліки у таких аспектах як: відсутність персоналізації, складність навігації та взаємодії, обмежені можливості адаптації. А штучний інтелект у розробці веб-додатків пропонує рішення, які дозволяють усунуть більшість із зазначених проблем завдяки впровадженню алгоритмів штучного інтелекту, обробки природної мови та інших інструментів. З інтеграцію ШІ веб-додатки можуть стати більш адаптивними а саме реагувати на поведінку користувачів і змінюватися відповідно до їхніх проблем, зможуть давати персоналізований контент і функціонал для кожного окремого користувача, і звичайно ефективними завдяки функціоналу ШІ користувач для досягнення результатів буде витрачати меньше часу та зусиль.

Водночас впровадження ШІ вимагає додаткових ресурсів і кваліфікованої роботи розробників. Додаткові витрати часу для впровадження

ШІ можуть бути виправвдані лише у випадках коли очікується суттєвий приріст ефективності та користувацької задоволеності.

Отже аналіз предметної області показав що інтеграція штучного інтелекту в розробку клієнтських інтерфейсів є перспективним, але складним напрямом.

2 РОЗРОБКА МЕТОДОЛОГІЇ ДОСЛІДЖЕННЯ

2.1 Визначення критеріїв оцінки ефективності клієнтських інтерфейсів

Оцінка ефективності клієнтських інтерфейсів є важливим етапом у процесі розробки веб-додатків. Клієнтські інтерфейси визначають, наскільки легко користувач може взаємодіяти з додатком, і яким чином забезпечується задоволення його потреб і вимог. Оскільки інтерфейс є основною точкою взаємодії користувача з веб-додатком, важливо мати чіткі критерії для оцінки його ефективності. Визначення таких критеріїв допомагає як розробникам, так і кінцевим користувачам зрозуміти, наскільки зручний і функціональний створений продукт.

Основні критерії оцінки ефективності клієнтських інтерфейсів

Зручність використання (Usability)

Зручність використання є основним критерієм, що характеризує ефективність інтерфейсу. Це стосується того, наскільки легко користувач може виконати завдання, використовуючи інтерфейс. Зручність використання оцінюється через декілька факторів:

– інтуїтивність — наскільки інтерфейс є зрозумілим і логічним для користувачів;

– час виконання завдання — скільки часу користувач витрачає на виконання певних дій у системі;

– кількість кроків — скільки кроків потрібно для досягнення цілі.

Задоволення користувача (User Satisfaction)

Задоволення користувача визначається через його емоційне ставлення до взаємодії з інтерфейсом. Це важливий аспект, оскільки навіть найефективніший інтерфейс може бути неприязним для користувачів, якщо

він не створює позитивних вражень. Вимірювання цього критерію може включати:

– оцінки користувачів за допомогою опитувань;

– спостереження за поведінкою користувача під час взаємодії з інтерфейсом;

– аналіз відгуків і коментарів, що надаються користувачами;

– ефективність (Efficiency).

Ефективність оцінюється через здатність інтерфейсу допомогти користувачу швидко і без помилок виконувати завдання. Ключовими аспектами є:

– мінімізація помилок користувача: наскільки система допомагає уникнути помилок або виправити їх;

– час виконання завдань: скільки часу необхідно для досягнення цілі;

– кількість необхідних взаємодій: наскільки проста або складна задача для виконання в межах інтерфейсу.

Адаптивність (Adaptability)

Адаптивність — це здатність інтерфейсу адаптуватися до різних пристроїв, роздільної здатності екрану, а також індивідуальних потреб користувачів. Цей критерій важливий у контексті сучасних веб-додатків, які повинні бути доступними на різних платформах (мобільні телефони, планшети, комп'ютери). Оцінка адаптивності включає:

– Чи легко інтерфейс адаптується до різних розмірів екранів?

– Чи є підтримка різних операційних систем і браузерів?

– Чи забезпечується гармонійна адаптація елементів інтерфейсу до змін в поведінці користувачів?

Доступність (Accessibility)

Доступність визначає, наскільки інтерфейс доступний для людей з обмеженими можливостями. Це включає врахування потреб людей з порушеннями зору, слуху, руховими обмеженнями або когнітивними порушеннями. Для оцінки цього критерію застосовуються такі підходи:

– використання екранних рідерів і голосових помічників;

– оцінка можливості навігації за допомогою клавіатури або жестів;

– забезпечення доступності контенту через аудіо- і текстові описи.

Продуктивність (Performance)

Продуктивність включає в себе здатність інтерфейсу працювати швидко, без затримок і збоїв. Оцінка продуктивності інтерфейсу передбачає вимірювання:

– часу завантаження сторінок і швидкості відгуку системи;

– стабільності роботи, наприклад, відсутності збоїв або падінь продуктивності під час високих навантажень;

– оптимізацію використання ресурсів пристроїв.

Безпека (Security)

Безпека є критичним фактором, оскільки клієнтські інтерфейси часто взаємодіють з конфіденційною інформацією користувачів. Оцінка безпеки включає:

– захист від несанкціонованого доступу до персональних даних;

– захист від атак, таких як SQL-ін'єкції, XSS (кросс-сайт скриптинг) та інші уразливості;

– надійність аутентифікації та авторизації користувачів.

Вибір методів оцінки критеріїв

Для оцінки вищезгаданих критеріїв використовуються різні методи, включаючи:

– тестування з користувачами (User Testing): оцінка за допомогою спостережень і фідбеку реальних користувачів;

– A/B тестування: порівняння різних варіантів інтерфейсу, щоб визначити, який з них ефективніший;

– автоматизовані тести: використання програмного забезпечення для тестування таких аспектів, як продуктивність, доступність і безпека;

– методи UX-аналізу: вивчення поведінки користувачів за допомогою спеціальних інструментів, що аналізують кліки, час на виконання завдань та інші параметри.

Чітке визначення критеріїв оцінки ефективності клієнтських інтерфейсів є необхідним для розробки веб-додатків, які будуть не тільки функціональними, але й зручними для користувачів. Вибір критеріїв, таких як зручність використання, задоволення користувача, адаптивність та доступність, дозволяє комплексно оцінити ефективність інтерфейсу і забезпечити високий рівень взаємодії з користувачем. Ретельне застосування цих критеріїв на різних етапах розробки допомагає створювати інтерфейси, які відповідають сучасним вимогам і потребам користувачів.

2.2 Побудова архітектури веб-додатків для тестування AI-підходів

У розробці веб-додатків, що використовують штучний інтелект (ШІ) для покращення клієнтських інтерфейсів, важливим етапом є побудова відповідної архітектури. Архітектура веб-додатку має забезпечити ефективне впровадження й тестування різних AI-підходів, таких як машинне навчання, обробка природної мови, комп'ютерний зір та інші технології ШІ. Оскільки результати тестування безпосередньо залежать від правильності побудови архітектури, важливо врахувати вимоги до масштабованості, швидкодії, безпеки та інтеграції з іншими системами.

Основні аспекти побудови архітектури:

Модульність і масштабованість.

Веб-додатки, що використовують AI-підходи, повинні бути модульними і здатними масштабуватися в разі необхідності. Це важливо для підтримки численних моделей штучного інтелекту, які можуть виконувати різні задачі (наприклад, класифікація, прогнозування, персоналізація контенту). Модульність дозволяє окремо тестувати кожен з компонентів додатку, таких

як система аутентифікації, модуль ШІ для рекомендацій, аналітика даних, а також інтерфейс користувача. Масштабованість важлива для забезпечення високої продуктивності в умовах зростання кількості користувачів або навантаження на систему.

Інтеграція ШІ-моделей.

Веб-додаток повинен забезпечити легку інтеграцію з різними типами моделей ШІ. Це може включати підключення до існуючих фреймворків для машинного навчання, таких як TensorFlow, PyTorch, або scikit-learn. Ідеальною архітектурою буде та, що дозволяє безперешкодно оновлювати та впроваджувати нові моделі без значних змін у загальній структурі додатку. Для цього можуть бути використані контейнери (Docker) і оркестрація (Kubernetes), що дозволяють ізолювати кожну модель і швидко масштабувати їх для обробки великої кількості запитів.

Розподілені системи та мікросервіси.

Використання архітектури мікросервісів є ефективним підходом для тестування різних AI-підходів. Кожен мікросервіс може відповідати за виконання певної функції, наприклад, обробку даних, навчання моделі або здійснення прогнозів. Мікросервіси дозволяють легко оновлювати і тестувати окремі компоненти без впливу на інші частини системи. Водночас, для забезпечення ефективної комунікації між мікросервісами використовуються API та обміну повідомленнями через брокери (наприклад, Kafka або RabbitMQ).

Обробка і зберігання даних.

Для тестування AI-підходів необхідно мати ефективну систему обробки та зберігання даних. Архітектура повинна включати компоненти для збору, зберігання та обробки великих обсягів даних, необхідних для тренування моделей штучного інтелекту. Це можуть бути бази даних NoSQL (наприклад, MongoDB), що підходять для зберігання неструктурованих даних, або реляційні бази даних для зберігання структурованої інформації. Для аналізу

даних можуть бути застосовані інструменти великих даних (Big Data), як Apache Hadoop або Apache Spark.

Тестування та моніторинг.

Веб-додаток для тестування AI-підходів повинен включати в себе систему моніторингу й тестування. Це дозволяє відстежувати продуктивність і точність моделей ШІ в реальному часі, а також виявляти потенційні проблеми з адаптацією моделей до нових умов або даних. Тестування ШІ включає перевірку якості моделі, її здатність до адаптації, швидкість обробки запитів, а також відповідність результатів прогнозування реальним даним. Важливо мати інструменти для автоматизованого тестування моделей, що дозволяють швидко оцінювати їх точність і продуктивність у різних сценаріях.

Безпека і конфіденційність.

Веб-додатки, що включають AI-підходи, повинні забезпечувати високий рівень безпеки і конфіденційності даних. Це передбачає використання шифрування для захисту даних, а також застосування механізмів аутентифікації та авторизації для забезпечення безпеки доступу до чутливих даних. Зважаючи на використання великих обсягів персональних даних для тренування моделей ШІ, важливо дотримуватись стандартів захисту персональної інформації, таких як GDPR або інші нормативи захисту даних.

Архітектурні рішення

Побудова архітектури веб-додатку для тестування AI-підходів може включати використання таких архітектурних рішень:

– API-first підхід: Веб-додаток може бути побудований за допомогою API-first архітектури, де всі сервіси мають чітко визначені API для взаємодії між різними компонентами системи. Це дозволяє легко інтегрувати моделі ШІ та інші сервіси;

– контейнери та мікросервіси: Використання Docker для контейнеризації моделей ШІ та Kubernetes для їх оркестрації дозволяє досягти високої гнучкості і масштабованості в розробці та тестуванні AI-підходів;

– платформи для розробки і тестування: Використання платформ для машинного навчання (наприклад, Google Cloud AI, Amazon AWS Machine Learning або Microsoft Azure ML) дозволяє значно полегшити розробку і тестування моделей ШІ.

Правильно побудована архітектура веб-додатків є основою для ефективного тестування та впровадження AI-підходів. Модульність, масштабованість, інтеграція з різними типами моделей, а також безпека і ефективне зберігання даних — всі ці аспекти забезпечують можливість успішного тестування, аналізу та вдосконалення ШІ-рішень для розробки інтерфейсів веб-додатків. Врахування цих факторів дозволяє створювати ефективні, адаптивні та безпечні веб-додатки, здатні покращити взаємодію з кінцевими користувачами.

2.3 Обґрунтування використання методів порівняння

Методи порівняння є ключовими інструментами у дослідженні ефективності штучного інтелекту (ШІ) для створення клієнтських інтерфейсів веб-додатків. Застосування порівняльного аналізу дозволяє оцінити результати роботи різних AI-підходів, виявити їх переваги та недоліки, а також визначити, який із методів є найбільш оптимальним для конкретного випадку.

Обґрунтування вибору методів порівняння базується на таких аспектах:

Актуальність порівняльного аналізу

У розробці клієнтських інтерфейсів веб-додатків основною метою є створення інтерфейсу, який буде одночасно зручним, інтуїтивно зрозумілим і естетично привабливим. ШІ може впливати на такі аспекти, як:

– персоналізація інтерфейсу;

– автоматизація процесів (наприклад, динамічне оновлення контенту);

– прогнозування поведінки користувача.

Порівняння ефективності різних AI-алгоритмів у цих аспектах є важливим для визначення найбільш перспективних рішень.

Основні критерії порівняння

Для обґрунтованого вибору оптимального підходу необхідно порівняти різні AI-методи за певними критеріями. Ці критерії були визначені у пункті 2.1, серед них:

– зручність користування (usability): оцінка впливу AI на швидкість і легкість виконання завдань користувачем;

– ефективність: порівняння швидкості та точності обробки даних;

– персоналізація: здатність системи адаптуватися до потреб конкретного користувача;

– відгук користувачів: оцінка сприйняття інтерфейсу через анкетування або зворотний зв'язок.

Методи порівняння.

Для оцінки ефективності AI-підходів у розробці інтерфейсів можуть бути використані такі методи:

Емпіричне порівняння

Це найпоширеніший підхід, що включає тестування декількох інтерфейсів, розроблених із використанням різних AI-методів. Учасники тестування виконують однакові завдання, після чого аналізуються їхні результати (наприклад, час виконання завдань, кількість помилок).

– перевага: дає прямі результати роботи з реальними користувачами;

– недолік: вимагає значних ресурсів для організації тестувань;

– експертна оцінка

Включає аналіз створених інтерфейсів групою фахівців у галузі UI/UX-дизайну та ШІ. Експерти оцінюють системи за визначеними критеріями, враховуючи можливості адаптації та зручність використання.

Перевага: швидкість проведення.

Недолік: суб'єктивність оцінки.

Аналіз продуктивності моделей

Порівняння базується на метриках роботи AI-моделей, таких як точність прогнозів, швидкість обробки даних, обсяг використаних ресурсів. Наприклад, моделі можуть бути протестовані на реальних даних, після чого їх продуктивність порівнюється за стандартизованими показниками.

– перевага: об'єктивність даних;

– недолік: не враховує враження користувачів.

Метод A/B-тестування

Цей підхід передбачає паралельне тестування двох варіантів інтерфейсів, створених із використанням різних AI-підходів, на реальних користувачах. Отримані дані використовуються для визначення переваг кожного варіанту.

– перевага: ефективність в реальних умовах використання;

– недолік: обмеження масштабів тестування.

Аргументи на користь порівняння:

Універсальність

Метод порівняння дозволяє оцінювати ефективність рішень у різних аспектах: як технічних (швидкодія та точність), так і психологічних (зручність і задоволеність користувачів).

Гнучкість

Різні методи порівняння можуть бути адаптовані до конкретних задач. Наприклад, для оцінки продуктивності AI-моделей підходять метрики точності та швидкості, а для визначення задоволеності користувачів – соціологічні опитування.

Практичне застосування

Отримані результати можуть бути безпосередньо використані для вдосконалення інтерфейсів, оптимізації моделей ШІ або покращення загального досвіду користувачів.

Використання методів порівняння є обґрунтованим підходом для оцінки ефективності ШІ в розробці клієнтських інтерфейсів веб-додатків. Завдяки різноманіттю методів, що враховують як технічні показники, так і досвід

користувачів, можна зробити висновки про доцільність застосування того чи іншого AI-підходу в конкретних умовах.

2.4 Висновки з 2 розділу

У другому розділі було проведено детальний аналіз ключових аспектів, пов’язаних із застосуванням штучного інтелекту (ШІ) у розробці клієнтських інтерфейсів веб-додатків. Розглянуто критерії оцінки ефективності таких інтерфейсів, побудовано архітектуру для тестування AI-підходів та обґрунтовано використання методів порівняння.

Основні висновки розділу:

Критерії оцінки ефективності клієнтських інтерфейсів

Визначено ключові параметри, що дозволяють об’єктивно оцінити результативність інтеграції ШІ у клієнтські інтерфейси, такі як зручність, ефективність, персоналізація та задоволеність користувачів. Ці критерії є основою для подальших тестувань та порівнянь.

Побудова архітектури веб-додатків

Розроблена архітектура включає модульну структуру, підтримку інтеграції з ШІ-моделями, мікросервісну організацію, а також системи для збору й аналізу даних. Вона забезпечує масштабованість, гнучкість та можливість тестування різних AI-підходів у реальних умовах використання.

Обґрунтування методів порівняння

Обґрунтовано доцільність застосування порівняльного аналізу для оцінки ефективності ШІ у розробці клієнтських інтерфейсів. Розглянуто методи емпіричного порівняння, експертної оцінки, аналізу продуктивності моделей та A/B-тестування, що дозволяють оцінити як технічні, так і психологічні аспекти.

Загальні рекомендації

Виявлено, що комплексний підхід до оцінки ефективності ШІ у створенні інтерфейсів дозволяє врахувати різноманітні чинники, включаючи досвід кінцевого користувача та технічну продуктивність моделей. Це створює підґрунтя для вибору найбільш доцільного AI-підходу залежно від потреб проєкту.

Таким чином, другий розділ закладає основу для практичної реалізації та тестування веб-додатків з інтегрованими AI-підходами, що стане предметом розгляду в наступному розділі. Отримані висновки ідентифікують необхідні кроки для подальшого вдосконалення клієнтських інтерфейсів за допомогою ШІ.

3 РЕАЛІЗАЦІЯ ВЕБ-ДОДАТКІВ ДЛЯ ТЕСТУВАННЯ

3.1 Розробка традиційного клієнтського інтерфейсу

3.1.1 Project Book

Для розробки клієнтського веб-додатку для проекту Project Book були застосовані традиційні підходи. Єдиною основою для побудови інтерфейсу є використання сучасних веб-технологій: HTML, CSS та Javascript, які є легко реалізованими та дуже гнучкими в дизайні.

HTML (HyperText Markup Language) – мова розмітки для створення структури веб-сторінки [3]. У нашому випадку використання HTML необхідне для відображення основних елементів сторінки, таких як заголовки, форми, кнопки, списки, поля введення тощо. Структурування коду було зроблено таким чином, щоб його було легко зрозуміти з точки зору користувацького інтерфейсу.

CSS: Каскадні таблиці стилів – це мова стилів, яка дозволяє проектувати вигляд конкретної веб-сторінки [4]. CSS уможливила використання градієнтів на фоні, декоративних елементів та анімації на кнопках, що зробило інтерфейс привабливим. Усі стилі були згруповані у зовнішньому файлі styles.css, де була збережена чистота щодо основного HTML-коду.

Для налаштування динамічної взаємодії між користувачем та інтерфейсом ми використали JavaScript. Ця мова програмування дозволила впровадити інтерактивні функції, такі як відображення рекомендацій щодо книг, коли користувач обирає жанр або натискає кнопку. Усі дані про книги зберігалися у форматі JSON (JavaScript Object Notation). JSON – це текстовий формат для зберігання структурованих даних, який легко розміщується в JavaScript [5] У нашому випадку це допомогло, оскільки зробило доступ до інформації про книги досить простим.

Під час розробки клієнтського інтерфейсу були виконані наступні кроки:

– створення макету сторінки за допомогою HTML;

– додавання стилів за допомогою CSS, включаючи адаптивність сторінки для мобільних пристроїв;

– написання JavaScript для обробки всієї взаємодії з користувачем;

– тестування сторінки на різних пристроях і браузерах.

Результатом цього етапу є функціональний веб-додаток, користувачі обирають жанр і отримують рекомендації щодо книг. Відсутня серверна частина та база даних спростили розгортання.

На цьому проєкті використали даний шрифт "Nunito", що підключається через бібліотеку Google Fonts, адже його сучасний, дружній і легкий для читання зовнішній вигляд робить шрифт ідеальним, щоб вписатися у веб-додатки для інтерфейсу.

Код для підключення цього шрифту:

@import
url('https://fonts.googleapis.com/css2?family=Nunito:wght@400;600;700&display=swap');

Ми мали різні варіації товщини шрифту (400, 600, 700). Зокрема: 400 для основного тексту, – наприклад, описів книг, для його комфортного читання, 600 для посилань у навігаційному меню й кнопок жанрів, підкреслюючи їх функціональність, 700 – для заголовків, щоб надати візуальної ваги й зрозуміти їх помітність.

Фонові стилі проектування

Насправді, ми використовували такі фони для нашого проекту, щоб задати оточуючий естетичний настрій нашому сайтові:

Легка основа, в якій століттями проглядали книгу (photo1. jpg)-додати атмосферності, специфічно підказуючи тему нашого сайту.

Кожній секції їй додано напівпрозорий фон з кольорами, які пов'язані з комфортом та читанням, такими, як світло-бурштинові, кремові, пастельний рожевий і блакитний, з метою поліпшення візуальної роздільності розділів,

зберігаючи національний стиль. Напівпрозорий фон дає відчуття легкості, а також забезпечує видимість фонового ілюстративного зображення.

Кнопки вибору жанрів

Кожен жанр має свій колір, який підкреслить його індивідуальність і спростить користувачу асоціацію з улюбленим жанром:

– фантастика: світлий жовтий колір, символізуючий фантазії, мрії;

– класика: світлуг зелений, символ традиційності та досконалості;

– детектив: світло-сірий, що тягне нейтральність і таємничість;

– романтичний: світло-рожевий, що нагадує про ніжність. Наукова література: світло-блакитний, свідчить про зв'язок із технологіями та знаннями;

– пригоди: світло-помаранчевий символізує енергію та динаміку.

Навігаційне меню.

Навігаційне меню зроблено з акцентом на простоту та інтуїтивність:

Світло-коричневий фон з напівпрозорістю, що робить його візуально легше.

Елементи меню також з плавним ефектом наведення (зміни кольору на темніший колір), які покращують якомога більше реакцію користувача на використання інтерфейсу.

Рекомендації

Зображення обкладинок, заголовки, короткі описи і кнопки "Підібрати" – все це потрапило до секції рекомендації книг. Такій формі кожного елемента застосоване компактне й лаконічне оформлення, щоб не перевантажити користувача зайвою інформацією.

Підвал (footer).

У єдиному стилі з підпірки оформлений у напівпрозорому фоні з соціальними посиланнями. Він надає повноту дизайну та доступ до інших ресурсів.

Загалом стиль виведено у всіх:

Плавні переходи для підведених елементів (кнопків, посилань), щоб забезпечити комфортний досвід використання.

Гнучкі макети з використанням Flexbox для того, щоб елементи адаптувалися до різних розмірів екрана.

Гранична ширина контейнерів для центрального розташування вмісту.

Таким чином, легко, естетично і адаптивно можна проектувати солідні дизайни, які відповідають вимогам дизайну.

Нижче наведен результат верстки проекту Project Book(index.html, styles.css), вихідний код наведено у Додатку А, рис. 3.1.

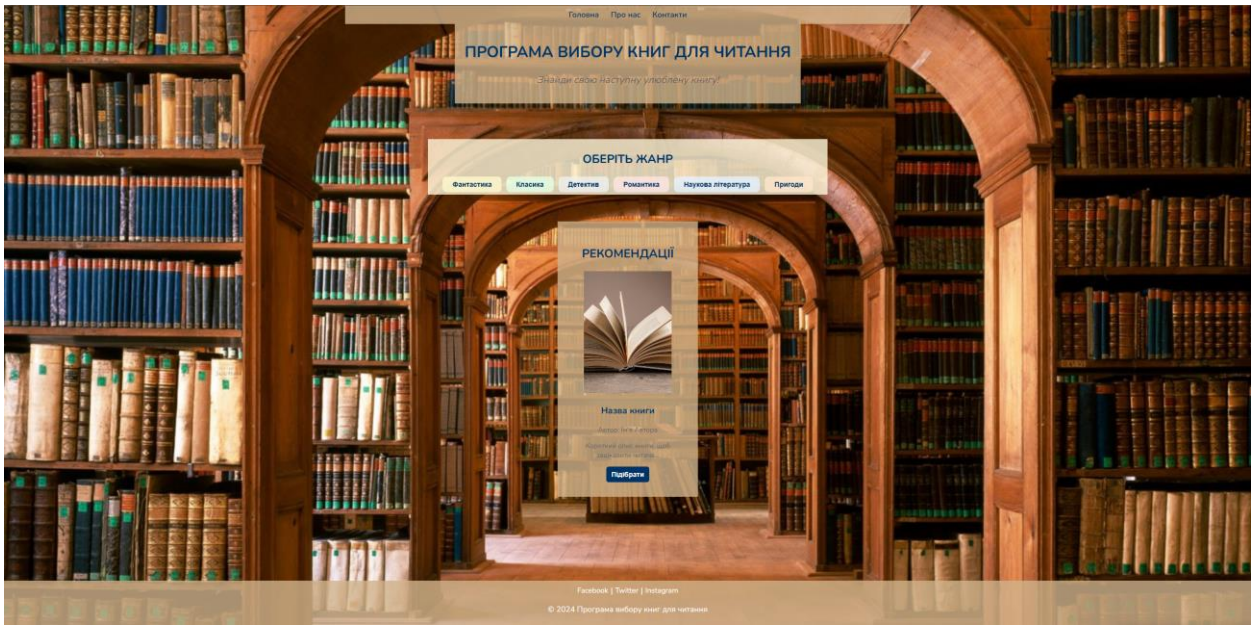


Рисунок 3.1– Project Book

Далі я хотів би продемонструвати практичну частину цього веб-додатку, а саме рекоменадації щодо вибору книги по жанру, код наведено у Додатку А, рис. 3.2.

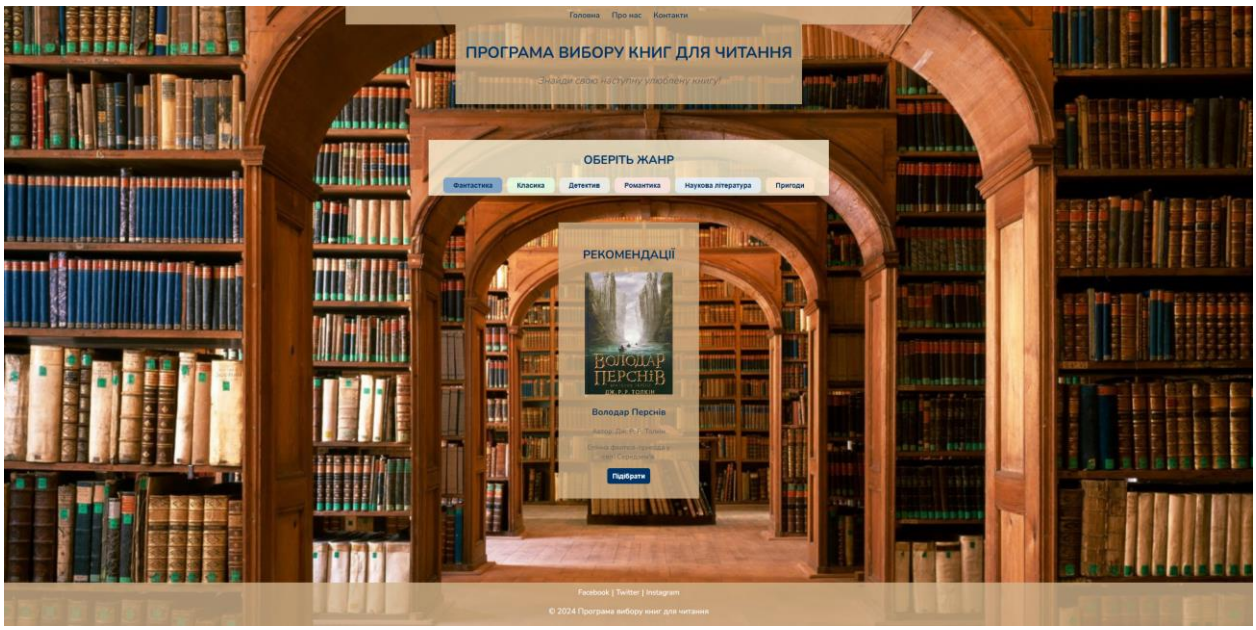


Рисунок 3.2 – Практична частина Project Book

Практична частина цього сайту працює завдяки файлу script.js, саме він забеспечує інтерактивність програми, взаємодію з кнопками вибору жанру та відображення інформації про книги. Код написанний на JavaScript і складається з таких функціональних частин:

а) писки книг за жанрами

На початку файлу визначений об'єкт «booksByGenre», який містить дані про книги, поділені за жанрами.

Кожен жанр має такі атрибути:

– title — назва книги.

– author — автор книги.

– description — короткий опис.

– image — шлях до обкладинки книги.

Це дозволяє зберігати всі необхідні дані про книги в структурованому форматі для динамічного відображення.

б) обробка кнопок по вибору жанру

Код:

```
document.querySelectorAll('.genre-buttons button').forEach(button => {
  button.addEventListener('click', () => {
```

```
            selectedGenre = button.dataset.genre;

            document.querySelectorAll('.genre-buttons button').forEach(btn => {
                btn.style.backgroundColor = ''; // Скидання кольору
            });

            button.style.backgroundColor = '#82a3c3'; // Виділення вибраного жанру
        });
    });
```

Функціонал:

Всі кнопки вибору жанрів прослуховують подію click.

При натисканні зчитується жанр із атрібута «data-genre» настиснутої кнопки, та виділення інших кнопок скидається, а вибрана кнопка змінює свій фон. Цей механізм дозволяє користувачу вибрати жанр книги в зручній та наочній формі.

в) Обробка кнопки "Підібрати"

Код:

```
document.querySelector('.details').addEventListener('click', () => {
    if (!selectedGenre) {
        alert("Будь ласка, оберіть жанр!");
        return;
    }

    const book = booksByGenre[selectedGenre];
    const bookItem = document.querySelector('.book-item');

    bookItem.querySelector('h3').textContent = book.title;
    bookItem.querySelector('p:nth-of-type(1)').textContent = `Автор: ${book.author}`;
    bookItem.querySelector('p:nth-of-type(2)').textContent = book.description;
```

```
        bookItem.querySelector('img').src = book.image;
        bookItem.querySelector('img').alt = `Обкладинка книги: ${book.title}`;
    });
```

Функціонал:

– кнопка "Підібрати" викликає обробник подій click;

– якщо користувач не вибрав жанр (selectedGenre не визначений), з’являється попередження за допомогою alert;

Якщо жанр вибрано, то зчитуються дані про книгу для цього жанру з об'єкта booksByGenre, в елемент, який відповідає за відображення вибраної книги (.book-item), оновлюється:

– назва книги (h3);

– автор (p:nth-of-type(1));

– опис (p:nth-of-type(2)).

Зображення обкладинки (img), включно з атрибутом alt для покращення доступності. Ця функція дозволяє динамічно змінювати вміст сторінки залежно від вибору користувача.

Весь код script.js написан у Додатку А.

3.1.2 Tipsday

Tipsday – це проект, розроблений як інтерактивний веб-додаток, що надає щоденні особисті мотивуючі поради. Ідея проекту полягає в тому, щоб допомогти людям надихатися, залишатися продуктивними та підвищувати якість життя за допомогою простих у використанні та корисних рекомендацій у різних категоріях.

Розробка проекту почалася зі створення основного макету сторінки, а також визначення ключових функцій, які мали бути реалізовані. Дизайн сайту мінімалістичний та естетичний. Простота, легкість взаємодії та комфортність – ось основні принципи, на яких базувався дизайн.

На головній сторінці додатку можна знайти наступні елементи:

– Заголовок та короткий опис. Вони розповідають користувачеві про призначення додатку та дарують позитивний настрій.

– форма для вибору категорії. Виконується у вигляді випадаючого списку, за допомогою якого користувач обирає одну з існуючих категорій: «Здоров'я», “Освіта”, “Відпочинок” або “Все”.

– блок, де будуть відображатися підказки. Містить простір, в якому користувач бачить заохочувальну підказку після натискання кнопки.

– отримати підказку. Цей елемент є тією кнопкою, яку користувач використовує для переходу в додаток.

– контейнер для результату. Тут відображається результат – обрана порада, що відповідає даній категорії.

Код макету сторінки index.html буде наведено в додатку А.

Дизайн сторінки створено в м'яких пастельних тонах, щоб створити затишну атмосферу. На задньому плані розмитий градієнт між яскравими відтінками полегшує сприйняття. Крім того, деякі елементи, такі як кнопки або блоки результатів, пропонують легкі ефекти взаємодії, такі як зміна кольору або тіні при наведенні, надаючи інтерфейсу динамічності та сучасності.

Код створення дизайна сайту style.css буде наведено в Додатку А.

Функціональність програми забезпечується скриптом на JavaScript. У цьому проекті масив tips є сховищем усіх підказок. Кожна підказка має дві характеристики: текст підказки (text) та категорію, до якої вона належить (category).

Коли користувач натискає кнопку «Отримати пораду», програма виконує наступні дії:

– зчитує обрану категорію з випадаючого списку;

– список порад фільтрується відповідно до категорії. Для пункту «Всі» фільтрація не застосовується.

Довільно виберіть одну з доступних відфільтрованих порад.

Текст цієї поради буде відображено в спеціальному блоці для результатів.

Цікавою є реалізація випадкового вибору порад, для чого був використаний генератор випадкових чисел, який гарантує, що з кожним натисканням кнопки користувач отримає нову пораду, а не повторення попередньої.

Код функціональної частини сайту script.js буде наведено в Додатку А.

Процес тестування проекту включав тестування на різних пристроях, таких як мобільні телефони, планшети та настільні комп'ютери. Завдяки адаптивному дизайну інтерфейс виглядає однаково зручно на всіх екранах.

Однією з важливих складових є інтерактивність додатку, що повністю задовольняє життя завдяки гармонійному поєднанню HTML, CSS та JavaScript. Стилі кнопок, полів вибору та їх результатів адаптовані так, щоб користуватися додатком було не тільки практично, але й приємно.

Проект Tipsday, невеликий, але повністю функціональний, ілюструє, як можна створити веб-додаток, що відповідає вимогам сучасного користувацького досвіду. Його простота використання та інтуїтивно зрозумілий інтерфейс будуть дуже корисними для щоденної мотивації значної частини користувачів.

У майбутньому планується додати функцію збереження улюблених порад або надсилання сповіщень про нові доступні поради.

Нижче буде продемонстрована головна сторінка цього проекту на якої показано як виглядає дизайн та макет сторінки, рис. 3.3.



Рисунок 3.3 – Головна сторінка

Також нижче наведена функціональна демонстрація проекту, в якому обрана категорія підсказки та виведен результат, рис. 3.4.

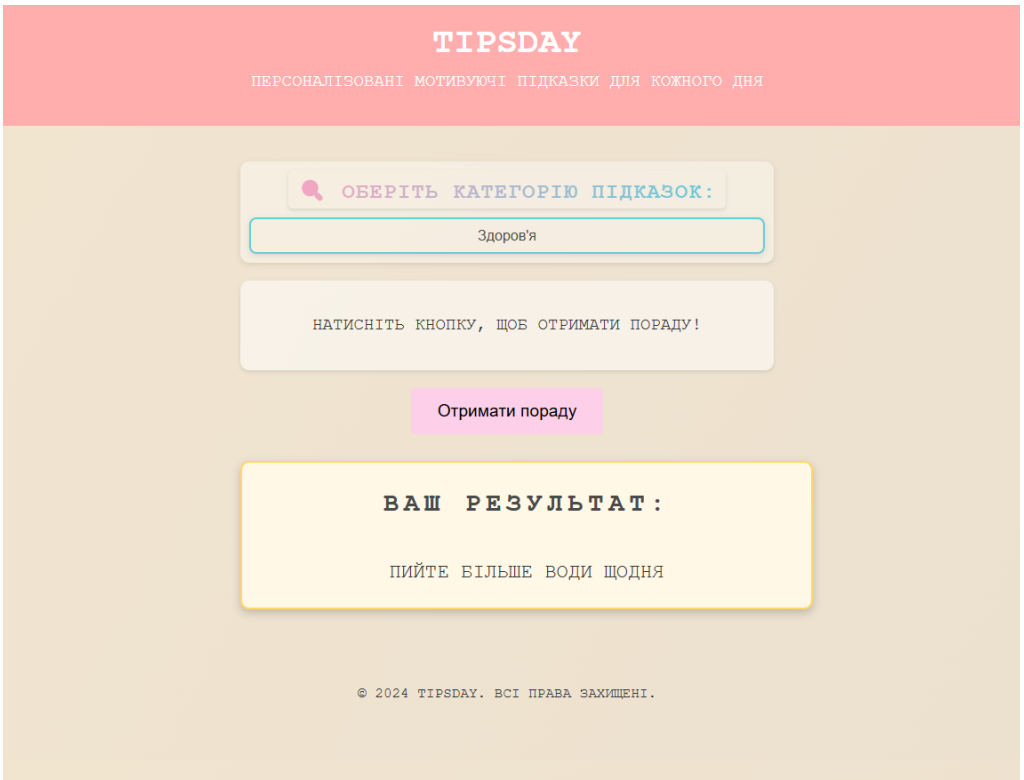


Рисунок 3.4 – Функціональна частина

3.2 Інтеграція штучного інтелекту в процес розробки веб-додатку

3.2.1 Project_AI

Розробка клієнтського інтерфейсу за допомогою штучного інтелекту: проект з рекомендацій книг

Сьогодні, в епоху цифрових технологій і легкого доступу до інформації, ми стикаємося з проблемою вибору відповідного контенту з тисяч варіантів, який найкраще відповідає смаку конкретного користувача. Серед проблем вибору в багатьох сферах, однією з них є проблема книг: з тисячами виданих книг, залишається складно знайти вибір, який може сподобатися читачеві. Метою проекту була розробка системи, яка використовує штучний інтелект для створення персоналізованих книжкових рекомендацій на основі вподобань користувача.

Технічна реалізація проекту.

Ядром проекту буде веб-додаток, побудований за допомогою Flask, який є мікрофреймворком для Python. Flask – це популярна мікрофреймворк для створення веб-додатків на мові програмування Python[6].Його основні характеристики включають: мінімалістичність, гнучкість, розширюваність, та легкість у використанні. Інтерфейс користувача додатку буде використовувати HTML, CSS та JavaScript, забезпечуючи сучасний та привабливий вигляд.

Коли користувач відкриває додаток, його зустрічає дуже пристойний інтерфейс, доповнений декоративними елементами, такими як анімовані кольорові кола, що оживляють дизайн. Основна частина сторінки містить заголовок, опис того, про що йдеться в додатку, і форму для введення текстових налаштувань щодо книг. Наприклад, користувач може вказати улюблений жанр, тему чи навіть конкретного автора. Після натискання кнопки «Отримати рекомендацію» додаток надсилає запит на сервер для обробки.

Зовнішній вигляд додатку визначається за допомогою файлу styles.css. Тут використовується шрифт Nunito, світлі кольорові акценти, напівпрозорі елементи для створення відчуття легкості та простоти. Крім того, він має адаптивний дизайн, завдяки якому інтерфейс добре відображається на великих екранах і на портативних пристроях.

Серверна частина та інтеграція зі штучним інтелектом.

Основна логіка обробки даних закладена у файлі “app.py”. Він використовує клієнт g4f, що надає доступ до моделі GPT-4o-mini. Це дозволяє надсилати користувацькі запити до нейромережі та отримувати змістовні відповіді у вигляді рекомендацій. G4F – це python пакет, що надає єдиний інтерфейс для доступу до різноманітних мовних моделей та AI-сервісів від різних провайдерів[7].

Сценарій роботи сервера включає наступне:

Дані, що вводяться користувачем, передаються в текстовому форматі через HTML-форму.

Сервер приймає дані за допомогою POST і звертається до клієнта для генерації відповіді на основі ШІ.

GPT-4o-mini аналізує запит, визначаючи його контекст, і надає рекомендацію, яка найкраще відповідає критеріям.

Відповідь відображається користувачеві.

Також реалізовано обробку помилок: якщо AI зіпсується або виникне інша помилка, то користувач отримає відповідне повідомлення на сторінці, що дозволить уникнути непорозумінь.

Додаткові функції та інтерактивність Цей скрипт призначений для інтерактивності: він обробляє введення тексту користувачем ще до надсилання запиту, якщо поле введення тексту порожнє. Пізніше буде розширено можливість додавання анімованих елементів у відповідь від сервера. Усі коди до цього проекту будуть у додатку Б.

Нижче наведена головна сторінка проєкту Project_AI, на ній показується макет сайту зі стилями та декоративними анімаційними елементами, рис. 3.5.

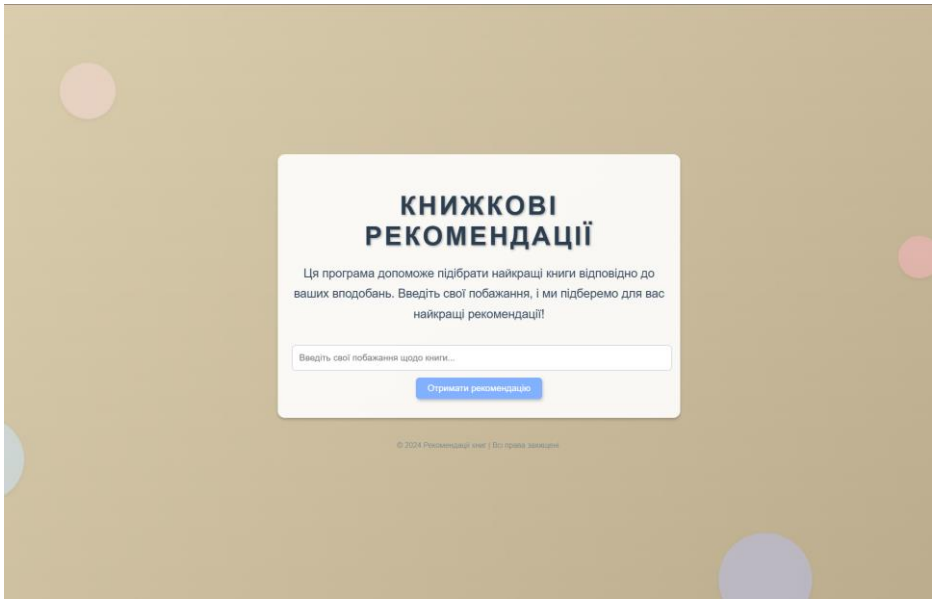


Рисунок 3.5 – Головна сторінка Project_AI

Щодо функціональнох частини веб-додатку ось як виглядає результат роботи серверної частини проєкту, рис. 3.6.

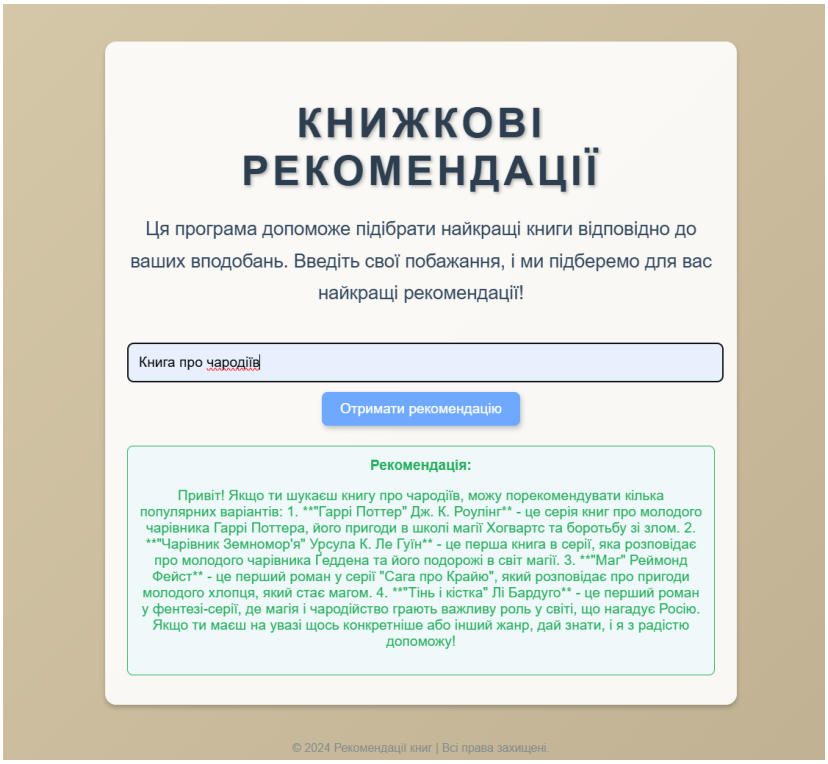


Рисунок 3.6 – Функціональна частина додатку

3.2.2 TipsAI

Розробка веб-додатку для персоналізованих порад: Проєкт TipsAI

В умовах сучасного життя кожен прагне до підвищення продуктивності, саморозвитку та ефективного управління своїм часом. Проєкт TipsAI був створений для вирішення цієї задачі: надання індивідуалізованих щоденних порад, що адаптуються до інтересів користувача за допомогою штучного інтелекту.

Мета та концепція проєкту.

TipsAI — це веб-додаток, який дозволяє користувачам отримувати персоналізовані поради відповідно до їхніх уподобань. Сервіс побудований на базі Flask і інтегрований із потужною мовною моделлю GPT-4o-mini, що забезпечує високу якість і актуальність рекомендацій.

Інтерфейс та дизайн.

Головна сторінка додатку вирізняється сучасним і стильним дизайном, який реалізований за допомогою HTML та CSS. У центрі уваги — форма для введення уподобань користувача, що відображається у вигляді картки з м’якими кольорами та анімованими ефектами.

Стиль оформлення задається у файлі styles.css, де використовується шрифт Nunito, легкі зелено-червоні акценти та адаптивний фон із зображенням. Ключові елементи:

Заголовок: Чіткий і контрастний текст, що створює перше враження.

Картки: Анімовані блоки, які змінюють вигляд при наведенні, додаючи динаміки.

Фон: Ніжні градієнти, що створюють комфортну візуальну атмосферу.

Крім основного функціоналу, сторінка пропонує користувачам цікаві факти, які відображаються у спеціально відведеній секції.

Клієнтська логіка.

Файл script.js відповідає за взаємодію користувача із системою. Код цього файлу буде у Додатку Б. Алгоритм обробки виглядає наступним чином:

Користувач вводить тему чи сферу інтересів (наприклад, "продуктивність").

Кнопка "Отримати пораду" ініціює запит на сервер через метод fetch.

У разі помилки користувач отримує відповідне повідомлення, що робить додаток зручним і зрозумілим у використанні.

Серверна частина та робота з ШІ.

Основна логіка додатку реалізована у файлі server.py. Сервер, побудований на Flask, отримує запити від клієнта, передає їх у мовну модель через API клієнта g4f і повертає результат у вигляді персоналізованої поради.

Алгоритм роботи серверної частини:

– отримання текстового запиту користувача;

– виклик моделі GPT-4o-mini для генерації відповіді;

– обробка відповіді й повернення її у вигляді JSON.

Особливу увагу приділено обробці помилок: якщо модель недоступна або виникають інші проблеми, сервер повертає повідомлення, що дозволяє уникнути негативного досвіду.

Результати та перспективи розвитку.

Проєкт TipsAI демонструє потенціал використання штучного інтелекту для персоналізації користувацького досвіду. Вже зараз додаток може:

– пропонувати щоденні поради, що адаптовані до уподобань;

– створювати позитивний настрій через інтерактивний і візуально приємний інтерфейс.

У майбутньому планується розширення функціоналу, зокрема додавання мультимовної підтримки, розширення бази тем та інтеграція з іншими сервісами, такими як календарі чи трекери задач.

Проєкт TipsAI є прикладом успішної реалізації сучасного веб-додатку, що поєднує в собі простоту, функціональність і використання інноваційних технологій.

Ось як виглядає головна сторінка з використанням усіх стилей, та дизайнерськими рішеннями, рис. 3.7.

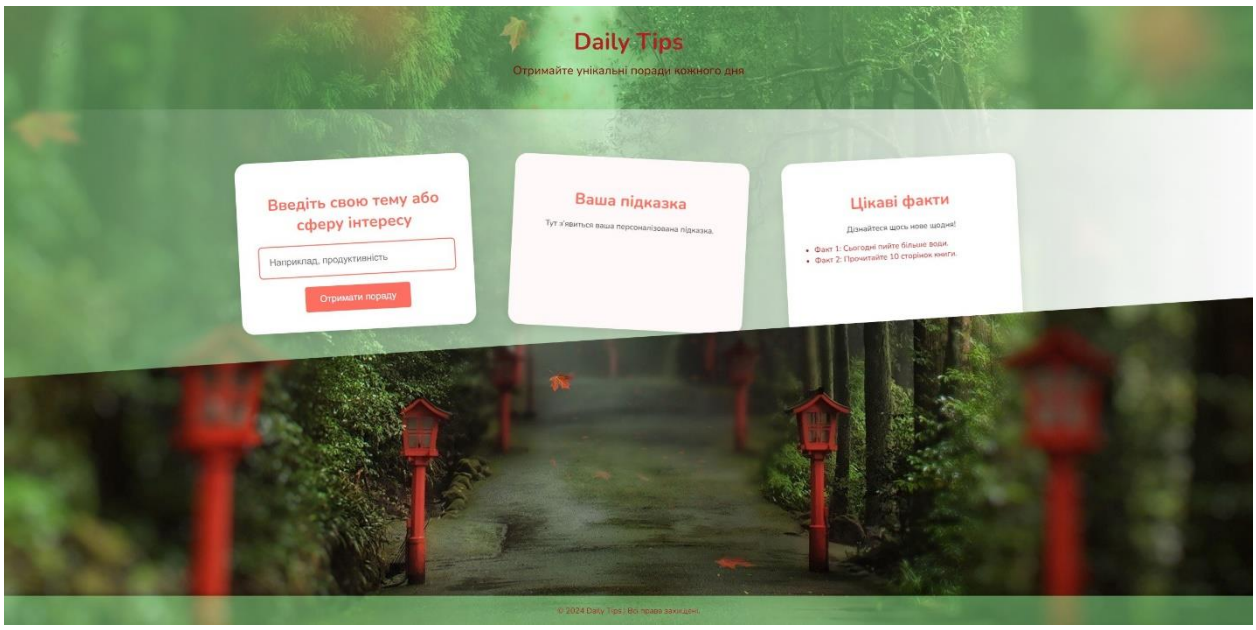


Рисунок 3.7 – Головна сторінка TipsAI

Далі буде показана демонстрація функціональної частини цього проекту, рис. 3.8.

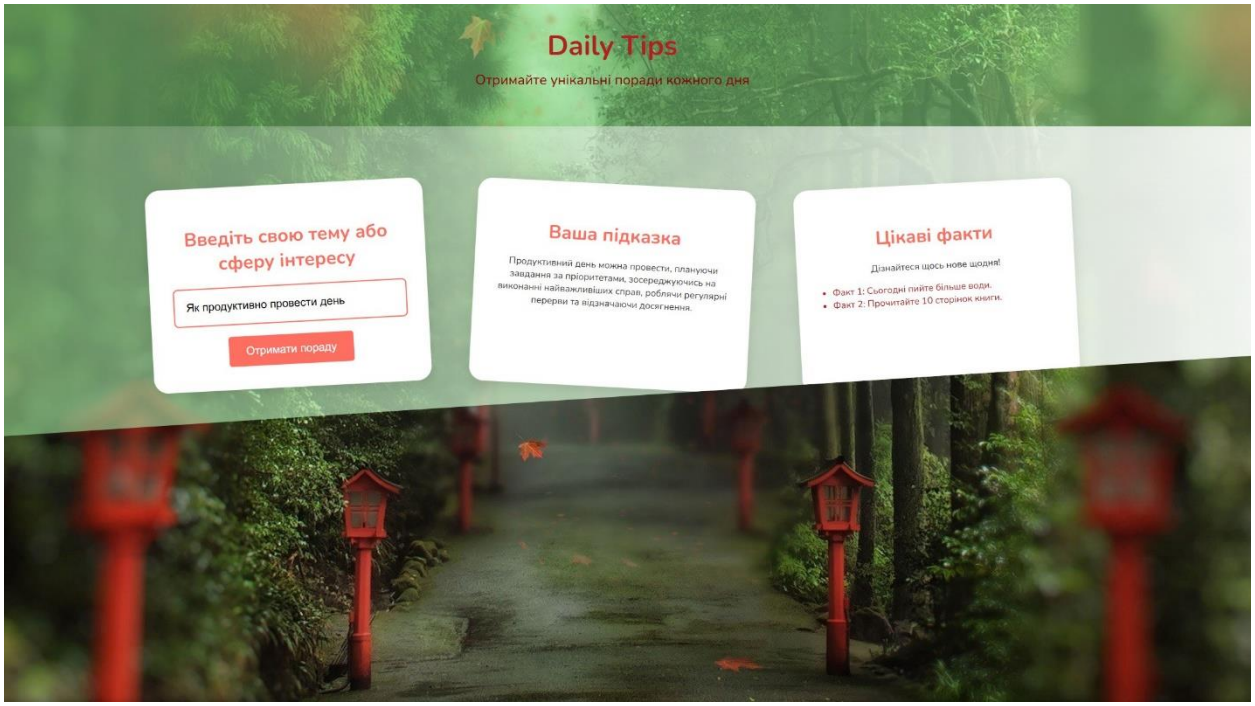


Рисунок 3.8 – Демонстрація функціональної частини проекту

3.3 Налаштування інструментів для порівняння проектів

У цьому пункті роботи будуть налаштовуватись проекти для порівнянння їх між собою. Зокрема інтеграція механізму вимірювання часу виконання запитів, що дозволить оцінити продуктивність кожного з проектів не тільки за ефективністю а ще й з точки зору швидкості обробки користувацьких запросів.

Веб-додатки, що використовують ШІ для надання персоналізованих рекомендацій, мають велику важливість для поліпшення взаємодії з користувачем. Однак, швидкість обробки запитів є ключовим фактором, що визначає зручність та ефективність роботи з таким додатком. Тому для оцінки і порівняння ефективності проектів була реалізована система вимірювання часу, який витрачається на обробку користувацьких запитів в проектах.

В проектах була інтегрована система вимірювання часу виконання запитів. Для цього в коді кожного проекту використовується API performance.now()[8], який дозволяє точно виміряти час виконання операцій у мілісекундах.:

const startTime = performance.now(); Початок відліку часу.

const endTime = performance.now(); Завершення відліку часу.

Таймери фіксують час на двох етапах:

– до відправлення запитіу: коли користувач вводить дані;

– після отримання відповіді: коли сервер обробляє запит і надсилає відповідь.

Це дозволяє точно виміряти час витрачений на обробку запиту від початку і до кінця. Таким чином користувач може побачити, скільки часу витратила система на обробку його запиту в кожному з проектів.

4 ЕКСПЕРИМЕНТАЛЬНЕ ПОРІВНЯННЯ ТА АНАЛІЗ

4.1 Проведення тестування часу обробки запитів веб-додатків

У цьому пункті роботи детально описується процес тестування часу обробки запитів для веб-додатків, які були розроблені для надання персоналізованих рекомендацій. Метою тестування є визначення продуктивності кожного з проектів, а саме вимірювання часу, необхідного для обробки запитів користувачів, а також порівняння результатів, отриманих від кожного додатку.

Тестування проводилося для кожного веб-додатку за наступною процедурою:

– рапуск веб-додатку: кожен додаток запускається в тестовому середовищі, на якому проводиться вимірювання часу. Для забезпечення однакових умов тестування використовувалися однакові вхідні дані (вподобання користувача для отримання рекомендацій або підказки);

– введення запиту користувачем: користувач вводить свої побажання щодо книги (для проекту з рекомендаціями книг) або іншу інформацію для генерації підказки (для проекту з підказками). Після цього запит відправляється на сервер для обробки;

– вимірювання часу: після відправлення запиту починається вимірювання часу до отримання відповіді. Час, що витрачається на обробку запиту, фіксується з точністю до мілісекунд;

– обробка запиту сервером: сервер отримує запит, виконує необхідні операції і надсилає відповідь;

– оцінка результатів: після отримання відповіді від сервера відображається час виконання запиту разом з результатом. Користувач також може побачити, скільки часу витратила система на обробку запиту.

Для тестування проекту з рекомендаціями книг було використано декілька сценаріїв:

– користувач обирає жанр книги з переліку, та нажимає на кнопку «Підібрати», далі веб-додаток обробляє запит, та виводить результат запиту, а саме фотографію обкладинки книги, назву, автора та короткий опис книги;

– користувач вводить жанр книги, свої уподобання щодо авторів або інші характеристики які допоможуть ШІ надіслати персоналізовану рекомендацію, запит обробляється через сервер на якому працює модель ШІ, та виводить результат.

Виміряний час виконання цього запиту дозволяє оцінити наскільки швидко система обробила запит.

Для проекту з підказками було проведено подібне тестування але система була менш складною:

– користувач на сайті обирає категорію підказок з переліку, далі нажимає на кнопку «Отримати пораду», веб-додаток обробляє запит та виводе результат на сторінку

– користувач вводить своє хобі, або вводить запит який стосується певної теми чи потребує підказки, далі запит обробляється через сервер на якому працює ШІ, і виводить відповідь яку надіслала ШІ.

4.2 Порівняння отриманих результатів за визначеними критеріям

Проведене тестування дозволило отримати важливі дані для оцінки продуктивності проектів. Зазначені результати тестування показують, що проекти ефективно обробляють запити користувача.

Нижче я склав таблицю з порівнянням витрати часу кожного з проектів на обробку запитів, таблиця 4.1.

Таблиця 4.1 – Порівняння часу виконання запитів

Назва проекту	ProjectBook	Project_AI	Tipsday	TipsAI
Час виконання	0.5037 с.= 503.7 мс.	1.208 с.= 1208 мс.	0.5021 с.= 502.1 мс.	1.26 с.= 1260 мс.

В результаті проведення тестування для кожного проекту було визначено що для проектів без використання ШІ час виконання запиту складав в середньому 502.9 мс.(0.5029 с.). Це пов’язано з тим що веб-додатку не потрібно відправляти запит на сервер, і чекати на відповідь, тому що відповіді він має у себе в масиві. А для проектів з інтегрованим ШІ у веб-додаток середній час виконання запитів склав 1.234 мс.(1,234 с.), що свідчить про більшу складність системи котра потребує більше часу для аналізу, обробки, відправлення запиту на сервер, та отримання відповіді від серверу.

Ось як це виглядає на прикладі проекту Tipsday, рис. 4.1.

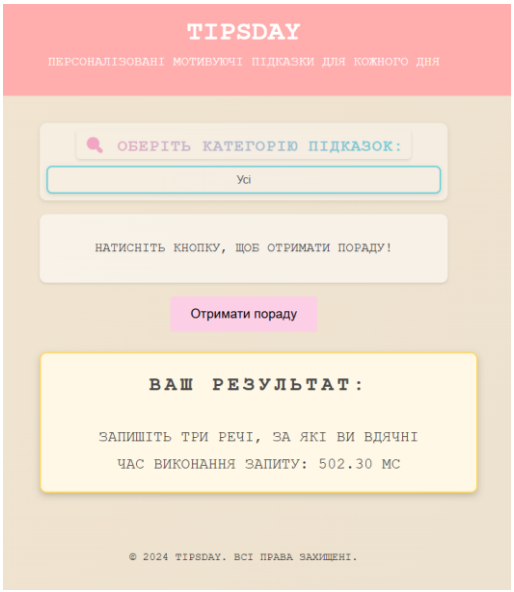


Рисунок 4.1 – Приклад реалізації таймеру виконання запиту

Ці результати можуть бути використані для подальшої оптимізації обох систем, що дозволить підвищити ефективність роботи веб-додатків і покращити взаємодію з користувачем, зокрема через зменшення часу очікування відповіді.

ВИСНОВКИ

У процесі виконання кваліфікаційнох роботи було досягнуто поставлених цілей, що полягали у дослідженні використання штучного інтелекту для розробки клієнтських інтерфейсів веб-додатків та порівняння підходів до створення традиційних і інтегрованих з ШІ інтерфейсів.

Було розроблено усього 4 проекта, два з яких розроблялись традиційним підходом до побудови інтерфейсів, а наступні два з залученням моделей штучного інтелекту. Також було проведено порівняльний аналіз ефективності цих рішень за часом обробки запитів.

Спочатку я розробив традиційний інтерфейс для веб-додатку, який рекомендував книги, я назвав його «Project Book». Він використовув заздалегідь запрограмовані правила для вибору рекомендацій, ця система була простою в реалізації та швидкою у виконанні запитів. Але в неї був недолік в тому що функціональна складова цього веб-додатку була обмежена – усі рекомендації були статичними і не могли адаптуватися під усі вподобання користувачів.

Наступним кроком було створення веб-додатку з інтеграцією штучного інтелекту під назвою «Project_AI». Для реалізації цього проекту я використав модель від OpenAI під назвою GPT-4o mini[9], яка може генерувати персоналізовані рекомендації на основі запитів користувача. Саме цей підхід значно підвищів якість взаємодії: користувач отримував унікальні пропозиції, що відповідали його вподобанням. Однак інтегрування ШІ викликало деякі складнощі у вигляді вибору API з доступом до моделі GPT, але я знайшов бібліотку g4f яка предоставляє вільний доступ до моделей ШІ.

Для переконання в ефективності системи, я впровадив таймери для вимірювання часу обробки запитів як для традиційної розробки так і з інтеграцією ШІ. Результати тестування були очікуваними, традиційна розробка веб-додатку демонструвала швидкість роботи завдяки своїй простоті

і відсутності потреби відправляти запит на сервер. Водночас веб-додаток з інтеграцією ШІ потребував більше часу для відображення результатів запиту через складні алгоритми аналізу даних і обчислень. Це показало, що інтеграція ШІ безумовно підвищує функціональність але й вимагає додаткових ресурсів.

Далі на черзі був проект під назвою «Tipsday» його я розробляв традиційним способом, цей веб-додаток був створений для мотиваційних порад. Його функціональність базувалася на заздалегіть визначених категоріях і списках підказок, які користувач міг отримувати за запитом. Простота реалізації дозволила забезпечити високу швидкість роботи. Однак ця система була також обмежена фіксованим наобором порад, що так само не дозволяло забеспечити індивідуальний підхід до користувача.

Крайній проект з використанням ШІ «TipsAI» створювався для генераціх мотиваційних порад. Модель ШІ аналізувала введений користувачем текст, намагаючись зрозуміти індивідуальний запит та створити відповідь на основі індивідуальних потреб. Це звичайно розширило можливості додатка, але й вплинуло на швидкість обробки запитів через складний алгоритм. Для аналізу продуктивності також було додано таймер, який дозволив порівняти швидкість обробки запитів між «Tipsday» і «TipsAI». Результати показали той же результат що і минулі два проекти, а саме те що традиційний підхід працює швидше, але інтеграція ШІ забеспечує кращу адаптивність і індивідуальність, хоч і вимагає додаткових ресурсів.

Загалом, ці проекти стали чудовою платформою для порівняння підходів до розробки веб-додатків. Традиційні методи виявилися швидкішими і більш простими в реалізації, але значно поступалися у функціональності. Також ця робота підтвердила що штучний інтелект відкриває значно більші можливості для створення веб-додатків попри те що він вимагає додаткові ресурси.

Цей досвід дозволив не лише оволодіти новими інструментами та технологіями, а й краще зрозуміти, як підбирати рішення для створення клієнтських інтерфейсів залежно від потреб користувача.

ПЕРЕЛІК ПОСИЛАНЬ

1. Ertel W. Introduction to Artificial Intelligence. United States of America, 2018. 383 с.

2. Савченко А.С., Синельніков О.О. Методи та системи штучного інтелекту : навч.посіб. Київ, 2017. 187 с.

3. Ducket J. HTML & CSS: Design and Build Websites. Boston, 2011. 490 p.

4. Eric A. Meyer. CSS: The Definitive Guide. Boston, 2023. 536 p.

5. Sierra K., Bates B. Head First Java. Chicago, 2022. 720 p.

6. Grinberg M. Flask Web Development: Developing Web Applications with Python. California, 2016. 258 с.

7. Githuv. URL: https://github.com/xtekky/gpt4free/releases (дата звернення : 11.10.2024)

8. Masse M. REST API Design Rulebook. Dallas, 2011. 116 с.

9. Caelen O., Blette M.A. Developing Apps with GPT-4 and ChatGPT: Build Intelligent Chatbots, Content Generators, and More. Chicago, 2023. 155 с.

ДОДАТОК А

Project Book

Index.html:

```
<!DOCTYPE html>
<div id="recommendation">
   <p id="recommendation-text">Тут з'явиться рекомендація.</p>
   <p id="execution-time">Час виконання запиту з'явиться тут.</p>
</div>
<html lang="uk">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Програма вибору книг</title>
   <link rel="stylesheet" href="styles.css">
</head>
<body>
   <div class="background">
      <!-- Навигация -->
      <nav class="navbar">
         <ul>
            <li><a href="#">Головна</a></li>
            <li><a href="#">Про нас</a></li>
            <li><a href="#">Контакти</a></li>
         </ul>
      </nav>
   </div>
   <!-- Головний заголовок -->
   <header class="main-header">
      <h1>Програма вибору книг для читання</h1>
      <p>Знайди свою наступну улюблену книгу!</p>
   </header>

   <!-- Розділ вибору жанру -->
   <section class="genre-selection">
      <h2>Оберіть жанр</h2>
      <div class="genre-buttons">
         <button class="fantasy" data-genre="fantasy">Фантастика</button>
         <button class="classics" data-genre="classics">Класика</button>
         <button class="detective" data-genre="detective">Детектив</button>
         <button class="romance" data-genre="romance">Романтика</button>
         <button class="science" data-genre="science">Наукова література</button>
         <button class="adventure" data-genre="adventure">Пригоди</button>
      </div>
   </section>

   <!-- Рекомендації -->
```

```
    <section class="recommendations" id="recommendations">
        <h2>Рекомендації</h2>
        <div id="book-list" class="book-list"></div>
        <div class="book-list">
            <div class="book-item">
                <img src="photo2.jpg" alt="Book Cover">
                <h3>Назва книги</h3>
                <p>Автор: Ім'я Автора</p>
                <p>Короткий опис книги, щоб зацікавити читача...</p>
                <button class="details">Підібрати</button>
            </div>
           <!---->
        </div>
    </section>

    <!-- Підвал -->
    <footer class="footer">
        <div class="social-links">
            <a href="#">Facebook</a> | <a href="#">Twitter</a> | <a href="#">Instagram</a>
        </div>
        <p>© 2024 Програма вибору книг для читання</p>
    </footer>
    <script src="script.js" charset="UTF-8" defer></script>
</body>
</html>
```

Script.js:

```
// Список книг по жанрам (на украинском языке)
let selectedGenre = null;
const booksByGenre = {
    fantasy: {
        title: "Володар Перснів ",
        author: "Дж. Р. Р. Толкін",
        description: "Епічна фентезі-пригода у світі Середзем'я.",
        image: "volodar.jpg", // Добавьте путь к изображению

    },

    classics: {
        title: "Гордість і упередження",
        author: "Джейн Остін",
        description: "Класичний роман про любов, суспільство і моральні дилеми.",
        image: "gordist.jpg",
    },
    detective: {
        title: "Шерлок Холмс",
        author: "Артур Конан Дойл",
        description: "Захопливі детективні розслідування відомого детектива.",
        image: "holms.jpg",
    },
    romance: {
        title: "Ніч у Лісабоні",
```

```
        author: "Еріх Марія Ремарк",
        description: "Романтична історія про долю та кохання під час війни.",
        image: "night.jpg",
    },
    science: {
        title: "Коротка історія часу",
        author: "Стівен Хокінг",
        description: "Науковий погляд на Всесвіт і його таємниці.",
        image: "history.jpg",
    },
    adventure: {
        title: "Навколо світу за 80 днів",
        author: "Жуль Верн",
        description: "Захоплюючі пригоди у всьому світі.",
        image: "navkolo.jpg",
    },
};

function fetchRecommendation() {
    if (!selectedGenre) {
        alert("Будь ласка, оберіть жанр!");
        return;
    }

    const startTime = performance.now(); // Время начала

    // Имитация ответа сервера
    setTimeout(() => {
        const endTime = performance.now(); // Время завершения
        const elapsedTime = (endTime – startTime).toFixed(2); // Вычисляем затраченное время

        // Обновляем текст рекомендации
        document.getElementById('recommendation-text').innerText =
            `Рекомендація для жанру "${selectedGenre}" успішно завантажена.`;

        // Отображаем время выполнения
        document.getElementById('execution-time').innerText = `Час виконання запиту: ${elapsedTime} мс`;
    }, 500); // Симуляция задержки запроса (500 мс)
}
// Обработчик нажатия на кнопки жанров
document.querySelectorAll('.genre-buttons button').forEach(button => {
    button.addEventListener('click', () => {
        selectedGenre = button.dataset.genre;

        // Убираем выделение с других кнопок
        document.querySelectorAll('.genre-buttons button').forEach(btn => {
            btn.style.backgroundColor = ''; // Сброс цвета
        });

        // Выделяем выбранную кнопку
        button.style.backgroundColor = '#82a3c3';
```

```
    });
});
document.querySelector('.details').addEventListener('click', () => {
    fetchRecommendation();
});

// Обработчик нажатия на кнопку "Підібрати"
document.querySelector('.details').addEventListener('click', () => {
    if (!selectedGenre) {
        alert("Будь ласка, оберіть жанр!");
        return;
    }

    // Получаем данные о книге для выбранного жанра
    const book = booksByGenre[selectedGenre];

    // Находим существующий элемент для книги
    const bookItem = document.querySelector('.book-item');

    // Обновляем содержимое
    bookItem.querySelector('h3').textContent = book.title;
    bookItem.querySelector('p:nth-of-type(1)').textContent = `Автор: ${book.author}`;
    bookItem.querySelector('p:nth-of-type(2)').textContent = book.description;
    bookItem.querySelector('img').src = book.image;
    bookItem.querySelector('img').alt = `Обкладинка книги: ${book.title}`;
});
```

Styles.css:

```
/* Основні стилі */
@import
url('https://fonts.googleapis.com/css2?family=Nunito:wght@400;600;700&display=swap');
body {
    height: 100%;
    display: flex;
    flex-direction: column;
    font-family: 'Nunito', sans-serif; /* Новый шрифт */
    margin: 0;
    padding: 0;
    color: #003366;
    position: relative; /* Щоб фон не перекривався іншими елементами */
}

.background {
    background-image: url('photo1.jpg');
    background-size: cover; /* Масштабування зображення, щоб покрити всю область */
    background-position: center; /* Центрування зображення */
    background-repeat: no-repeat; /* Запобігання повторенню зображення */
    height: 100%;
    width: 100%;
    position: absolute; /* Робимо фонове зображення абсолютним */
    z-index: -1; /* Поміщуємо його за всі інші елементи */
    top: 0;
```

```
    left: 0;
}

/* Контейнер для всіх секцій */
.container {
    max-width: 960px; /* Максимальна ширина */
    margin: 0 auto; /* Центрування контейнера */
    padding: 0 20px; /* Відступи всередині контейнера */
}

/* Навігація */
.navbar {
    background-color: rgba(210, 180, 140, 0.8); /* Світло-коричневий напівпрозорий фон */
    text-align: center;
    padding: 8px 180px;
    position: relative;
    z-index: 1;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1); /* Легка тінь */
    margin-bottom: 20px; /* Відступ знизу */
    max-width: 800px;
    margin: 0 auto;
}

    .navbar ul {
        list-style: none;
        margin: 0;
        padding: 0;
    }

    .navbar li {
        display: inline;
        margin: 0 10px;
    }

    .navbar a {
        text-decoration: none;
        color: #003366;
        font-size: 16px;
        font-weight: 600;
        transition: color 0.3s;
    }

        .navbar a:hover {
            color: #0056b3; /* При наведенні змінюємо колір тексту */
        }

/* Головний заголовок */
.main-header {
    background-color: rgba(210, 180, 140, 0.8); /* Світло-коричневий напівпрозорий фон */
    padding: 15px 20px;
    text-align: center;
    flex: 1;
```

```
    margin-top: 10px; /* Відступ зверху, щоб не перекривав navbar */
    max-width: 800px;
    margin: 38px auto;
}

    .main-header h1 {
        font-size: 32px;
        font-weight: bold;
        text-transform: uppercase;
    }

    .main-header p {
        font-size: 20px;
        font-style: italic;
        color: #666666;
    }

/* Розділ вибору жанру */
.genre-selection {
    padding: 5px 30px;
    background-color: rgba(255, 248, 220, 0.8); /* Світло-коричневий напівпрозорий фон */
    text-align: center;
    text-transform: uppercase;
    max-width: 900px;
    margin: 35px auto;
}

    .genre-selection h2 {
        font-size: 24px;
        font-weight: bold;
        color: #003366;
    }

/* Кнопки жанрів */
.genre-buttons {
    display: flex;
    justify-content: center;
    gap: 10px;
}

button {
    padding: 8px 20px;
    border: none;
    border-radius: 8px;
    cursor: pointer;
    font-size: 14px;
    font-weight: 600;

    transition: background-color 0.3s, transform 0.3s;
    background-color: rgba(255, 255, 255, 0.8); /* Полупрозорий фон */
}
```

```
    button.fantasy {
        background-color: rgba(255, 250, 205, 0.8); /* Світло-жовтий */
        color: #003366;
    }

    button.classics {
        background-color: rgba(230, 255, 230, 0.8); /* Світло-зелений */
        color: #003366;
    }

    button.detective {
        background-color: rgba(240, 240, 240, 0.8); /* Світло-сірий */
        color: #003366;
    }

    button.romance {
        background-color: rgba(255, 230, 230, 0.8); /* Світло-рожевий */
        color: #003366;
    }

    button.science {
        background-color: rgba(230, 242, 255, 0.8); /* Світло-голубий */
        color: #003366;
    }

    button.adventure {
        background-color: rgba(255, 230, 204, 0.8); /* Світло-оранжевий */
        color: #003366;
    }

    /* Ефект натискання на кнопку */
    button:hover {
        background-color: #82a3c3;
        color: #FFFFFF;
        transform: scale(1.05); /* Легкий ефект збільшення */
    }

/* Рекомендації */
.recommendations {
    padding: 30px 50px;
    background-color: rgba(210, 180, 140, 0.8); /* Світло-коричневий напівпрозорий фон */
    max-width: 600px;
    margin: 20px auto;
}

    .recommendations h2 {
        font-size: 24px;
        font-weight: bold;
        color: #003366;
        text-transform: uppercase; /* Всі букви заглавні */
        text-align: center;
    }
```

```
.book-list {
    display: flex;
    justify-content: center;
    gap: 10px;
    flex-wrap: wrap;
}

.book-item {
    text-align: center;
    max-width: 180px;
}


    .book-item img {
        width: 180px;
        height: 250px;
        object-fit: cover;
    }

    .book-item h3 {
        font-size: 18px;
        font-weight: bold;
        color: #003366;
    }

    .book-item p {
        font-size: 14px;
        color: #666666;
    }

.details {
    background-color: #003366;
    color: #FFFFFF;
    padding: 8px 10px;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    transition: background-color 0.3s;
}

    .details:hover {
        background-color: #0056b3;
    }

/* Підвал */
.footer {
    background-color: rgba(210, 180, 140, 0.8);
    color: #FFFFFF;
    text-align: center;
    padding: 5px 0; /* Увеличиваем отступы */
    margin-top: 152px; /* Это гарантирует, что футер будет на дне */
```

```
}

   .footer .social-links {
      display: flex;
      justify-content: center;
      gap: 5px;
      margin-top: 5px; /* Подаем отступы для элементов */
   }

      .footer .social-links a {
         text-decoration: none;
         color: #FFFFFF;
         font-size: 15px;
         transition: color 0.3s;
      }

         .footer .social-links a:hover {
            color: #0056b3;
            font-size: 2px;
         }
```

Tipsday

Index.html:

```
<!DOCTYPE html>
<html lang="uk">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Tipsday – Отримай натхнення</title>
   <link rel="stylesheet" href="styles.css">
</head>
<body>
   <header>
      <h1>Tipsday</h1>
      <p>Персоналізовані мотивуючі підказки для кожного дня</p>
   </header>
   <main>
      <div class="container">
         <div class="category-select">
            <label for="category">
               <span class="label-text">🔍 Оберіть категорію підказок:</span>
            </label>
            <select id="category" class="styled-select">
               <option value="all">Усі</option>
               <option value="health">Здоров'я</option>
               <option value="study">Навчання</option>
               <option value="relax">Відпочинок</option>
            </select>
         </div>
         <div id="tip-display">
            <p class="tip-text">Натисніть кнопку, щоб отримати пораду!</p>
```

```
        </div>
        <button id="get-tip">Отримати пораду</button>
        <div id="result-container" class="result-box">
          <h3>Ваш результат:</h3>
          <p id="result-text">Тут з’явиться ваша персоналізована підказка.</p>
        </div>
      </div>
    </main>
    <footer>
      <p>&copy; 2024 Tipsday. Всі права захищені.</p>
    </footer>
    <script src="script.js"></script>
</body>
</html>
```

Script.js:

```
// script.js

const tips = [
    { text: "Пийте більше води щодня", category: "health" },
    { text: "Зробіть ранкову зарядку", category: "health" },
    { text: "Прочитайте 10 сторінок книги сьогодні", category: "study" },
    { text: "Заплануйте завдання на день", category: "study" },
    { text: "Погуляйте на свіжому повітрі", category: "relax" },
    { text: "Подивіться улюблений фільм", category: "relax" },
    { text: "Проведіть час із родиною", category: "relax" },
    { text: "Спробуйте новий рецепт", category: "health" },
    { text: "Запишіть три речі, за які ви вдячні", category: "study" },
    { text: "Зробіть перерву на 15 хвилин", category: "relax" },
];

const getTipButton = document.getElementById("get-tip");
const resultText = document.getElementById("result-text");
const tipDisplay = document.getElementById("tip-display");
const categorySelect = document.getElementById("category");

// Добавляем блок для отображения времени выполнения
const executionTime = document.createElement("p");
executionTime.id = "execution-time";
executionTime.innerText = "Час виконання запиту з'явиться тут.";
document.querySelector("#result-container").appendChild(executionTime);

getTipButton.addEventListener("click", () => {
    const startTime = performance.now(); // Начало отсчета времени

    const selectedCategory = categorySelect.value;
    let filteredTips = tips;

    // Фильтрация по категории
    if (selectedCategory !== "all") {
        filteredTips = tips.filter((tip) => tip.category === selectedCategory);
    }
```

```
    // Симуляция времени выполнения с задержкой
    setTimeout(() => {
        // Выбор случайной подсказки
        const randomIndex = Math.floor(Math.random() * filteredTips.length);
        const randomTip = filteredTips[randomIndex];

        // Отображаем подсказку
        resultText.innerText = randomTip.text;

        // Замер времени выполнения
        const endTime = performance.now();
        const elapsedTime = (endTime – startTime).toFixed(2); // Время в миллисекундах

        // Обновляем текст времени выполнения
        executionTime.innerText = `Час виконання запиту: ${elapsedTime} мс`;
    }, 500); // Искусственная задержка 500 мс
});
getTipButton.addEventListener("click", () => {
    const selectedCategory = categorySelect.value;
    let filteredTips = tips;

    // Фільтруємо по категорії
    if (selectedCategory !== "all") {
        filteredTips = tips.filter((tip) => tip.category === selectedCategory);
    }

    // Вибираємо випадкову пораду
    const randomIndex = Math.floor(Math.random() * filteredTips.length);
    const randomTip = filteredTips[randomIndex];

    // Відображаємо пораду в правильном элементе
    resultText.innerText = randomTip.text;
});
```

Styles.css:

```
/* styles.css */


body {
    font-family: 'Courier New', Courier, monospace;
    margin: 0;
    padding: 0;
    background: linear-gradient(120deg, #f6f5f4, #ded9d5);
    color: #333;
    text-align: center;
    text-transform: uppercase;
}

.container {
    max-width: 600px;
```

```
    margin: 0 auto;
    padding: 20px;
}

.label-text {
    font-size: 1.5em;
    font-weight: 600;
    color: #4a4a4a;
    text-transform: uppercase;
    letter-spacing: 0.05em;
    background: linear-gradient(90deg, #f8a5c2, #63cdda);
    -webkit-background-clip: text;
    -webkit-text-fill-color: transparent;
    display: inline-block;
    margin-bottom: 10px;
    padding: 5px 10px;
    border-radius: 5px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}

.tip-text {
    font-size: 1.2em;
    color: #333;
}

header {
    background: #ffadad;
    padding: 20px;
    color: #fff;
}

    header h1 {
        margin: 0;
        font-size: 2.5em;
    }

    header p {
        font-size: 1.2em;
        margin-top: 10px;
    }

main {
    padding: 20px;
}

.category-select {
    margin-bottom: 20px;
    padding: 10px;
    border-radius: 10px;
    background: #ffffff5c;
    box-shadow: 0 2px 6px rgba(0, 0, 0, 0.1);
}
```

```
button {
    background-color: #ffd6a5;
    border: none;
    padding: 15px 30px;
    border-radius: 5px;
    font-size: 1.2em;
    cursor: pointer;
    transition: background-color 0.3s ease;
}

    button:hover {
        background-color: #fdcfe8;
    }

#tip-display {
    margin: 20px 0;
    font-size: 1em;
    padding: 20px;
    background: rgba(255, 255, 255, 0.5);
    border-radius: 10px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}

.styled-select {
    width: 100%;
    padding: 10px 15px;
    font-size: 1em;
    font-weight: 500;
    color: #4a4a4a;
    border: 2px solid #63cdda;
    border-radius: 8px;
    background: transparent;
    appearance: none;
    outline: none;
    box-shadow: 0 2px 6px rgba(0, 0, 0, 0.1);
    transition: all 0.3s ease;
    cursor: pointer;
    text-align: center;
}

    .styled-select:hover {
        border-color: #f8a5c2;
        box-shadow: 0 4px 8px rgba(0, 0, 0, 0.15);
    }

    .styled-select:focus {
        border-color: #fd9644;
        box-shadow: 0 0 10px rgba(253, 150, 68, 0.5);
    }

    .styled-select option {
```

```
        font-size: 1em;
        color: #333;
        background: rgba(255, 255, 255, 0.5);
        padding: 5px;
    }
        .styled-select option:hover,
        .styled-select option:focus {
            background: rgba(255, 255, 255, 0.2);
        }

/* Ñåêö³ÿ äëÿ ðåçóëüòàòó */
.result-box {
    margin: 30px auto;
    padding: 20px;
    width: 100%;
    background: #fff8e6;
    border: 2px solid #ffd966;
    border-radius: 10px;
    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.2);
    text-align: center;
    transition: transform 0.3s ease, box-shadow 0.3s ease;
}

    .result-box:hover {
        transform: scale(1.02);
        box-shadow: 0 6px 14px rgba(0, 0, 0, 0.3);
    }

    .result-box h3 {
        font-size: 1.8em;
        font-weight: bold;
        color: #4a4a4a;
        margin-top: 10px;
        margin-bottom: 40px;
        text-transform: uppercase;
        letter-spacing: 0.2em;
    }

    .result-box p {
        font-size: 1.4em;
        font-weight: 500;
        color: #333;
        margin: 0;
        line-height: 1.8;
    }

body {
    padding-bottom: 50px;
    background: linear-gradient(120deg, #f4e7d1, #eae0cf);
}
```

ДОДАТОК Б

Project_AI

Index.html:

```
<!DOCTYPE html>
<html lang="uk">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Рекомендації книг</title>
   <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
</head>
<body>
   <div class="decorative-circle"></div>
   <div class="decorative-circle"></div>
   <div class="decorative-circle"></div>
   <div class="decorative-circle"></div>

   <div class="container">
      <header>
         <h1>КНИЖКОВІ РЕКОМЕНДАЦІЇ</h1>
         <p class="description">
            Ця програма допоможе підібрати найкращі книги відповідно до ваших вподобань.
            Введіть свої побажання, і ми підберемо для вас найкращі рекомендації!
         </p>
      </header>
      <main>
         <form method="POST" action="/recommend">
            <input type="text" name="preferences" id="preferences" placeholder="Введіть свої
побажання щодо книги..." required>
            <button type="button" onclick="getRecommendation()">Отримати
рекомендацію</button>
         </form>

         <div id="recommendation-text"></div>
         <div id="time-taken"></div> <!-- Елемент для відображення часу виконання -->

         {% if recommendation %}
         <div class="result">
            <strong>Рекомендація:</strong>
            <p>{{ recommendation }}</p>
         </div>
         {% endif %}

         {% if error %}
         <div class="error">
            <strong>Помилка:</strong>
```

```
            <p>{{ error }}</p>
          </div>
          {% endif %}
        </main>
    </div>

    <footer>
        © 2024 Рекомендації книг | Всі права захищені.
    </footer>
    <script src="static/script.js"></script>
</body>
</html>
```

Styles.css:

```
/* Базовые стили */
body {
    font-family: 'Nunito', sans-serif;
    margin: 0;
    padding: 0;
    background: linear-gradient(135deg, #dcd0b0, #b9a98b); /* Темный фон, цвет книжной страницы */
    color: #333;
    min-height: 100vh;
    display: flex;
    flex-direction: column;
    justify-content: center;
    overflow: hidden;
    position: relative;
}

/* Декоративные круги */
.decorative-circle {
    position: absolute;
    border-radius: 50%;
    animation: float 8s infinite ease-in-out;
    box-shadow: 0px 4px 8px rgba(0, 0, 0, 0.05);
}

.decorative-circle:nth-child(1) {
    width: 120px;
    height: 120px;
    top: 10%;
    left: 15%;
    background: rgba(248, 218, 218, 0.5);
    animation-delay: 0s;
}

.decorative-circle:nth-child(2) {
    width: 180px;
    height: 180px;
    top: 70%;
    left: 5%;
```

```
    background: rgba(201, 231, 255, 0.5);
    animation-delay: 1s;
}

.decorative-circle:nth-child(3) {
    width: 90px;
    height: 90px;
    top: 40%;
    left: 85%;
    background: rgba(252, 182, 193, 0.5);
    animation-delay: 2s;
}

.decorative-circle:nth-child(4) {
    width: 200px;
    height: 200px;
    top: 90%;
    left: 70%;
    background: rgba(183, 192, 241, 0.5);
    animation-delay: 3s;
}

.decorative-circle:nth-child(5) {
    width: 130px;
    height: 130px;
    top: 25%;
    left: 55%;
    background: rgba(250, 224, 166, 0.5);
    animation-delay: 4s;
}

/* Полосы */
.decorative-stripe {
    position: absolute;
    width: 100%;
    height: 10px;
    background: linear-gradient(90deg, rgba(255, 235, 209, 0.3), rgba(216, 236, 255, 0.3));
    opacity: 0.4;
    animation: moveStripe 10s infinite linear;
}

.decorative-stripe:nth-child(6) {
    top: 20%;
    animation-delay: 0s;
}

.decorative-stripe:nth-child(7) {
    top: 50%;
    animation-delay: 2s;
}

.decorative-stripe:nth-child(8) {
```

```
    top: 80%;
    animation-delay: 4s;
}

/* Анимации */
@keyframes float {
    0%, 100% {
        transform: translateY(0);
    }
    50% {
        transform: translateY(-15px);
    }
}

@keyframes moveStripe {
    0% {
        transform: translateX(-100%);
    }
    100% {
        transform: translateX(100%);
    }
}

/* Контейнер */
.container {
    max-width: 800px;
    margin: 0 auto;
    padding: 40px 30px;
    text-align: center;
    background-color: rgba(255, 255, 255, 0.9); /* Легкая полупрозрачность */
    box-shadow: 0px 4px 6px rgba(0, 0, 0, 0.2);
    border-radius: 15px;
    position: relative;
    z-index: 10;
}

/* Заголовок */
header h1 {
    font-size: 3.5em; /* Увеличенный размер шрифта */
    text-transform: uppercase;
    color: #2c3e50;
    margin-bottom: 15px;
    letter-spacing: 5px;
    text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.3);
}

header .description {
    font-size: 1.6em; /* Увеличенный размер шрифта для описания */
    color: #34495e;
    margin-bottom: 30px;
    line-height: 1.7;
}
```

```
/* Поля формы */
form input[type="text"] {
    width: calc(100% – 20px);
    padding: 15px;
    font-size: 1.2em; /* Увеличенный размер шрифта */
    margin: 15px 10px 15px 0; /* Отодвигаем влево */
    border: 1px solid #bdc3c7;
    border-radius: 8px;
}

/* Кнопка */
form button {
    background-color: #82b1ff;
    color: #fff;
    font-size: 1.2em; /* Увеличенный размер шрифта */
    padding: 12px 25px;
    border: none;
    border-radius: 8px;
    cursor: pointer;
    box-shadow: 2px 4px 6px rgba(0, 0, 0, 0.2);
    transition: background-color 0.3s ease, transform 0.2s ease;
}

form button:hover {
    background-color: #6fa9ff;
    transform: translateY(-2px);
}

/* Результаты */
.result, .error {
    margin-top: 25px;
    padding: 15px;
    border-radius: 8px;
    font-size: 1.2em; /* Увеличенный размер шрифта */
}

.result {
    background-color: #f0f9f9;
    color: #27ae60;
    border: 1px solid #27ae60;
}

.error {
    background-color: #fbeae9;
    color: #c0392b;
    border: 1px solid #c0392b;
}

/* Футер */
footer {
    margin-top: 50px;
```

```
    font-size: 1em;
    color: #7f8c8d;
    text-align: center;
}
```

Script.js

```
function getRecommendation() {
    const preferences = document.getElementById('preferences').value.trim();
    if (!preferences) {
        document.getElementById('recommendation-text').innerText = 'Будь ласка, введіть свої уподобання.';
        return;
    }

    // Запуск таймера
    const startTime = Date.now();

    fetch('/recommend', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
        },
        body: JSON.stringify({ preferences }),
    })
        .then(response => response.json())
        .then(data => {
            // Зупинка таймера та розрахунок часу виконання
            const endTime = Date.now();
            const duration = (endTime – startTime) / 1000; // в секундах

            document.getElementById('recommendation-text').innerText = data.recommendation || 'Сталася помилка. Спробуйте знову.';
            document.getElementById('time-taken').innerText = `Час виконання запиту: ${duration} секунд`;

            if (data.error) {
                document.getElementById('recommendation-text').innerText = `Помилка: ${data.error}`;
            } else {
                document.getElementById('recommendation-text').innerText = data.recommendation;
            }
        })
        .catch((error) => {
            console.error('Error:', error);
            document.getElementById('recommendation-text').innerText = 'Сталася помилка. Спробуйте знову.';
        });
}
```

App.py:

```
from flask import Flask, request, jsonify, render_template
import logging
```

```
from g4f.client import Client
import asyncio
from asyncio import WindowsSelectorEventLoopPolicy

# Устанавливаем политику event loop для Windows
asyncio.set_event_loop_policy(WindowsSelectorEventLoopPolicy())

# Создаем клиент для API
client = Client()

# Инициализация Flask-приложения
app = Flask(__name__)

# Главная страница
@app.route('/')
def home():
    return render_template('index.html')

# Рекомендация книги
@app.route('/recommend', methods=['POST'])
def recommend_book():
    user_input = request.json.get('preferences')  # Исправлено получение JSON данных
    try:
        # Отправляем запрос к модели через g4f
        response = client.chat.completions.create(
            model="gpt-4o-mini",
            messages=[
                {"role": "user", "content": user_input}
            ],
        )

        # Правильный способ извлечения ответа
        recommendation = response.choices[0].message.content.strip()
        return jsonify(recommendation=recommendation)

    except Exception as e:
        # Логирование ошибки и отображение сообщения на странице
        logging.error(f"Ошибка API: {e}")
        return jsonify(error="Ошибка при обработке запроса. Проверьте настройки.")

# Запуск приложения
if __name__ == '__main__':
    logging.basicConfig(level=logging.DEBUG)
    app.run(debug=True)
```

TipsAI

Index.html:

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Daily Tips – Персоналізовані підказки</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
</head>
<body>
    <header class="main-header">
        <div class="logo">
            <h1>Daily Tips</h1>
            <p>Отримайте унікальні поради кожного дня</p>
        </div>
    </header>
    <main>
        <section class="content-section">
            <div class="floating-card" id="form-card">
                <h2>Введіть свою тему або сферу інтересу</h2>
                <input type="text" id="preferences" class="input-field" placeholder="Наприклад, продуктивність">
                <button id="get-tip" onclick="getRecommendation()">Отримати пораду</button>
            </div>
            <div class="floating-card" id="result-card">
                <h2>Ваша підказка</h2>
                <p id="recommendation-text">Тут з’явиться ваша персоналізована підказка.</p>
                <p id="time-taken">Час виконання запиту: -- секунд</p> <!-- Новий елемент для відображення часу виконання -->
            </div>
            <div class="floating-card" id="extra-card">
                <h2>Цікаві факти</h2>
                <p>Дізнайтеся щось нове щодня!</p>
                <ul>
                    <li>Факт 1: Сьогодні пийте більше води.</li>
                    <li>Факт 2: Прочитайте 10 сторінок книги.</li>
                </ul>
            </div>
        </section>
    </main>
    <footer>
        © 2024 Daily Tips | Всі права захищені.
    </footer>
    <script src="{{ url_for('static', filename='script.js') }}"></script>
</body>
</html>
```

Script.js:

```
function getRecommendation() {
    const preferences = document.getElementById('preferences').value.trim();
    if (!preferences) {
        document.getElementById('recommendation-text').innerText = 'Будь ласка, введіть свої уподобання.';
        return;
    }

    // Запуск таймера
```

```
    const startTime = Date.now();

    fetch('/recommend', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
        },
        body: JSON.stringify({ preferences }),
    })
    .then(response => response.json())
    .then(data => {
        // Зупинка таймера та розрахунок часу виконання
        const endTime = Date.now();
        const duration = (endTime – startTime) / 1000; // в секундах

        if (data.error) {
            document.getElementById('recommendation-text').innerText = `Помилка: ${data.error}`;
        } else {
            document.getElementById('recommendation-text').innerText = data.recommendation;
        }

        document.getElementById('time-taken').innerText = `Час виконання запиту: ${duration.toFixed(2)} секунд`;
    })
    .catch(error => {
        console.error('Error:', error);
        document.getElementById('recommendation-text').innerText = 'Сталася помилка. Спробуйте знову.';
    });
}
```

Styles.css:

```
/* styles.css */
@import url('https://fonts.googleapis.com/css2?family=Nunito:wght@400;700&display=swap');

body {
    font-family: 'Nunito', sans-serif;
    margin: 0;
    padding: 0;
    background: #fafafa;
    color: #333;
    text-align: center;
    display: flex;
    flex-direction: column;
    min-height: 100vh;
    background-image: url('photo.jpg');
    background-size: cover;
    background-position: center;
}

.main-header {
    background: rgba(144, 238, 144, 0.5);
```

```
    color: #fff;
    padding: 40px 20px;
    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.2);
    text-align: center;
}

.logo h1 {
    margin: 0;
    font-size: 3em;
    color: #b22222;

.logo p {
    font-size: 1.5em;
    margin-top: 10px;
    color: #8b0000;
}

.content-section {
    display: flex;
    flex-wrap: wrap;
    justify-content: center;
    padding: 60px 20px;
    background: linear-gradient(to right, rgba(144, 238, 144, 0.5), #fafafa); /* Ìÿãêèé çåëåíûé öâåò
ñ ïðîçðà÷íîñòüþ */
    clip-path: polygon(0 0, 100% 0, 100% 70%, 0 100%);
    flex: 1;
}

.floating-card {
    background: #fff;
    border-radius: 25px;
    box-shadow: 0 8px 20px rgba(0, 0, 0, 0.1);
    padding: 40px;
    max-width: 400px;
    width: 100%;
    text-align: center;
    transition: transform 0.3s ease, background 0.3s ease, color 0.3s ease;
    margin: 40px;
    transform: rotate(-3deg);
}

    .floating-card:nth-child(even) {
        transform: rotate(3deg);
    }

    .floating-card:hover {
        transform: rotate(0deg);
        background: #ff6f61;
        color: #fff;
    }

        .floating-card:hover h2,
```

```
        .floating-card:hover p,
        .floating-card:hover ul,
        .floating-card:hover li {
            color: #fff;
        }

h2 {
    font-size: 2em;
    margin-bottom: 20px;
    color: #ff6f61;
}

.input-field {
    width: calc(100% – 40px);
    padding: 20px;
    font-size: 1.2em;
    border: 2px solid #ff6f61;
    border-radius: 8px;
    outline: none;
    margin-bottom: 20px;
}

button {
    background-color: #ff6f61;
    color: #fff;
    border: none;
    padding: 15px 30px;
    border-radius: 5px;
    font-size: 1.2em;
    cursor: pointer;
    transition: background-color 0.3s ease;
}

    button:hover {
        background-color: #e64a19;
    }

footer {
    background: rgba(144, 238, 144, 0.5);
    color: #b22222;
    text-align: center;
    padding: 20px;
    box-shadow: 0 -4px 10px rgba(0, 0, 0, 0.2);
    margin-top: auto;
}

ul {
    text-align: left;
    list-style-type: disc;
    padding-left: 20px;
    margin: 0;
    color: #b22222;
```

```
}
```

Server.py:

```
from flask import Flask, render_template, request, jsonify
import logging
from g4f.client import Client  # Убедитесь, что g4f установлен и настроен корректно
import asyncio
from asyncio import WindowsSelectorEventLoopPolicy

# Устанавливаем политику event loop для Windows
asyncio.set_event_loop_policy(WindowsSelectorEventLoopPolicy())

# Создаем клиент для API
client = Client()

# Инициализация Flask-приложения
app = Flask(__name__)

# Главная страница
@app.route('/')
def home():
    return render_template('index.html')

# Рекомендация книги
@app.route('/recommend', methods=['POST'])
def recommend_book():
    data = request.get_json()
    user_input = data.get('preferences')  # Получаем предпочтения от пользователя
    try:
        # Отправляем запрос к модели через g4f
        response = client.chat.completions.create(
            model="gpt-4o-mini",  # Убедитесь, что модель поддерживается вашим клиентом
            messages=[
                {"role": "user", "content": user_input}
            ],
        )

        # Правильный способ извлечения ответа
        recommendation = response.choices[0].message.content.strip()
        return jsonify({'recommendation': recommendation})

    except Exception as e:
        # Логирование ошибки и отображение сообщения на странице
        logging.error(f"Ошибка API: {e}")
        return jsonify({'error': "Ошибка при обработке запроса. Проверьте настройки."}), 500

# Запуск приложения
if __name__ == '__main__':
    logging.basicConfig(level=logging.DEBUG)
    app.run(debug=True)
```