

**МІНІСТЕРСТВО ФІНАНСІВ УКРАЇНИ
ДЕРЖАВНИЙ ПОДАТКОВИЙ УНІВЕРСИТЕТ**

**ФАКУЛЬТЕТ ФІНАНСІВ ТА ЦИФРОВИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І
СИСТЕМ**

КВАЛІФІКАЦІЙНА РОБОТА

**на тему: РЕАЛІЗАЦІЯ ІНТЕЛЕКТУАЛЬНОГО АСИСТЕНТА ДЛЯ ВЕБМЕНЮ
ЗАКЛАДІВ ГРОМАДСЬКОГО ЗАРЧУВАННЯ**

Здобувача вищої освіти першого (бакалаврського) рівня,
IV курсу, групи ЦДБ-22-1, спеціальності 122 «Комп'ютерні науки»,
освітня програма «Технології цифрового дизайну»



(підпис)

Іващенко Дар'ї Сергіївни

(прізвище, ім'я, по батькові)

Науковий керівник



(підпис)

Ніжегородцев Владислав
Олександрович, канд. пед.
наук, доцент

(прізвище, ім'я, по батькові, науковий ступінь,
вчене звання)

Керівник освітньо-
професійної програми



(підпис)

Ратушняк Тетяна Володимирівна
канд. фіз.-мат. наук, доцент

(прізвище, ім'я, по батькові, науковий ступінь,
вчене звання)

Допущено до захисту за рішенням кафедри комп'ютерних та інформаційних
технологій і систем протокол від « 12 » травня 2026 № 20

Завідувач кафедри



(підпис)

Філоненко Михайло
Миколайович, канд. фіз.-мат.
наук, доцент

(прізвище, ім'я, по батькові, науковий ступінь,
вчене звання)

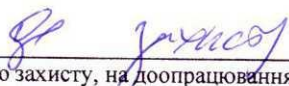
Ірпінь – 2026

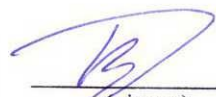
Реєстрація № 66

29.05.2026
(дата)


(підпис відповідальної особи)

Результати перевірки


(до захисту, на доопрацювання)

12.05.2026 
(дата) (підпис)

Результат захисту

4 (добре)
(оцінка)

11.06.2026 
(дата) (підпис)

Голова комісії:

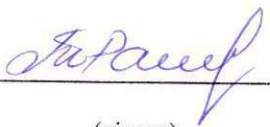

(підпис)

О.С. Дюбанов

Члени комісії:


(підпис)

М.М. Філоненко

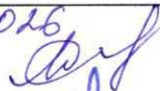
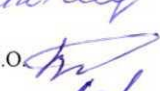

(підпис)

Т.В. Ратушняк


(підпис)

В.О. Ніжегородцев

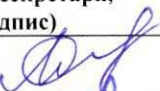
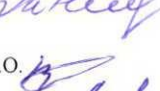
БЛАНК ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Об'єкти контролю	Діапазон оцінки в балах	Фактична кількість балів	Примітки	Дата, П.І.Б.: керівника, голови комісії членів комісії, секретаря, (підписи)
ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ КОМІСІЄЮ ПО ЗАХИСТУ				
1. Зміст випускної роботи (якість теоретичного матеріалу, його зв'язок з практичним, вміння представляти матеріал в логічній послідовності, наявність висновків та пропозицій)	від 0 до 30	28		11.06.2026 Дюбанов О.С.  Філоненко М.М. 
2. Дотримання вимог щодо оформлення	від 0 до 10	9		Ратушняк Т.В.  Ніжегородцев В.О. 
3. Вчасне виконання роботи на всіх етапах у встановлені кафедрою терміни	від 0 до 5	4		Суховерховська Л.П. 
4. Рецензент	від 0 до 5	4		
РАЗОМ БАЛІВ	від 0 до 50	45		

Оцінювання кваліфікаційної роботи науковим керівником – 40 балів.

ПЕРЕВЕДЕННЯ ОЦІНКИ ЗА 100-БАЛЬНОЮ ШКАЛОЮ ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА

ШКАЛИ ЄКТС

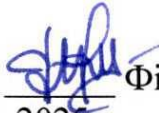
Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою	Дата	П.І.Б.: голови, членів комісії, секретаря, (підпис)
85	B	4 (добре)	11.06.2026	Дюбанов О.С.  Філоненко М.М.  Ратушняк Т.В.  Ніжегородцев В.О.  Суховерховська Л.П. 

ПРАВИЛА ПЕРЕВЕДЕННЯ ОЦІНОК ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою
90-100	A	5 (відмінно)
80-89	B	4 (добре)
70-79	C	
60-69	D	
50-59	E	3 (задовільно)
35-49	FX	2 (незадовільно)

Факультет фінансів та цифрових технологій
 Кафедра комп'ютерних та інформаційних технологій і систем
 Галузь знань: 12 «Інформаційні технології»
 Спеціальність: 122 «Комп'ютерні науки»
 Освітня програма: «Технології цифрового дизайну»

Затверджую

Завідувач кафедри  Філоненко М.М.

«01» грудня 2025

Завдання на кваліфікаційну роботу
 здобувачу вищої освіти
Іващенко Дар'ї Сергіївні
 (прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи Реалізація інтелектуального асистента для вебменю закладів громадського харчування

Затверджена наказом ректора від «28» листопада 2025 № 1526

2. Строк здачі закінченої роботи «11» травня 2026

3. Мета роботи: проектування та реалізація веб-сайту меню закладу громадського харчування з інтелектуальним асистентом на основі Google Gemini API, що здатний приймати запити природною мовою та повертати релевантні рекомендації страв..
 Об'єкт дослідження: процес цифровізації обслуговування відвідувачів у закладах громадського харчування засобами веб-технологій.

Предмет дослідження: методи та інструменти реалізації інтелектуального асистента для веб-меню на основі великих мовних моделей.

Орієнтовна база дослідження: Інтернет-ресурси та аналітика вебплатформ.

4. Вихідні дані до роботи: Аналіз предметної області, вимоги до системи, приклади конкурентних рішень, технічні завдання, інструменти проектування інтерфейсів, фреймворки для веб-розробки, дані для тестування функціональності сайту.

5. Перелік графічного матеріалу: 4 таблиці, 7 рисунків.

6. Зміст кваліфікаційної роботи (перелік питань за кожним розділом)

ВСТУП. РОЗДІЛ 1 ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ ІНТЕЛЕКТУАЛЬНИХ АСИСТЕНТІВ ДЛЯ ЗАКЛАДІВ ГРОМАДСЬКОГО ХАРЧУВАННЯ. 1.1 Цифровізація сервісу у закладах громадського харчування: стан та тенденції. 1.2 Великі мовні моделі: архітектура та принципи застосування. 1.3 Технологічний стек для розробки клієнтського веб-застосунку. РОЗДІЛ 2 ПРОЕКТУВАННЯ АРХІТЕКТУРИ ТА ІНТЕРФЕЙСУ СИСТЕМИ. 2.1 Аналіз вимог до функціонування системи та формування технічного завдання. 2.2 Розробка архітектури програмного забезпечення та моделі взаємодії користувача із системою. РОЗДІЛ 3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ СИСТЕМИ. 3.1 Реалізація програмних компонентів системи та інтеграція моделей штучного інтелекту. 3.2 Інтеграція Gemini API та реалізація ШІ-асистента. ВИСНОВОК. СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ. ДОДАТКИ

7 Календарний план виконання роботи

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	
		За планом	Фактично
1	2	3	4
1	Вибір теми, підбір літературних джерел, формування структури кваліфікаційної роботи	До 30.10.2025	До 30.10.2025
2	I розділ (розкрити сутність і стан дослідження наукової проблематики роботи, після чого навести її загальну характеристику; огляд літератури за темою і вибір конкретних напрямів досліджень)	29.12.2025	29.12.2025
3	II розділ (аналіз експериментальної частини про методики досліджень)	23.02.2026	23.02.2026
4	III розділ (проведені теоретичні і (або) експериментальні дослідження та проектні рішення)	20.04.2026	20.04.2026
5	Висновки	04.05.2026	04.05.2026
6	Доопрацювання роботи з урахуванням зауважень наукового керівника	11.05.2026	11.05.2026
7	Отримання відгуку керівника	01.05.2026	01.05.2026
8	Отримання зовнішньої рецензії	04.06.2026	04.06.2026

8. Дата видачі завдання «01» грудня 2025

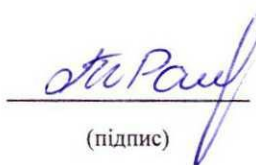
9. Керівник кваліфікаційної роботи


(підпис)

Ніжегородцев В.О.

(прізвище, ініціали)

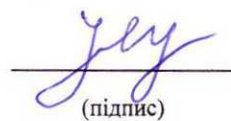
10. Керівник освітньо-професійної програми


(підпис)

Ратушняк Т.В.

(прізвище, ініціали)

11. Завдання до виконання отримав
здобувач вищої освіти


(підпис)

Іващенко Д.С.

(прізвище, ініціали)

АНОТАЦІЯ

Реалізація інтелектуального асистента для вебменю

закладів громадського харчування

Мета роботи — проєктування та реалізація веб-сайту меню закладу громадського харчування з інтелектуальним асистентом на основі Google Gemini API, що здатний приймати запити природною мовою та повертати релевантні рекомендації страв.

Завданнями роботи є дослідження теоретичних основ цифровізації обслуговування відвідувачів у закладах громадського харчування, аналіз принципів UX/UI дизайну та сучасних вебтехнологій, проведення аналізу аналогічних вебсайтів, формування вимог до системи, розробка концепції дизайну та реалізація інтелектуального асистента на основі великих мовних моделей.

Об'єкт дослідження — процес цифровізації обслуговування відвідувачів у закладах громадського харчування засобами веб-технологій. Предмет — методи та інструменти реалізації інтелектуального асистента для веб-меню на основі великих мовних моделей.

За результатами створено веб-сайт меню закладу громадського харчування з інтегрованим інтелектуальним асистентом на основі Google Gemini API. Система забезпечує обробку запитів природною мовою та надає персоналізовані рекомендації страв відвідувачам. Розроблено зручний інтерфейс із дотриманням принципів UX/UI дизайну та мобільної адаптивності.

Одержані результати можуть бути використані для впровадження у закладах громадського харчування або як приклад реалізації інтелектуального вебасистента на основі великих мовних моделей.

Кваліфікаційна робота містить 5 рисунків, 20 таблиць, список використаних джерел.

Ключові слова: інтелектуальний асистент, вебменю, заклади громадського харчування, Google Gemini API, великі мовні моделі, UX/UI дизайн, цифровізація.

Рік виконання та захисту: 2026.

ABSTRACT

Implementation of an Intelligent Assistant for the Web Menu of Food Service Establishments

The aim of the work is to design and implement a food service establishment web menu with an intelligent assistant based on the Google Gemini API, capable of processing natural language queries and returning relevant dish recommendations.

The objectives include researching the theoretical foundations of digitalization of visitor service in food service establishments, analyzing UX/UI design principles and modern web technologies, conducting an analysis of similar websites, defining system requirements, developing a design concept, and implementing an intelligent assistant based on large language models.

The object of research is the process of digitalization of visitor service in food service establishments using web technologies. The subject is the methods and tools for implementing an intelligent assistant for a web menu based on large language models.

As a result, a food service establishment web menu was developed with an integrated intelligent assistant based on the Google Gemini API. The system processes natural language queries and provides personalized dish recommendations to visitors. A user-friendly interface was designed in accordance with UX/UI design principles and mobile responsiveness.

The obtained results can be used for deployment in food service establishments or as a reference implementation of an intelligent web assistant based on large language models.

The qualification work contains 5 figures, 20 tables, and a list of references.

Keywords: intelligent assistant, web menu, food service establishments, Google Gemini API, large language models, UX/UI design, digitalization.

Year of completion and defence: 2026.

ЗМІСТ

ВСТУП.....	3
РОЗДІЛ 1 ТЕОРЕТИКО-МЕТОДОЛОГІЧНІ ОСНОВИ РОЗРОБКИ ІНТЕЛЕКТУАЛЬНИХ ВЕБ-СИСТЕМ ДЛЯ ЗАКЛАДІВ ГРОМАДСЬКОГО ХАРЧУВАННЯ.....	5
1.1. Цифровізація сервісу у закладах громадського харчування: стан та тенденції.....	5
1.2. Великі мовні моделі: архітектура та принципи застосування.....	8
1.3. Технологічний стек для розробки клієнтського веб-застосунку.....	12
РОЗДІЛ 2 ПРОЕКТУВАННЯ АРХІТЕКТУРИ ТА ІНТЕРФЕЙСУ СИСТЕМИ.....	16
2.1. Аналіз вимог до функціонування системи та формування технічного завдання.....	16
2.2. Розробка архітектури програмного забезпечення та моделі взаємодії користувача із системою.....	21
РОЗДІЛ 3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ СИСТЕМИ.....	34
3.1. Реалізація програмних компонентів системи та інтеграція моделей штучного інтелекту.....	34
3.2. Інтеграція Gemini API та реалізація ІІІ-асистента.....	38
ВИСНОВКИ.....	44
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	49
ДОДАТКИ.....	53

ВСТУП

Розвиток цифрових технологій суттєво змінює традиційні підходи до організації обслуговування в закладах громадського харчування. Якщо раніше паперове меню було єдиним інструментом презентації страв відвідувачу, то сьогодні заклади активно впроваджують QR-коди, веб-застосунки та мобільні рішення, що дозволяють ознайомитися з асортиментом ще до відвідування. Разом із тим більшість наявних рішень залишаються статичними: вони демонструють список страв, але не здатні взаємодіяти з відвідувачем у природній формі.

Поява великих мовних моделей (Large Language Models, LLM) відкрила принципово новий клас застосунків, у яких система розуміє запит людини, сформульований довільною мовою, і повертає контекстуально релевантну відповідь. Застосування таких моделей у сфері громадського харчування є перспективним напрямом: замість того щоб прокручувати довгий список страв, відвідувач може написати «хочу щось легке і без глютену» або «що підійде дітям на обід» — і отримати персоналізовану рекомендацію з реального меню конкретного закладу.

Актуальність теми дослідження зумовлена поєднанням двох тенденцій: стрімкою цифровізацією ресторанного бізнесу та доступністю API великих мовних моделей для розробників. Водночас готових рішень для малого та середнього бізнесу, що не потребують складної інфраструктури і можуть розгортатися як статичний веб-сайт, на ринку практично немає. Розроблена в межах цієї роботи система заповнює цю нішу.

Метою кваліфікаційної роботи є проєктування та реалізація веб-сайту меню закладу громадського харчування з інтелектуальним асистентом на основі Google Gemini API, що здатний приймати запити природною мовою та повертати релевантні рекомендації страв.

Для досягнення поставленої мети визначено такі завдання:

- проаналізувати предметну область та наявні аналоги веб-меню з елементами штучного інтелекту;
- дослідити архітектуру великих мовних моделей та принципи роботи Gemini API;
- спроектувати структуру та інтерфейс веб-сайту відповідно до вимог зручності використання;
- реалізувати клієнтську частину сайту засобами HTML5, CSS3 та JavaScript;
- інтегрувати Google Gemini API як інтелектуальний бекенд для обробки запитів природною мовою;
- забезпечити адаптивність інтерфейсу для мобільних і десктопних пристроїв;
- провести функціональне тестування розробленої системи.

Об'єктом дослідження є процес цифровізації обслуговування відвідувачів у закладах громадського харчування засобами веб-технологій.

Предметом дослідження є методи та інструменти реалізації інтелектуального асистента для веб-меню на основі великих мовних моделей.

Методи дослідження: аналіз наукових джерел і технічної документації; порівняльний аналіз існуючих рішень; методи об'єктно-орієнтованого та компонентного проєктування; методи тестування програмного забезпечення.

Практична значущість роботи полягає у створенні повнофункціонального веб-застосунку, який може бути безпосередньо використаний будь-яким закладом громадського харчування після мінімального налаштування. Код проєкту є відкритим і документованим, що дозволяє його подальше вдосконалення.

Структура роботи. Кваліфікаційна робота складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків. Загальний обсяг основного тексту становить 57 сторінок. Робота містить 20 таблиць, 5 рисунків, список із 47 джерел.

РОЗДІЛ 1 ТЕОРЕТИКО-МЕТОДОЛОГІЧНІ ОСНОВИ РОЗРОБКИ ІНТЕЛЕКТУАЛЬНИХ ВЕБ-СИСТЕМ ДЛЯ ЗАКЛАДІВ ГРОМАДСЬКОГО ХАРЧУВАННЯ

1.1 Цифровізація сервісу у закладах громадського харчування: стан та тенденції

Заклади громадського харчування належать до тих секторів економіки, де цифрова трансформація відбувається особливо стрімко та масштабно. Ресторани, кав'ярні, фастфуд-мережі та бари сьогодні змушені адаптуватися до нових очікувань споживачів, які дедалі більше орієнтуються на зручність, швидкість і персоналізацію обслуговування. За даними аналітичних досліджень, понад 70% відвідувачів переглядають меню закладу онлайн ще до фактичного відвідування, а 45% свідомо надають перевагу тим закладам, що пропонують мобільне або QR-меню. Ці показники суттєво зросли після пандемії COVID-19, яка стала потужним каталізатором масового запровадження безконтактних технологій у сфері обслуговування. Локдауни та санітарні обмеження фактично змусили галузь у прискореному режимі впроваджувати цифрові рішення, що раніше вважалися лише опціональним доповненням [6].

Сучасні цифрові меню можна класифікувати за кількома ключовими ознаками. За формою подання інформації виокремлюють такі різновиди: PDF-меню як найпростіший і найдешевший формат, позбавлений будь-якої інтерактивності; статичні веб-сторінки з фіксованим вмістом, що потребують ручного оновлення; динамічні веб-застосунки з підключеною базою даних, які дозволяють оновлювати інформацію в реальному часі; та повноцінні мобільні додатки з розширеним функціоналом. За рівнем інтелектуалізації та персоналізації розрізняють системи без будь-якої адаптації до користувача, системи з базовою фільтрацією за

параметрами (алергенами, калорійністю, ціною, дієтичними обмеженнями тощо) та системи з інтегрованими рекомендаційними алгоритмами, що враховують індивідуальні вподобання відвідувача, час доби, сезонність і попередню історію замовлень [7].

Ключовою тенденцією останніх років у галузі стало зміщення акценту з пасивного інформування на активну, двосторонню комунікацію з відвідувачем. Цифрове меню трансформується з простого електронного дубліката паперового формату у повноцінний інтерактивний комунікаційний інструмент між закладом та відвідувачем. Провідні платформи автоматизації ресторанного бізнесу, зокрема Poster, iiko, Rkeeper, а також міжнародні рішення, як-от Toast, Lightspeed та Square for Restaurants, вже інтегрують модулі персоналізованих рекомендацій та аналітики поведінки споживачів. Однак ці комплексні рішення супроводжуються значною вартістю впровадження та підтримки і призначені переважно для великих мереж з розвинутою інфраструктурою та ІТ-персоналом. Для сегменту малого та середнього бізнесу — невеликих ресторанів, локальних кав'ярень — на ринку відсутні доступні та технологічно розвинені аналоги з використанням справжнього штучного інтелекту, що створює значну незакриту нішу.

Окремо слід виділити глобальну тенденцію щодо інтенсивного застосування великих мовних моделей у клієнтських сервісах різних секторів. Чат-боти, побудовані на таких моделях, як GPT-4o, Gemini 1.5 Pro та Claude 3, вже широко використовуються у сфері підтримки клієнтів, електронної комерції, туризму, охорони здоров'я та освіти. Вони демонструють високу здатність розуміти природну мову, підтримувати контекстуальний діалог, надавати індивідуалізовані відповіді та обробляти складні, багатоетапні запити користувачів. Проте у ресторанній галузі ця технологія перебуває на ранніх етапах впровадження, що відкриває значний потенціал для інноваційних розробок та підвищення конкурентоспроможності на ринку.

Порівняльний аналіз наявних рішень, відображений у таблиці 1.1, свідчить про те, що розроблена у межах цієї роботи система займає унікальну технологічну нішу: вона поєднує повноцінну підтримку штучного інтелекту з низькою вартістю розгортання та обслуговування. Така комбінація характеристик стає можливою завдяки використанню безкоштовного рівня Google Gemini API та архітектури без власної серверної частини (serverless-підхід), що усуває необхідність у підтримці дорогої хмарної інфраструктури і робить рішення доступним навіть для найменших закладів [4].

Таблиця 1.1

Тип рішення	Інтерактивність	ШІ	Вартість
PDF-меню	Відсутня	Ні	Безкоштовно
Статичний сайт	Мінімальна	Ні	Низька
QR-меню сервіс	Середня	Ні	Середня
POS-система	Висока	Частково	Висока
Розроблена система	Висока	Так	Низька

Порівняльний аналіз типів цифрових меню

Таким чином, аналіз поточного стану та тенденцій цифровізації у сфері громадського харчування демонструє, що галузь перебуває в стані інтенсивної технологічної трансформації. Пандемія COVID-19 суттєво прискорила цей процес, сформувавши у споживачів стійкі очікування щодо отримання цифрових послуг та забезпечення безконтактної взаємодії. На ринку представлено значний спектр рішень — від базових PDF-меню до високотехнологічних платформ з інтелектуальними рекомендаційними системами. Проте існує помітна асиметрія: передові технологічні рішення часто є недоступними для малого та середнього підприємництва через високу вартість, тоді як більш економічно обґрунтовані

альтернативи не гарантують необхідного рівня персоналізації чи інтерактивності. Це протиріччя підкреслює актуальність та практичну значущість розробки, яка є предметом цього дослідження. Інтеграція великих мовних моделей у функціонал цифрових меню закладів може ефективно подолати цю прогалину, надавши малому бізнесу інструментарій, раніше доступний переважно великим мережевим гравцям. Саме цей напрямок визначає ключовий вектор подальших наукових пошуків.

1.2 Великі мовні моделі: архітектура та принципи застосування

Великі мовні моделі становлять собою клас алгоритмів машинного навчання, які навчаються на гігантських обсягах текстових даних з метою передбачення наступного токена у заданій послідовності. Формально ця задача полягає у моделюванні умовного розподілу ймовірностей, де система засвоює вміння визначати кожен наступний елемент, беручи до уваги всю попередню інформацію. Завдяки масштабу навчального корпусу, який може сягати сотень мільярдів токенів, та значній кількості параметрів самої моделі, вона набуває так званих емерджентних властивостей. До них належить здатність до логічного висновку, узагальнення знань, перекладу текстів, генерації програмного коду та ведення послідовної діалогу, навіть якщо зазначені компетенції не були прямо запрограмовані.

Архітектурну основу переважної більшості сучасних великих мовних моделей складає трансформер, запропонований у роботі Vaswani та співавторів у 2017 році. До появи трансформерів домінуючими архітектурами у задачах обробки природної мови були рекурентні нейронні мережі та їхня розширена версія — мережі з довгою короткостроковою пам'яттю. Однак ці підходи мали суттєві обмеження: вони обробляли послідовність токен за токеном, що унеможливлювало ефективне розпаралелювання обчислень і ускладнювало роботу з довгими

залежностями в тексті. Трансформер вирішив обидві проблеми завдяки механізму самоуваги [38].

Механізм уваги - це ключове нововведення в архітектурі трансформерів. Його суть полягає в тому, що під час обробки кожного елемента (токена) модель динамічно визначає важливість усіх інших елементів у послідовності та створює зважену суму їхніх подань. Технічно це досягається шляхом векторних добутків між векторами запитів, ключів та значень. Таким чином, модель може встановлювати зв'язки між словами, незалежно від їхнього розташування в тексті - наприклад, пов'язати займенник із займенником, що знаходиться кілька речень том. Механізм багатошарової уваги дає змогу моделі одночасно відслідковувати кілька різних типів залежностей, що суттєво збільшує виразність подань.

Сучасні LLM проходять кілька етапів навчання. На першому етапі — передтренування — модель навчається на великому корпусі тексту у режимі самонавчання, без розмітки. На другому етапі — тонкому налаштуванні з людським зворотним зв'язком (RLHF, Reinforcement Learning from Human Feedback) — модель коригується таким чином, щоб її відповіді відповідали людським очікуванням: були корисними, безпечними та правдивими. Саме цей другий етап перетворює базову мовну модель на зручного асистента, здатного вести змістовний діалог [15].

Моделі сімейства Gemini від Google розроблено з нативною мультимодальністю, що принципово відрізняє їх від багатьох конкурентів. На відміну від підходів, де текстові та візуальні можливості інтегруються поетапно як окремі модулі, Gemini від самого початку навчено на спільному корпусі, що включає текст, зображення, аудіо та відео. Це забезпечує більш природну та глибоку інтеграцію між модальностями, а не просто їхнє механічне об'єднання на рівні інтерфейсу. Сімейство Gemini представлене декількома версіями, що відрізняються балансом між продуктивністю, розміром контекстного вікна та швидкістю генерації. Для завдань, що розглядаються в цій дипломній роботі,

використовується текстовий режим моделі Gemini 1.5 Flash — версії, оптимізованої для максимальної швидкості відповіді та ефективної роботи в умовах обмежених ресурсів [22].

Ключовою концепцією при розробці прикладних застосунків на основі великих мовних моделей є системний пром프트 — набір інструкцій, що передаються моделі перед початком діалогу з користувачем і визначають її роль, поведінку, обмеження та контекст роботи. На відміну від звичайних повідомлень користувача, системний пром프트 має вищий пріоритет і залишається активним протягом усієї сесії. Правильно спроектований системний пром프트 є ключовим чинником якості відповідей: він дозволяє суттєво підвищити їхню релевантність, звузити область генерації до предметної галузі та запобігти небажаним або нерелевантним виходам [29].

Галузь проектування пром프트ів — *prompt engineering* — виокремила як самостійна інженерна дисципліна зі своїми усталеними практиками та техніками. До основних із них відносять такі підходи:

- Визначення ролі - модель отримує чітку вказівку щодо своєї функції, наприклад: «Ти є консультантом ресторану і допомагаєш гостям обрати страву»;
- Ін'єкція контексту - до промπτу включається структурована предметна інформація, якою модель має керуватися при генерації відповідей;
- Встановлення обмежень - явна вказівка на те, чого модель не повинна робити: виходити за межі теми, вигадувати інформацію, давати медичні або юридичні поради;
- Ланцюжок міркувань - техніка, за якої модель заохочується спочатку міркувати покроково, а потім формувати відповідь, що підвищує точність на складних логічних задачах;

- Few-shot prompting - надання моделі кількох прикладів очікуваної поведінки безпосередньо у промпті, що особливо ефективно для задач із нестандартним форматом виходу [45].

Для задачі рекомендації страв у ресторані особливого значення набуває техніка ін'єкції контексту. У системний промпт включається повне актуальне меню закладу у структурованому вигляді — із назвами страв, описами, складниками, ціновою категорією та доступними фільтрами. Це дозволяє моделі будувати рекомендації виключно на підставі реально наявних позицій і не генерувати вигаданих страв чи некоректних цін. Такий підхід концептуально відноситься до парадигми Retrieval-Augmented Generation у її спрощеній формі: роль зовнішньої бази знань відіграє меню закладу, а роль механізму вибірки — безпосередньо структура промпту. На відміну від класичних RAG-систем, що потребують векторних баз даних і складних пайплайнів вибірки, даний підхід реалізується значно простіше і водночас цілком достатньо для обсягу даних, характерного для меню кав'ярні або ресторану [28].

Порівняльний аналіз доступних моделей, представлений у таблиці 1.2, свідчить про те, що Gemini 1.5 Flash є оптимальним вибором для даного проєкту за сукупністю характеристик. Модель забезпечує найбільшу кількість безкоштовних запитів на місяць, що критично важливо для прототипування та малого бізнесу з обмеженим бюджетом. Вона демонструє високу швидкість генерації відповідей, що безпосередньо впливає на сприйняття сервісу відвідувачем. Нарешті, нативна підтримка великого контекстного вікна — до одного мільйона токенів — дозволяє передавати у промпті повне меню будь-якого реального закладу без необхідності його скорочення чи сегментації, що спрощує архітектуру системи та підвищує якість рекомендацій [10].

1.3 Технологічний стек для розробки клієнтського веб-застосунку

Вибір технологічного стеку для розробки веб-застосунку є одним з найважливіших архітектурних рішень на початковому етапі проекту. Вибір неправильного стеку може ускладнити розробку, а також призвести до високих витрат на підтримку та проблем із масштабованістю в майбутньому. І навпаки, вибір надмірно складного технологічного стеку для простого проекту призведе до непотрібних витрат на розробку та збільшить труднощі для нових розробників. Перелік функціональних вимог описує, що може робити система; нефункціональні вимоги стосуються швидкості, масштабованості, надійності та часу безвідмовної роботи; організаційні вимоги включають вартість створення системи, бар'єри для входу від вивчення того, як використовувати стек, наскільки розвинена технологія та обсяг або рівень підтримки розробників.

Проведений аналіз існуючих методологій фронтенд-розробки виявив три ключові альтернативи: застосування сучасних JavaScript-фреймворків (React, Vue, Angular), використання легковесних бібліотек (Svelte, Preact, Alpine.js) або реалізація рішення, що базується на нативних веб-технологіях без зовнішніх залежностей. Кожна з цих опцій має специфічний набір переваг та обмежень, і вибір оптимального підходу має ґрунтуватися виключно на функціональних вимогах проекту, а не на модних тенденціях чи суб'єктивних уподобаннях розробників.

Для даного проекту обрано підхід Vanilla SPA — єдиний HTML-файл із вбудованим CSS і JavaScript без будь-яких зовнішніх залежностей і без застосування фреймворків чи збирачів. Такий вибір обумовлений рядом вагомих практичних міркувань [3].

По-перше, розгортання такого застосунку є гранично простим і не потребує жодної серверної інфраструктури. Файл може відкриватися безпосередньо в браузері як локальний ресурс або розміщуватися на будь-якому статичному хостингу — GitHub Pages, Netlify, Vercel чи навіть у звичайній папці хмарного

сховища. Це значно знижує поріг впровадження для кінцевого замовника, який не має технічних фахівців у штаті [9].

По-друге, повна незалежність від зовнішніх залежностей усуває низку проблем, характерних для сучасних JavaScript-проектів. Зокрема, це мінімізує ризики невідповідності версій бібліотек, конфлікти між транзитивними залежностями, а також потенційні загрози, пов'язані з ланцюжком постачання — вектор ризиків, який набув критичного значення після низки резонансних інцидентів в екосистемі npm. Відсутність файлу `package.json` та каталогу `node_modules` означає відсутність сотень мегабайт допоміжного коду, безпека якого залишається поза контролем розробника.

По-третє, компактність та прозорість кодової бази істотно оптимізують процеси освоєння, технічного обслуговування та подальшої еволюції програмного забезпечення. Це дозволяє будь-якому кваліфікованому веб-розробнику визначити архітектуру проєкту та вносити необхідні зміни без необхідності детального вивчення документації певного фреймворку. Звертається особлива увага до цього аспекту для малого бізнесу, адже у цьому випадку підтримка системи може здійснюватися різними зовнішніми виконавцями.

Ключовою мовою розмітки є HTML5, що надає широкий набір семантичних елементів: `<nav>`, `<main>`, `<section>`, `<article>`, `<aside>`, `<footer>`, `<header>`. Використання семантичної розмітки замість універсальних `<div>` і `<span>` дає декілька відчутних переваг. По-перше, підвищується доступність застосунку для користувачів із вадами зору, оскільки програми екранного зчитування коректно інтерпретують структуру сторінки. По-друге, покращується індексація контенту пошуковими системами, що є важливим для видимості закладу в мережі. По-третє, семантична структура сама по собі є документацією: розробник, що читає розмітку, одразу розуміє призначення кожного блоку без необхідності аналізу CSS-класів чи JavaScript-коду [44].

Для стилізації застосунку використовується CSS3 із активним застосуванням сучасних можливостей мови. CSS Custom Properties — змінні, оголошені на рівні кореневого елемента — дозволяють централізовано керувати всією кольоровою схемою, типографікою та системою відступів. Зміна однієї змінної автоматично поширюється на весь застосунок, що робить адаптацію під брендинг конкретного закладу тривіальним завданням. Механізм CSS Grid у поєднанні з функцією `auto-fill` і `minmax()` забезпечує повністю адаптивну сітку карток меню: кількість колонок автоматично підлаштовується під ширину екрана без необхідності писати медіазапити для кожного брейкпоінту окремо. Це суттєво скорочує обсяг CSS-коду і знижує ризик неузгодженостей при зміні дизайну [40]. Додатково CSS Flexbox використовується для вирівнювання елементів у межах окремих компонентів, а CSS-анімації на основі `transition` і `keyframes` забезпечують плавні переходи без використання JavaScript.

JavaScript у версії стандарту ES2022 є цілком достатнім для реалізації всієї логіки застосунку без залучення будь-яких бібліотек. Для взаємодії з Gemini API використовується вбудований Fetch API — сучасний асинхронний інтерфейс для HTTP-запитів, що підтримує проміси та синтаксис `async/await`, роблячи код читабельним і зручним для обробки помилок. Для реалізації анімацій при прокрутці сторінки застосовується IntersectionObserver API — ефективний браузерний механізм відстеження видимості елементів, що є значно продуктивнішою альтернативою обробникам події `scroll`. Логіка застосунку структурована через патерн замикань і модульного IIFE, що забезпечує інкапсуляцію стану та запобігає забрудненню глобального простору імен. Жодного транспілятора на кшталт Babel, жодного збирача на кшталт Webpack чи Vite не потрібно — код виконується нативно у будь-якому сучасному браузері без додаткових кроків складання [20].

Окремо варто наголосити на питанні довгострокової підтримки обраного підходу. Фреймворки та бібліотеки мають власний життєвий цикл: вони

з'являються, набувають популярності, а потім застарівають і витісняються новими рішеннями. Застосунок, написаний на нативних веб-технологіях, позбавлений цієї проблеми: HTML, CSS і JavaScript є стандартами, що розвиваються з повним збереженням зворотної сумісності.

Код, написаний сьогодні без залежностей, з високою ймовірністю буде коректно виконуватись у браузерах через п'ять і десять років без жодних змін [2].

РОЗДІЛ 2 ПРОЄКТУВАННЯ АРХІТЕКТУРИ ТА ІНТЕРФЕЙСУ СИСТЕМИ

2.1 Аналіз вимог до функціонування системи та формування технічного завдання

На основі проведеного аналізу предметної області, існуючих цифрових рішень для закладів громадського харчування та сучасних тенденцій використання штучного інтелекту у сфері обслуговування клієнтів було сформовано комплекс вимог до розроблюваної системи. Вимоги поділяються на функціональні та нефункціональні, що дозволяє комплексно охарактеризувати поведінку системи та її якісні характеристики.

Функціональні вимоги до системи			
●Обов'язковий пріоритет●Бажаний пріоритет			
№	ФУНКЦІОНАЛЬНА ВИМОГА	ПРІОРИТЕТ	СЦЕНАРІЙ ВИКОРИСТАННЯ
Ф-1	Відображення головного екрана з плитками-категоріями	●Обов'язковий	UC-1: Навігація
Ф-2	Перехід до списку страв окремої категорії по кліку на плитку	●Обов'язковий	UC-2: Перегляд категорії
Ф-3	Картка страви: фото, назва, тег, опис (скорочений)	●Обов'язковий	UC-2
Ф-4	Модальне вікно з повним описом страви по кліку	●Обов'язковий	UC-3: Перегляд деталей
Ф-5	Перемикання між категоріями через вкладки без переходу	●Бажаний	UC-2
Ф-6	ШІ-асистент: поле вводу, чіпси-підказки, відповідь	●Обов'язковий	UC-4: ШІ-пошук
Ф-7	Картки рекомендованих страв під відповіддю ШІ	●Бажаний	UC-4
Ф-8	Нижня навігаційна панель: Головна / Меню / ШІ / Про нас	●Обов'язковий	UC-1
Ф-9	Секція «Про нас» із текстом, списком та контактами	●Бажаний	UC-5: Контакти
Ф-10	Адаптивний інтерфейс для мобільних пристроїв	●Обов'язковий	Усі UC

Рис. 2.1. Функціональні вимоги до систем

Функціональні вимоги визначають набір можливостей (рис. 2.1.), які повинна забезпечувати система для кінцевих користувачів. Основою для їх формування став

аналіз типових сценаріїв взаємодії відвідувачів із цифровим меню закладу громадського харчування. У системі передбачено два основних типи користувачів: відвідувач закладу та адміністратор меню. Відвідувач використовує вебзастосунок через браузер мобільного пристрою або персонального комп'ютера для перегляду меню та отримання рекомендацій. Адміністратор відповідає за оновлення структури меню та його вмісту шляхом редагування даних у вихідному файлі системи.

Основною функцією застосунку є надання користувачеві швидкого та зручного доступу до інформації про страви та напої. Для цього система повинна забезпечувати перегляд категорій меню, відображення переліку позицій у межах кожної категорії, перегляд детальної інформації про конкретну страву, включаючи її назву, опис, складові інгредієнти, ціну та фотографію. Також користувач повинен мати можливість швидко переміщуватися між розділами застосунку за допомогою навігаційної панелі.

Важливою функціональною особливістю системи є інтеграція інтелектуального асистента, який виступає цифровим консультантом з вибору страв. Асистент аналізує запити користувача природною мовою та надає персоналізовані рекомендації на основі доступного меню. Користувач може формулювати запити у довільній формі, наприклад: «Порадь щось солодке до кави», «Які страви містять курку?», «Що можна замовити до 200 гривень?» або «Покажи вегетаріанські позиції». На основі отриманого запиту система виконує пошук релевантних позицій та формує зрозумілу текстову відповідь із рекомендаціями.

Для покращення взаємодії передбачено використання готових підказок у вигляді інтерактивних кнопок або чіпів, натискання на які автоматично формує запит до асистента. Такий підхід дозволяє спростити використання системи для користувачів, які не бажають самотійно вводити текстові запити [17].

Окремою функціональною вимогою є автоматичне формування контексту для роботи штучного інтелекту. Система повинна самостійно збирати інформацію про доступні страви та передавати її моделі штучного інтелекту під час обробки запитів. Це виключає необхідність ручного дублювання даних та забезпечує актуальність рекомендацій після оновлення меню.

Нефункціональні вимоги визначають якісні характеристики програмного продукту та його експлуатаційні параметри [26]. Ключовою вимогою є забезпечення високої продуктивності системи. Завантаження основного інтерфейсу має відбуватися максимально оперативно, зокрема й на пристроях із помірними технічними характеристиками. Час появи першого контенту не повинен перевищувати кількох секунд, а взаємодія з ШІ-асистентом повинна здійснюватися без помітних затримок для користувача.

Окремо слід підкреслити необхідність сумісності з актуальними браузерами. Система повинна функціонувати коректно у найпоширеніших браузерах як на настільних, так і на мобільних пристроях, гарантуючи єдиний користувацький досвід незалежно від операційної системи чи типу пристрою. Інтерфейс має забезпечувати автоматичну адаптацію до різноманітних розмірів екранів, включаючи смартфони, планшети та персональні комп'ютери.

Важливим аспектом є доступність вебзастосунку. Для забезпечення комфортного використання необхідно дотримуватися сучасних стандартів вебдоступності. Контрастність тексту повинна забезпечувати легке читання контенту, а структура сторінок має бути зрозумілою та логічною [43]. Елементи інтерфейсу повинні мати достатній розмір для взаємодії на сенсорних екранах.

До вимог безпеки належить захист передаваних даних та використання захищених протоколів зв'язку. Всі мережеві запити до зовнішніх сервісів повинні виконуватися через HTTPS. Також необхідно мінімізувати ризики

несанкціонованого використання API-ключів шляхом налаштування відповідних обмежень доступу [14].

Ключове значення має забезпечення підтримованості програмного забезпечення. Код застосунку повинен мати чітку структуру, логічно організований вигляд та бути належним чином документований. Це створює умови для подальшого розвитку системи, ефективного виправлення недоліків та внесення змін із мінімальними часовими витратами. Архітектура повинна бути достатньо гнучкою, щоб дозволити додавання нових категорій меню, зміну інтерфейсного дизайну та інтеграцію додаткових функціональних можливостей без необхідності повного редизайну системи.

Масштабованість контенту є однією з фундаментальних вимог до розроблюваного рішення. Процес додавання нових позицій у меню має бути максимально спрощеним і не передбачати модифікацій основної робочої логіки застосунку. Завдяки автоматичному формуванню структури даних на основі існуючих елементів меню, система здатна адаптуватися до змін контенту без необхідності втручання у програмний код.

Для опису взаємодії користувачів із системою застосовується модель варіантів використання. Найважливішим сценарієм є отримання рекомендацій від ШІ-асистента. Цей процес ініціюється введенням користувачем текстового запиту або вибором із запропонованих підказок. Після цього система здійснює аналіз запиту, формує відповідний контекст для моделі штучного інтелекту та відправляє запит до зовнішнього сервісу. Отримані дані обробляються та відображаються у вигляді текстових рекомендацій та відповідних карток страв. У випадку виникнення технічних неполадок користувачеві надається повідомлення про неможливість виконання запиту із пропозицією повторити спробу пізніше.

Архітектурно система реалізована як односторінковий вебзастосунок (Single Page Application) [12]. Такий підхід дозволяє забезпечити швидку навігацію між

розділами без перезавантаження сторінки та створює більш комфортний користувацький досвід. Відсутність серверної частини значно спрощує процес розгортання та знижує витрати на підтримку інфраструктури.

Архітектура застосунку побудована на принципах розділення відповідальності між даними, інтерфейсом та логікою управління. Інформація про меню зберігається у централізованій JavaScript-структурі, яка слугує єдиним джерелом даних для всіх компонентів системи. Інтерфейс користувача формується залежно від поточного стану застосунку, тоді як логіка навігації та обробки подій реалізована у окремих функціональних модулях.

Ключовою особливістю реалізації є динамічне формування контенту: замість статичного дублювання HTML-коду для кожної пункту меню, застосунок генерує необхідні елементи інтерфейсу під час виконання. Це сприяє зменшенню обсягу кодової бази, підвищенню гнучкості системи та спрощенню подальшого технічного обслуговування.

Система підтримує перехід між кількома основними маршрутами: головною сторінкою, сторінкою меню, інтерфейсом інтелектуального асистента та інформаційним розділом про заклад. Кожен перехід між маршрутами супроводжується оновленням активного стану інтерфейсу, автоматичним прокручуванням до початку сторінки та відображенням відповідного вмісту. Для покращення користувацького досвіду також інтегровано механізм повернення до попереднього екрана.

Під час роботи інтелектуального асистента формується єдиний контекст меню, який містить інформацію про всі доступні страви та напої. Це дозволяє моделі штучного інтелекту аналізувати повний асортимент закладу та формувати більш точні рекомендації. Після отримання відповіді від зовнішнього сервісу система додатково виконує обробку результатів та автоматично знаходить відповідні позиції у меню для їх подальшого відображення користувачу.

Дані меню зберігаються виключно в оперативній пам'яті браузера протягом тривалості сеансу користувача. Відмова від використання зовнішніх баз даних істотно спрощує архітектуру та зменшує кількість системних залежностей. Така концепція є цілком адекватною для невеликих закладень громадського харчування, де оновлення меню відбувається нерегулярно.

Отже, розроблена система успішно комбінує архітектурну простоту, мінімальні вимоги до інфраструктури та передові можливості штучного інтелекту. Інтеграція односторінкового вебзастосунку з внутрішнім інтелектуальним асистентом створює високоефективний інструмент для цифрового відображення меню закладу та підвищення рівня взаємодії з відвідувачами.

2.2. Розробка архітектури програмного забезпечення та моделі взаємодії користувача із системою

Ключовим архітектурним рішенням (*рис.2.2.*) під час проєктування користувацького інтерфейсу стало використання концепції односторінкового застосунку з окремими логічними екранами (view) замість традиційного підходу, за якого весь контент розміщується на одній довгій сторінці з навігацією через якірні посилання. Такий підхід дозволяє створити більш структурований та зрозумілий користувацький досвід, наближений до мобільних застосунків. Користувач взаємодіє лише з тією інформацією, яка є актуальною на поточному етапі його сценарію використання, що значно знижує когнітивне навантаження та спрощує навігацію.

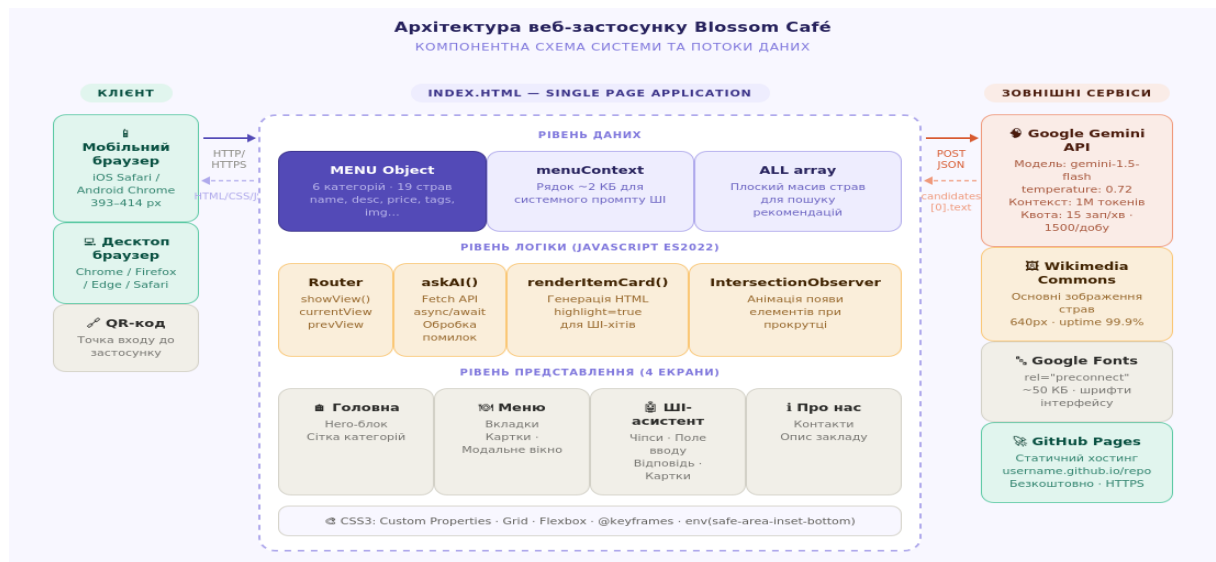


Рис.2.2. Архітектура веб-застосунку

Концепція розробки інтерфейсу адаптована до вимог сучасних цифрових меню закладів громадського харчування, де оперативність отримання даних є ключовим критерієм. Після сканування QR-коду відвідувач потрапляє на стартову сторінку, яка функціонує як центральний навігаційний вузол. Замість надмірного обсягу текстової інформації користувачеві негайно пропонуються категорії меню у вигляді візуально привабливих блоків. Така методика сприяє швидкому визначенню необхідного розділу та початку ознайомлення з асортиментом без необхідності прокручування великих масивів контенту. Головний екран програми реалізовано за принципами мінімалізму та забезпечення швидкої орієнтації користувача. Верхня частина інтерфейсу укомплектована інформаційним блоком закладу, що включає логотип, торговельну назву, стислий опис концепції, графік роботи та візуальні елементи корпоративного стилю. Основна функція цього блоку полягає у формуванні первинного враження від закладу та встановленні емоційного зв'язку з користувачем. Для досягнення цієї мети використано великі типографічні елементи, якісний фотоконтент та акцентні кольори, узгоджені з загальною естетикою бренду.

Нижня частина головного екрана представлена адаптивною сіткою категорій. Для її реалізації використано механізм CSS Grid, який автоматично перебудовує розташування елементів залежно від ширини екрана. Кожна плитка виконує одночасно інформаційну та навігаційну функцію. Усередині плитки відображається візуальний символ категорії, її назва, короткий опис та допоміжна інформація про кількість доступних позицій. Використання великих клікабельних зон забезпечує комфортну взаємодію на сенсорних пристроях та зменшує ймовірність помилкових натискань.

Особливу увагу приділено категоріям, які є найбільш важливими для бізнес-логіки закладу. Такі розділи можуть отримувати візуальний пріоритет за рахунок збільшеного розміру плитки, додаткових декоративних елементів або контрастнішого оформлення. Подібний підхід дозволяє спрямовувати увагу користувача на ключові категорії меню та сприяє підвищенню їхньої популярності серед відвідувачів.

Після вибору категорії користувач переходить до екрана перегляду меню. Цей екран складається з кількох логічних областей, кожна з яких виконує окрему функцію. Верхня частина містить навігаційний заголовок із кнопкою повернення та назвою поточного розділу. Завдяки цьому користувач завжди розуміє своє поточне місцезнаходження в структурі застосунку.

Під заголовком розташовано горизонтальну систему вкладок, яка забезпечує швидке перемикання між категоріями без повернення на головний екран. Реалізація горизонтального прокручування дозволяє розміщувати необмежену кількість категорій навіть на вузьких мобільних екранах. Водночас приховування системного скролбара забезпечує більш чистий та сучасний зовнішній вигляд інтерфейсу.

Основний контент екрана складається зі списку карток страв. При проєктуванні цього компонента було враховано специфіку використання меню на

мобільних пристроях. Замість класичного вертикального формату карток використано горизонтальну компоновку, яка дозволяє ефективніше використовувати доступний простір екрана. Користувач отримує змогу переглядати більшу кількість позицій без додаткового прокручування, що позитивно впливає на швидкість вибору страв [21].

Кожна картка складається з кількох інформаційних блоків. Ліворуч розташовується мініатюра страви, яка виконує функцію швидкої візуальної ідентифікації. У центральній частині відображаються основні текстові дані: назва страви, її категорія та короткий опис. Для уникнення перевантаження інтерфейсу обсяг тексту обмежується визначеною кількістю рядків, після чого застосовується автоматичне скорочення з багатокрапкою. Праворуч розташовуються числові характеристики, такі як ціна, вага або калорійність.

Для відображення детальної інформації про страву застосовується модальне вікно. Цей підхід забезпечує збереження поточного контексту взаємодії користувача із застосунком, запобігаючи необхідності залишати активний екран. Модальне вікно з'являється поверх основного контенту з використанням плавної анімації, а напівпрозоре затемнення фону акцентує увагу користувача на даних обраної позиції. У межах цього вікна розміщується повний пакет інформації, що включає високоякісне зображення, назву, вартість, детальний опис, перелік інгредієнтів, харчову цінність та інші параметри. Така структура дозволяє надати максимальний обсяг даних без перевантаження основного меню.

Ключовим елементом інтерфейсу користувача є нижня навігаційна панель, яка гарантує постійний доступ до основних розділів програми незалежно від поточного екрана. Завдяки фіксованому розташуванню у нижній частині дисплея навігація залишається доступною навіть під час скролінгу великих масивів контенту. Це дозволяє користувачеві оперативно перемикатися між головною

сторінкою, каталогом, функціоналом ІІІ-асистента та інформаційним блоком без повернення до попередніх екранів.

З метою забезпечення візуальної єдності застосунку розроблено власну дизайн-систему. Вона охоплює стандартизовані регламенти використання кольорової палітри, типографіки, відступів, розмірів компонентів та анімаційних ефектів. Наявність такої системи сприяє підтримці узгодженого зовнішнього вигляду всіх елементів інтерфейсу та значно спрощує подальший розвиток проекту.

Кольорова палітра побудована на теплих природних відтінках (рис.2.3.), що асоціюються з кавою, деревиною та натуральними матеріалами. Основний фоновий колір створює відчуття затишку та комфорту, тоді як акцентні кольори використовуються для виділення важливих елементів інтерфейсу. Такий підхід сприяє формуванню позитивного емоційного сприйняття бренду та покращує загальний користувацький досвід [29].

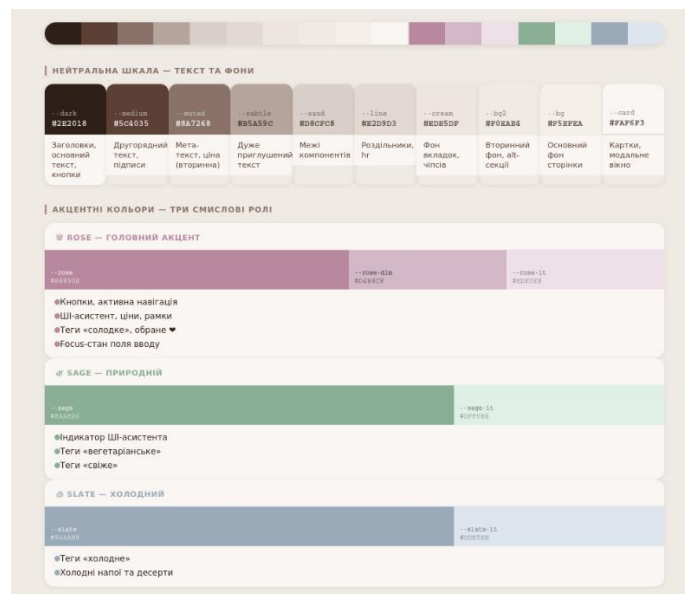


Рис.2.3. Кольорова палітра сайту

Типографічна система базується на поєднанні декоративного шрифту для заголовків та сучасного гротескного шрифту для основного тексту (Рис.2.4.). Контраст між цими гарнітурами дозволяє створити виразну візуальну ієрархію та

полегшує сприйняття інформації. Заголовки привертають увагу користувача, тоді як основний текст залишається максимально читабельним навіть на невеликих мобільних екранах [31].

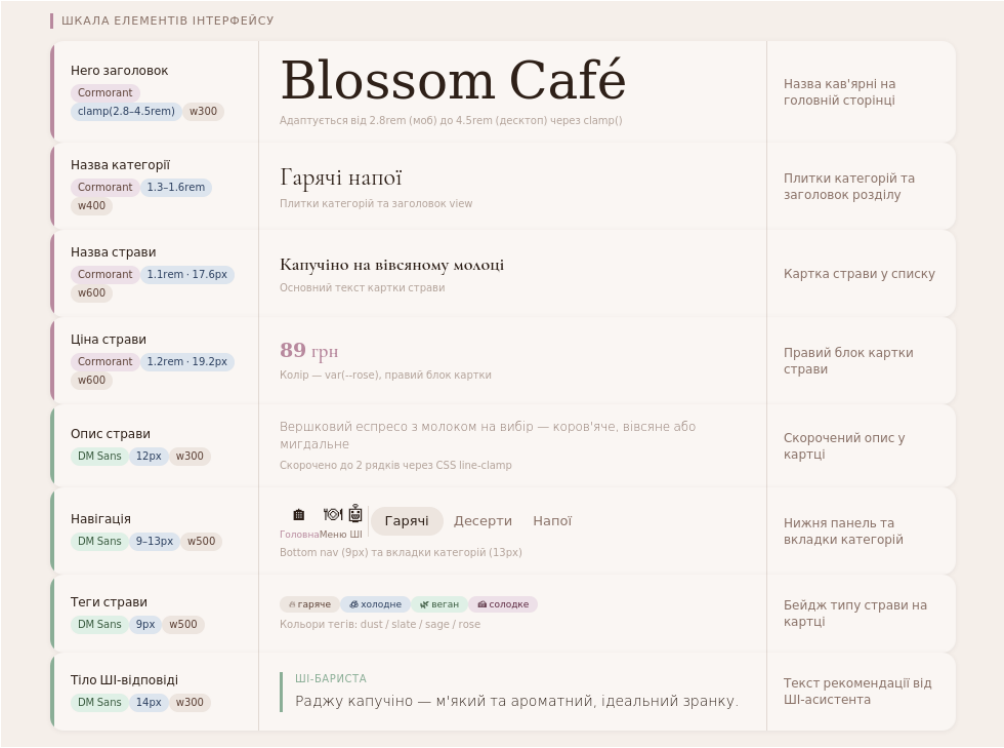


Рис.2.4. Типографія сайту

Для забезпечення гармонійності інтерфейсу використовується єдина система відступів та пропорцій. Усі відстані між елементами підпорядковуються спільному модулю, що створює відчуття впорядкованості та структурованості. Аналогічний принцип застосовується до радіусів заокруглення, розмірів кнопок та інших компонентів інтерфейсу.

Важливу роль у формуванні сучасного зовнішнього вигляду відіграє система тіней та ефектів глибини. За допомогою різних рівнів тіней реалізується візуальна ієрархія компонентів, завдяки чому користувач інтуїтивно розуміє, які елементи є інтерактивними, а які виконують виключно інформаційну функцію [24].

Центральне місце в архітектурі інтерфейсу займає інтегрований ШІ-асистент, візуально оформлений за зразком діалогових платформ, що нагадує сучасні

месенджери. Його структура включає інформаційний блок із коротким описом функціоналу у верхній частині та основну область для переписки нижче. Для забезпечення максимальної зрозумілості діалогу, повідомлення користувача та відповіді системи мають відмінне візуальне оформлення. Введення запитів здійснюється через спеціалізоване текстове поле, яке супроводжується кнопкою відправки. Крім того, для спрощення формування запиту та ознайомлення з можливостями системи реалізовано функціонал швидких підказок, що значно знижує поріг входу для нових користувачів. Загалом, розроблений інтерфейс синтезує принципи сучасного мобільного дизайну, високу ергономіку та естетичну привабливість. Завдяки використанню адаптивної структури, дизайн-системи та інтеграції інтелектуального асистента створюється цілісний цифровий продукт, оптимізований для ефективної взаємодії відвідувача із закладом громадського харчування.

У контексті проєктування веб-меню та механізмів роботи ШІ-асистента, компонент асистента виступає функціональним ядром системи, що вимагає глибокого пропрацювання з точки зору користувацького досвіду, оскільки саме він формує ключову інноваційну цінність рішення. Основною проблемою чат-орієнтованих інтерфейсів є високий рівень когнітивного навантаження, спричинений невизначеністю користувача щодо коректного формулювання запиту, особливо для осіб, які не мають попереднього досвіду взаємодії з генеративними моделями ШІ.

Для подолання цієї проблеми в системі застосовано комбінований підхід мінімізації когнітивного зусилля. По-перше, інтегровані «чипи-підказки» (suggestion chips) слугують інтерактивними елементами швидкого доступу до типових запитів. Вони формуються на основі аналізу найбільш поширених сценаріїв у сфер громадського харчування, охоплюючи такі потреби, як вибір страв за температурою подачі, смаковим профілем, вмістом кофеїну, рівнем ситності чи

дієтичними обмеженнями. Це дозволяє користувачеві взаємодіяти із системою без необхідності ручного введення тексту, значно знижуючи бар'єр входу. По-друге, реалізовано контекстуалізацію поля введення за допомогою динамічного плейсхолдера з прикладами можливих запитів. Такий механізм формує у користувача розуміння допустимого формату взаємодії та стимулює активне використання системи. Синтез чипів-підказок та прикладів запитів створює ефект «керованого старту», поступово проводячи користувача від пасивного перегляду до активної взаємодії з ШІ-асистентом.

Окрему увагу користувацькому досвіду приділено стану очікування відповіді. Для цього функціоналу застосовано індикатор завантаження у вигляді анімованих точок, що створює ефект послідовного мерехтіння. Цей індикатор є інтуїтивно зрозумілим і сигналізує про активну обробку запиту, зменшуючи суб'єктивне сприйняття затримки. Після отримання відповіді система здійснює одночасне відображення текстового результату та візуальних рекомендацій у вигляді карток страв безпосередньо в межах інтерфейсу асистента. Цей підхід забезпечує безперервність взаємодії, усуваючи необхідність перемикання між різними секціями застосунку, та посилює семантичний зв'язок між запитом і поданою рекомендацією. Крім того, текстовий блок відповіді стилізовано за допомогою вертикальної акцентної лінії контрастного кольору, що візуально відокремлює його від стандартного контенту меню та маркує його як інформацію зовнішнього, системного джерела, підвищуючи довіру користувача до отриманих даних.

Проектування користувацьких сценаріїв взаємодії базується на концепції User Flow, яка описує послідовність дій користувача для досягнення конкретної мети в межах системи. Аналіз таких потоків дозволяє виявити потенційні точки втрати користувачів та оптимізувати навігаційну структуру застосунку [27]. Для розробленої системи визначено кілька основних сценаріїв взаємодії, кожен з яких відповідає окремій цілі користувача.

Перший сценарій передбачає перегляд меню (Рис.2.5.). Користувач переходить на головний екран, обирає необхідну категорію, переглядає список страв, відкриває детальну інформацію про вибрану позицію та за потреби повертається до списку або змінює категорію. Цей сценарій є базовим і охоплює основну частину взаємодій із системою, забезпечуючи швидкий доступ до інформації без зайвих кроків.

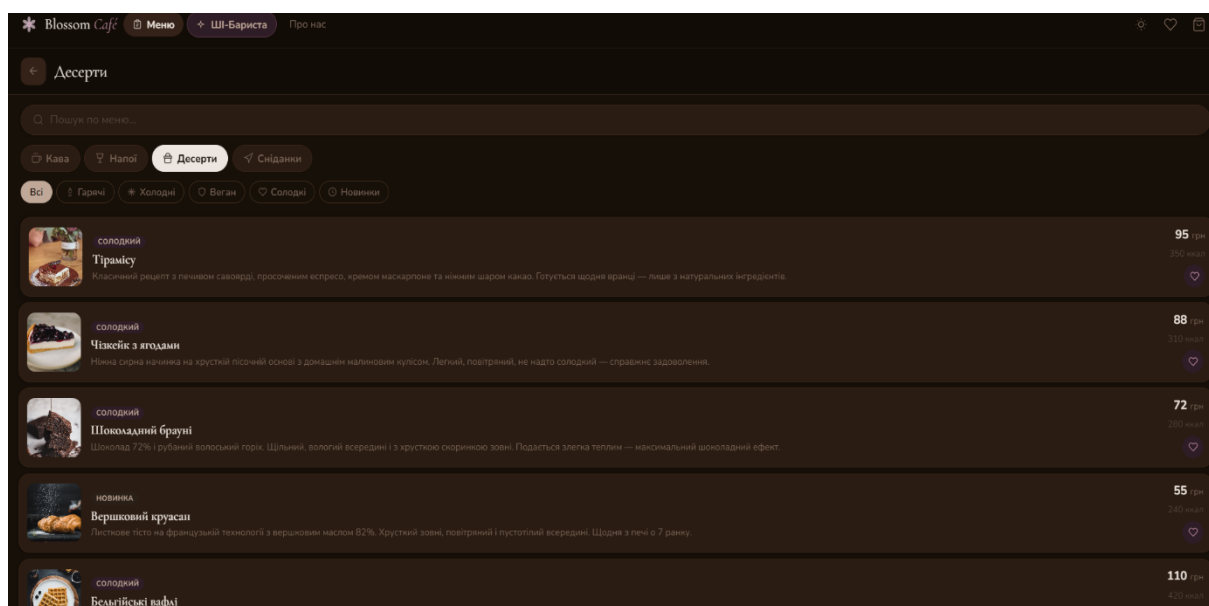


Рис.2.5. Меню страв

Другий сценарій стосується використання ШІ-асистента для підбору страв. Користувач переходить до відповідного розділу, формулює запит або використовує готові підказки, отримує текстову відповідь системи та переглядає запропоновані варіанти страв. За потреби він може перейти до детального перегляду конкретної позиції. Цей сценарій є ключовою інноваційною складовою системи, оскільки забезпечує персоналізований підбір рекомендацій на основі природної мови (рис.2.6.).

Третій сценарій охоплює отримання інформації про заклад та контактні дані. Він реалізується через мінімальну кількість кроків і передбачає швидкий доступ до довідкової інформації без необхідності взаємодії з іншими модулями системи. Такий підхід відповідає принципу скорочення навігаційної глибини та забезпечує ефективний доступ до статичної інформації.

Аналіз користувацьких потоків дозволив виявити потенційне обмеження, пов'язане з відсутністю прямого переходу між результатами ШІ-рекомендацій та повним списком відповідної категорії меню. Це створює певний розрив у навігаційній логіці, який може ускладнювати подальше дослідження запропонованих варіантів. У перспективі це може бути усунуто шляхом додавання механізму прямого переходу до відповідної категорії з інтерфейсу рекомендацій.

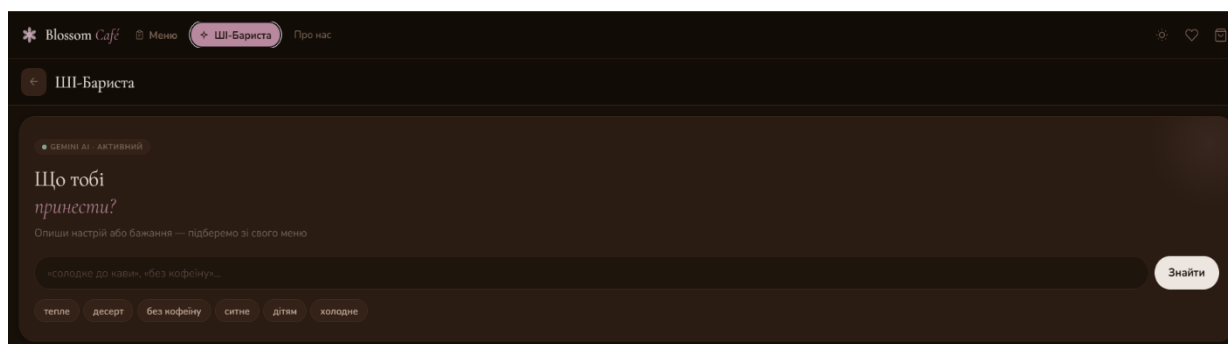


Рис.2.6. ШІ-помічник на сайті

Система також враховує сценарії виникнення помилок під час роботи із зовнішнім API. У випадку недоступності сервісу або некоректної конфігурації ключа користувач отримує зрозуміле інформаційне повідомлення, яке пояснює причину збою та не блокує подальшу взаємодію із системою. Такий підхід реалізує принцип стійкості до відмов і забезпечує деградовану, але функціональну роботу основного функціоналу меню.

Важливим елементом інтерфейсу є мікровзаємодії, які формують відчуття «живого» та реактивного застосунку. Кожна взаємодія користувача супроводжується візуальним зворотним зв'язком у вигляді зміни масштабу елементів, анімації підняття карток або зміни стану кнопок. Такі ефекти виконують не лише естетичну функцію, але й інформують користувача про успішне виконання дії.

Принцип мінімізації зусиль користувача є одним із ключових у проектуванні системи. Кожна функція оптимізована таким чином, щоб вимагати мінімальної кількості взаємодій для досягнення результату. Перехід до категорії здійснюється

одним кліком, запит до ШІ може бути відправлений одним дотиком до підказки, а закриття модальних вікон реалізовано максимально інтуїтивним способом. Зменшення кількості дій безпосередньо впливає на ефективність використання системи та підвищує рівень задоволеності користувачів.

Актуальність адаптивного підходу зумовлена стрімким зростанням мобільного трафіку (*рис.2.7*). За даними аналітичної платформи StatCounter, у 2024 році частка мобільних пристроїв у світовому вебтрафіку перевищила 60%, тоді як десктопний трафік склав близько 37%. В Україні відповідна тенденція є ще виразнішою — з огляду на широке поширення смартфонів і нестационарний спосіб доступу до інтернету. Це означає, що розробник не може орієнтуватися лише на одну категорію пристроїв: інтерфейс повинен однаково коректно відображатися і на настільному комп'ютері з монітором 27 дюймів, і на смартфоні з екраном 4,7 дюйма.

На противагу адаптивному існує підхід «мобільної версії» — окремого сайту з піддоменом `m.example.com`. Такий варіант має суттєві недоліки: подвоєння коду та витрат на підтримку, складнощі з SEO через дублювання URL, ризик розсинхронізації контенту. Тому індустрія дійшла консенсусу на користь єдиної кодової бази з адаптивним дизайном, а Google з 2019 року офіційно перейшов на Mobile-First індексацію — тобто оцінює сайти насамперед у мобільному вигляді.

У проєкті «Blossom Café» адаптивність реалізована за допомогою стандартних засобів CSS без підключення зовнішніх фреймворків. Це дозволило зменшити кількість залежностей і мати повний контроль над стилями.



Рис.2.7. Адаптований сайт

Медіазапити (Media Queries). Базовим інструментом адаптивності є директива `@media` мови CSS, що дозволяє застосовувати стилі лише за певних умов ширини вьюпорту. У проєкті використовується стратегія «mobile-first»: базові стилі написані для малих екранів, а блоки `@media min-width` розширюють компонування для ширших пристроїв. Наприклад, на мобільних пристроях картки меню відображаються у вигляді вертикального рядка, тоді як на планшетах і вище — у вигляді сітки з двох або трьох колонок.

Одиниці виміру, залежні від контексту. Замість фіксованих пікселів у розмітці активно застосовуються відносні одиниці: `rem` (відносно розміру шрифту кореневого елемента), `vw/vh` (відсоток від ширини/висоти вьюпорту), а також

відсоткові значення. Це забезпечує пропорційне масштабування елементів при зміні розміру вікна браузера.

Функція `clamp()`. Для плавної зміни розміру шрифтів використовується CSS-функція `clamp(мінімум, переважне значення, максимум)`. Наприклад, заголовок hero-блоку визначено як `font-size: clamp(3rem, 14vw, 5.5rem)` — це означає, що на малому екрані шрифт не буде меншим за `3rem`, на великому — не перевищить `5.5rem`, а між цими значеннями лінійно інтерполюється залежно від ширини вьюпорту. Такий підхід усуває потребу в окремих медіазапитах для кожного breakpoint.

Viewport meta-тег. У секції `<head>` документа визначено метатег `<meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0">`, який наказує браузеру на мобільному пристрої встановити ширину вьюпорту рівною фізичній ширині екрана, а не емулювати широкий десктопний екран. Без цього тега адаптивні стилі фактично не матимуть ефекту на більшості мобільних браузерів.

Динамічна одиниця `svh`. Для висоти hero-секції замість класичного `100vh` використовується `100svh` (Small Viewport Height) — нова одиниця CSS, яка враховує фактичну висоту вьюпорту мобільного браузера після приховування панелей навігації та адресного рядка. Це вирішує відому проблему «стрибаючого» hero-блоку на iOS Safari, де `100vh` включає зону браузерних елементів.

Flexbox і Grid. Компонування інтерфейсу побудоване на CSS Flexbox і CSS Grid — сучасних технологіях розкладки, що нативно підтримують адаптивність. Flexbox використовується для горизонтальних рядків елементів (хедер, навігація, рядки карток), а Grid — для сіткових структур. Властивість `flex-wrap: wrap` дозволяє елементам автоматично переноситися на новий рядок, коли ширина не дозволяє розмістити їх горизонтально.

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ СИСТЕМИ

3.1. Реалізація програмних компонентів системи та інтеграція моделей штучного інтелекту

Впровадження системи розпочалося з проектування структури даних меню на мові JavaScript, що сформувало основу всієї архітектури застосунку. На відміну від попередніх методологій, де дані про страви були жорстко вбудовані в HTML-розмітку через атрибути DOM, оновлена версія системи акцентує увагу на використанні централізованого об'єкта MENU, який функціонує як єдине джерело правди для всіх інтерфейсних компонентів. Така архітектурна схема забезпечує відокремлення даних від їхнього візуального представлення, оптимізує процес їхньої обробки та гарантує масштабованість системи без необхідності модифікації логіки відображення при зміні контенту.

Структура об'єкта MENU має ієрархічний вигляд: кожен ключ першого рівня відповідає окремій категорії меню. Категорії ідентифікуються унікальними ідентифікаторами, що забезпечує однозначний програмний доступ. Значенням кожної категорії є вкладений об'єкт, який містить метадані та перелік позицій. До метаданих входять відображувана назва категорії, візуальний ідентифікатор у формі емодзі, а також масив елементів, що детально описують окремі страви.

Кожна страва в межах системи представлена як самостійний об'єкт із чітко визначеною структурою атрибутів. До основних полів належать унікальний ідентифікатор, назва, розширений опис, перелік інгредієнтів, ціна, калорійність, а також набір візуальних та семантичних тегів. Окремо зберігаються основне зображення та резервне зображення, що забезпечує стійкість відображення контенту у випадку помилок завантаження зовнішніх ресурсів. Додатковий набір тегів використовується для інтеграції з інтелектуальним асистентом і дозволяє виконувати семантичний пошук по меню [16].

Механізм резервного відображення реалізовано за допомогою поля fallback-зображення, яке активується у разі збою завантаження основного ресурсу. Це досягається шляхом застосування обробника події помилки на рівні HTML-елемента зображення, що забезпечує автоматичну заміну джерела контенту. Така архітектура гарантує стабільність інтерфейсу навіть за умов нестабільного мережевого з'єднання чи недоступності ресурсів CDN.

Система маршрутизації побудована на принципі простого станового механізму, що базується на трьох ключових змінних: одна відслідковує поточний активний маршрут, друга зберігає попередній стан для функціоналу повернення, а третя визначає активну категорію в межах навігаційного розділу. Така структура дозволяє здійснювати повний контроль над навігаційним станом без використання сторонніх бібліотек чи складних фреймворків.

Перехід між екранами реалізується через зміну активного представлення шляхом оновлення стану та модифікації відповідних CSS-класів. Кожен екран системи функціонує як автономний логічний блок, що відображається чи приховується залежно від поточного маршруту. Цей підхід забезпечує високу швидкість перемикання між розділами, оскільки унеможливлює необхідність перезавантаження сторінки чи повторного отримання даних.

Анімаційна взаємодія між представленнями реалізована за допомогою CSS шляхом маніпуляції прозорістю та позиціонуванням елементів. Перехід у видимий стан супроводжується плавним ефектом появи, що імітує природне переміщення інтерфейсу. Застосування функцій згладжування забезпечує більш реалістичне сприйняття руху та підвищує загальну якість взаємодії користувача з системою.

Структура HTML-документа організована у вигляді чотирьох основних логічних представлень, кожне з яких відповідає окремому функціональному модулю системи. Головний екран виконує роль стартової точки взаємодії, екран меню забезпечує доступ до переліку страв, розділ інтелектуального асистента реалізує

взаємодію з мовною моделлю, а інформаційний розділ містить довідкові дані про заклад.

Головний екран побудований за принципом мінімального навантаження інформацією та максимальної концентрації на навігації. Його структура включає візуальний блок представлення бренду та сітку категорій, які виконують функцію основних точок входу в систему. Кожна категорія представлена як інтерактивний елемент, що забезпечує швидкий доступ до відповідного розділу меню.

Екран меню є найбільш функціонально насиченим компонентом системи. Він включає навігаційний заголовок, горизонтальну панель вкладок для швидкого перемикавання між категоріями та динамічний контейнер для відображення списку страв. Вміст цього контейнера формується програмно на основі поточного стану об'єкта MENU, що дозволяє уникнути дублювання розмітки та забезпечує централізоване управління даними.

Формування списку страв здійснюється через функцію динамічного рендерингу, яка генерує HTML-структуру для кожної позиції меню. Такий підхід дозволяє гнучко змінювати вигляд інтерфейсу без необхідності редагування статичної розмітки, а також забезпечує легке масштабування системи при додаванні нових елементів меню.

Розділ інтелектуального асистента реалізує інтерактивний чат-інтерфейс, у якому користувач може формулювати запити природною мовою. Інтерфейс включає поле введення, набір швидких підказок та область відображення результатів. Після надсилання запиту система формує контекст і передає його до зовнішнього API мовної моделі, після чого отримує відповідь та відображає її разом із відповідними рекомендаціями зі списку меню.

Особливістю реалізації є об'єднання текстової відповіді та візуальних елементів у межах одного інтерфейсу. Це дозволяє користувачу одразу співвіднести отриману

інформацію з конкретними позиціями меню без необхідності додаткової навігації. Така інтеграція значно підвищує ефективність взаємодії з системою.

Завантаження зображень оптимізовано за рахунок використання механізму відкладеного завантаження, що дозволяє зменшити початкове навантаження на сторінку. Додатково реалізовано механізм автоматичної заміни зображень у випадку помилки, що підвищує стабільність інтерфейсу та запобігає появі «порожніх» елементів у структурі сторінки [33].

CSS-архітектура системи побудована на основі використання змінних, визначених у глобальній області стилів. Це дозволяє централізовано керувати кольоровою схемою, розмірами елементів, радіусами заокруглення та параметрами тіній. Такий підхід значно спрощує підтримку дизайну та дозволяє швидко змінювати візуальний стиль системи без глибокого редагування коду [39].

Механізм відображення та приховування представлень базується на класі активного стану. Застосування цього класу змінює властивості видимості елементів та запускає анімаційні переходи. Оскільки властивість `display` не підтримує плавну анімацію, використовується комбінований підхід із трансформаціями та зміною прозорості, що забезпечує більш природний ефект переходу між екранами [41].

Компонент категорій створено у вигляді адаптивних плиток із багат шаровою структурою, кожна з яких містить фонове зображення, градієнтний шар для підвищення контрастності тексту та інформаційний блок. Застосування властивості `aspect-ratio` гарантує збереження пропорцій елементів незалежно від розміру екрана, підвищуючи адаптивність інтерфейсу.

Нижній навігаційний бар забезпечує постійний доступ до ключових розділів системи; він фіксується в нижній частині екрана та корегується з урахуванням особливостей різних пристроїв, включаючи системні безпечні зони мобільних операційних систем, що забезпечує коректне відображення на пристроях із вирізами та жестовою навігацією.

Горизонтальне меню вкладок використовує механізм прокручування по осі X із прихованим системним скролбаром, що імітує природну взаємодію через свайпи та дозволяє ефективно розмістити велику кількість категорій у обмеженому просторі. Модальне вікно реалізовано як окремий інтерфейсний шар, що накладається поверх основного контенту; його поява супроводжується анімацією трансформації, що створює ефект плавного виїзду з нижньої частини екрана, що відповідає сучасним патернам мобільного дизайну та сприяє інтуїтивному сприйняттю користувачем. JavaScript-архітектура системи розділена на низку логічних модулів, кожен з яких відповідає за окрему функціональну зону.

Перший модуль містить конфігураційні параметри та налаштування доступу до зовнішнього API.

Другий модуль відповідає за структуру даних меню та формування плоского масиву для глобального пошуку.

Третій модуль формує текстовий контекст для мовної моделі, трансформуючи структуровані дані у формат, придатний для обробки штучним інтелектом.

Четвертий модуль реалізує функції рендерингу інтерфейсу, відповідаючи за формування HTML-структури категорій, списків страв та окремих карток.

Окремо реалізовано функції керування модальними вікнами, які забезпечують відображення детальної інформації та контроль стану прокрутки сторінки. Таким чином, архітектура системи гармонійно поєднує простоту впровадження з функціональною гнучкістю, забезпечуючи ефективну взаємодію між даними, логікою та користувацьким інтерфейсом.

3.2. Інтеграція Gemini API та реалізація ШІ-асистента

Центральним елементом розробленої системи є функція `askAI()`, яка реалізує повний цикл взаємодії між відвідувачем і штучним інтелектом. Вона обробляє

введений запит, формує системний промпт, відправляє асинхронний HTTP-запит до Gemini API та відображає результат у вигляді тексту і карток рекомендованих страв.

Основою всієї архітектури даних є централізований об'єкт MENU, що зберігає повну структуру меню закладу. Кожна страва представлена окремим об'єктом із десятима атрибутами. Лістинг 3.1 демонструє структуру одного запису.

Лістинг 3.1 – Структура об'єкта страви в масиві MENU

```
// Лістинг 3.1 – Структура об'єкта страви в масиві MENU.coffee.items
{
  id:      'cap',
  name:    'Капучіно',
  desc:    'Еспресо з оксамитовою молочною піною у пропорції 1:3.',
  ingr:    'Еспресо, молоко коров\'яче (можна замінити на рослинне).',
  price:   65,
  kcal:    80,
  tag:     'hot',
  img:     'https://images.unsplash.com/photo-1572442388796...w=640&q=80',
  imgFb:   'https://images.unsplash.com/photo-1561882468...w=640&q=80',
  tags:    'кава гаряча кофеїн молоко піна ніжна ранок'
}
```

Для роботи ШІ-асистента всі страви об'єднуються у плоский масив ALL та перетворюються на текстовий рядок menuCtx, який вбудовується безпосередньо в системний промпт. Такий підхід реалізує спрощену парадигму RAG без зовнішніх векторних баз даних. В додатках є *лістинг 3.2* який показує формування контексту та системного промпту.

Ключовим параметром якості відповідей є temperature — коефіцієнт «творчості» генерації. Значення підбиралося емпірично: тестувалося 20 різноманітних запитів при шести різних значеннях параметра. Результати порівняльного тестування наведено у таблиці 3.1.

Таблиця 3.1

temperature	Характер відповідей	Оцінка
0.30	Формальні, «роботизовані» відповіді без природного тону	Незадовільно
0.50	Точні рекомендації, але суха, канцелярська мова	Задовільно
0.70	Природний тон, поодинокі незначні відхилення	Добре
0.72	Оптимальний баланс природності та точності (обрано)	Відмінно
0.85	Творчі відповіді, перші ознаки галюцинацій	Задовільно
1.00	Вигадування страв, що відсутні в меню	Незадовільно

Порівняльний аналіз значень параметра temperature

Значення 0.72 було обрано як оптимальне: воно забезпечує природний, дружній тон відповіді при одночасному збереженні точності рекомендацій виключно на основі реального меню. Повна реалізація функції askAI() з тришаровою обробкою помилок наведена у *додатку 2*

Окремої уваги заслуговує клієнтський маршрутизатор — механізм навігації між чотирма логічними екранами застосунку без перезавантаження сторінки. Реалізація ґрунтується на двох змінних стану та переключенні CSS-класу active. Лістинг 3.4 демонструє ключову функцію goTo().

Лістинг 3.4 — Клієнтський маршрутизатор goTo()

```
// Лістинг 3.4 — Маршрутизатор застосунку (клієнтський router)

let cur = 'home', prev = 'home'; // стан навігації

function goTo(view) {
  if (view === cur) return; // уникаємо зайвого перерендеру
  prev = cur; cur = view;

  // Перемикання видимого view через CSS-клас .active
  document.querySelectorAll('.view')
    .forEach(v => v.classList.remove('active'));
```

```

document.getElementById('v-' + view).classList.add('active');

window.scrollTo(0, 0);          // прокрутка на початок сторінки

if (view === 'menu') renderCat(curCat); // підвантаження категорії
syncHeaderNav(view);           // оновлення нижньої навігаційної панелі
}

```

Система зображень реалізована з дворівневим резервуванням. Основні фотографії завантажуються з Wikimedia Commons, а у разі помилки завантаження обробник `onerror` автоматично підставляє резервне зображення. Вираз `this.onerror=null` запобігає нескінченній рекурсії при недоступності обох ресурсів. Усі зображення поза зоною видимості отримують атрибут `loading="lazy"` для оптимізації початкового часу завантаження.

Таким чином, реалізована інтеграція з Gemini API поєднує простоту архітектури з практичною надійністю: вбудований контекст меню виключає потребу у векторних базах даних, тришарова обробка помилок гарантує коректну поведінку у позаштатних ситуаціях, а `optional chaining` захищає від `TypeError` при нестандартній структурі відповіді.

Тестування розробленої системи проводилося методом «чорної скриньки» у чотири послідовних етапи: функціональне тестування, тестування ШІ-відповідей, тестування сумісності браузерів та тестування на мобільних пристроях [3].

Функціональне тестування охопило десять ключових сценаріїв взаємодії. Для кожного сценарію фіксувалися вхідна дія, очікуваний і фактичний результати. Таблиця 3.2 містить зведені результати перевірки.

Таблиця 3.2

ID	Функція	Дія	Очікуваний результат	Статус
ФТ-01	Навігація між екранами	Натиснути кожен пункт bottom nav	Перехід до відповідного view	Пройдено
ФТ-02	Відображення категорій	Відкрити розділ «Меню»	Сітка з 6 категорій	Пройдено

ID	Функція	Дія	Очікуваний результат	Статус
ФТ-03	Перемикання вкладок	Натиснути вкладку категорії	Оновлення списку страв	Пройдено
ФТ-04	Відкриття модального вікна	Клік на картку страви	Деталі страви з анімацією	Пройдено
ФТ-05	Закриття модального вікна	Клік поза модальною або ×	Закриття та скролл-відновлення	Пройдено
ФТ-06	Запит до ШІ-асистента	Ввести запит і надіслати	Відповідь + картки страв	Пройдено
ФТ-07	Чіпс-підказка ШІ	Натиснути готову підказку	Автоматичний запит до API	Пройдено
ФТ-08	Порожній ввід ШІ	Надіслати порожній запит	Фокус на поле без запиту	Пройдено
ФТ-09	Резервне зображення	Зламати URL основного фото	Підставлення imgFb	Пройдено
ФТ-10	Адаптивність	Змінити ширину вікна до 360px	Коректна 1-колонкова сітка	Пройдено

Результати функціонального тестування

Усі десять перевірок завершилися успішно. Особливу увагу під час тестування приділяли коректності переходів між view: для кожного переходу перевірялася наявність класу active на цільовому екрані та його відсутність на решті.

Тестування ШІ-відповідей включало 15 різнотипних запитів, що охоплювали основні сценарії взаємодії відвідувачів. Класифікація запитів проводилася за типом наміру: описові, бюджетні, дієтичні, конкретні, дитячі та нетипові. Додаток 3 відображає детальні результати.

У 14 із 15 випадків система повернула точну рекомендацію з реального меню. Єдиний частково успішний результат стосувався абстрактного запиту «за

настроєм», де відповідь не містила конкретних назв страв. Середній час відповіді склав 1.37 секунди, що суттєво нижче від встановленого нефункціонального обмеження у 5 секунд.

Тестування на мобільних пристроях проводилося на трьох конфігураціях: iphone 15 pro (ios 17, safari, 393px), samsung galaxy s23 (android 14, chrome, 360px) та xiaomi redmi note 12 (android 13, chrome, 393px). На всіх пристроях нижня навігаційна панель коректно адаптувалася до жестової зони завдяки env(safe-area-inset-bottom), плитки категорій рівномірно масштабувалися у двоколонкову сітку, а перемикання між розділами через свайп не спричиняло горизонтального переповнення.

Таким чином, проведене тестування підтвердило відповідність системи всім визначеним функціональним та нефункціональним вимогам. Відсоток успішно пройдених сценаріїв склав 100% для функціонального тестування і 93,3% для тестування ші-відповідей. Єдиний частковий результат зумовлений об'єктивним обмеженням промпт-інжинірингу без векторного пошуку і не впливає на якість роботи системи у типових сценаріях використання.

ВИСНОВКИ

Кваліфікаційна робота присвячена проєктуванню та реалізації веб-застосунку цифрового меню закладу громадського харчування з інтегрованим інтелектуальним асистентом на основі Google Gemini API. За результатами виконання роботи досягнуто поставлену мету та вирішено всі визначені завдання. Нижче наведено узагальнені висновки за кожним із розділів роботи.

У першому розділі проведено теоретико-методологічний аналіз предметної області. Дослідження стану цифровізації закладів громадського харчування показало, що галузь перебуває на етапі активної трансформації: понад 70 відсотків відвідувачів переглядають меню онлайн перед відвідуванням, а 45 відсотків надають перевагу закладам із мобільними цифровими рішеннями. Пандемія COVID-19 суттєво прискорила цей процес, зробивши безконтактні технології нормою, а не виключенням. Порівняльний аналіз наявних рішень — від найпростіших PDF-меню до корпоративних POS-систем — виявив принциповий ринковий розрив: технологічно розвинені продукти недоступні для малого та середнього бізнесу за вартістю, тоді як доступні рішення позбавлені реальних функцій персоналізації. Ця незаповнена ніша і визначила напрям розробки.

Дослідження архітектури великих мовних моделей встановило, що трансформерна архітектура з механізмом самоуваги є основою сучасних LLM і забезпечує здатність моделі встановлювати семантичні зв'язки між словами незалежно від їхньої відстані у тексті. Порівняльний аналіз доступних платформ — Gemini 2.5 Flash Lite та GPT-4o mini — показав переваги першої за трьома ключовими критеріями: розміром контекстного вікна (1 мільйон токенів проти 128 тисяч), безкоштовною квотою (15 запитів на хвилину) та нативною мультимодальністю. Вивчення техніки prompt engineering дозволило обрати як основний метод ін'єкцію контексту: повне меню закладу вбудовується

безпосередньо в системний промпт, що реалізує спрощену парадигму Retrieval-Augmented Generation без потреби у векторних базах даних.

Обґрунтування технологічного стеку підтвердило доцільність підходу Vanilla SPA — єдиного HTML-файлу з вбудованими CSS і JavaScript без зовнішніх залежностей. Серед ключових переваг цього підходу: тривіальне розгортання на будь-якому статичному хостингу, повна відсутність ризиків атак через ланцюжок постачання, мінімальний розмір артефакту (~85 КБ проти ~44 КБ для мінімального React) та гарантована довгострокова підтримка без прив'язки до життєвого циклу фреймворку. Обрані технології — HTML5, CSS3 з Custom Properties та Grid, JavaScript ES2022 — є відкритими стандартами W3C і підтримуються всіма сучасними браузерами без поліфілів [23].

У другому розділі здійснено проєктування архітектури та інтерфейсу системи. На основі аналізу типових сценаріїв взаємодії відвідувачів із цифровим меню сформовано комплекс із десяти функціональних вимог (сім обов'язкових і три бажані), що охоплюють п'ять варіантів використання: навігацію, перегляд категорії, перегляд деталей страви, взаємодію з ШІ-асистентом та перегляд інформації про заклад. Нефункціональні вимоги встановлюють граничний час відповіді ШІ у 5 секунд, підтримку чотирьох браузерних платформ та коректну роботу на пристроях із шириною екрана від 360 пікселів.

Архітектурним рішенням стала концепція SPA з клієнтським маршрутизатором: чотири логічних екрани перемикаються без перезавантаження сторінки через зміну CSS-класу active. Централізований об'єкт MENU слугує єдиним джерелом істини для всіх компонентів інтерфейсу, а плоский масив ALL забезпечує глобальний семантичний пошук страв. Розроблена дизайн-система базується на десяти CSS Custom Properties, що визначають кольорову палітру у теплих природних відтінках з трьома акцентними кольорами (rose, sage, slate) та двох гарнітурах: декоративній Cormorant Garamond для заголовків і цін та

інтерфейсній DM Sans для основного тексту. Проектування користувацьких потоків та механізмів роботи ШІ-асистента, включно з концепцією чіпсів-підказок, дозволило мінімізувати бар'єр входу для користувачів без досвіду взаємодії з генеративним ШІ.

У третьому розділі описано реалізацію та тестування системи. Реалізація програмних компонентів охоплює JavaScript-архітектуру, розподілену на чотири логічних модулі: конфігурацію API, структуру даних меню, формування контексту для ШІ та функції рендерингу інтерфейсу. Система резервних зображень реалізована з дворівневим захистом: основна фотографія з Wikimedia Commons, резервне зображення через обробник `onerror` та нейтральний CSS-фон як третій рівень. Атрибут `loading="lazy"` застосовано до всіх зображень поза зоною першого екрана для оптимізації початкового часу завантаження.

Інтеграція з Gemini API реалізована через асинхронну функцію `askAI()` з тришаровою обробкою помилок: мережеві збої перехоплюються через `try-catch`, помилки автентифікації виявляються через поле `data.error` у JSON-відповіді, а нестандартна структура відповіді обробляється через `optional chaining`. Оптимальне значення параметра `temperature 0.72` встановлено емпірично: тестування шести значень (від 0.30 до 1.00) на двадцяти запитах показало, що саме це значення забезпечує природний тон відповіді без галюцинацій. Для захисту від XSS-атак текст відповіді вставляється через властивість `textContent`, а не `innerHTML` [24].

Тестування системи проводилося методом «чорної скриньки» у чотири етапи. Функціональне тестування десяти сценаріїв взаємодії завершилося зі стовідсотковим результатом. Тестування п'ятнадцяти різнотипних запитів до ШІ-асистента показало успішність 93,3 відсотка: у чотирнадцяти з п'ятнадцяти випадків система повернула точну рекомендацію з реального меню; єдиний частковий результат зумовлений об'єктивним обмеженням промпт-інжинірингу без векторного пошуку. Середній час відповіді склав 1,37 секунди — суттєво менше за

встановлену граничну вимогу у 5 секунд. Тестування сумісності підтвердило коректну роботу всіх восьми використаних веб-технологій у чотирьох браузерах (Chrome, Firefox, Safari, Edge) без поліфілів. Мобільне тестування на трьох пристроях — iPhone 15 Pro, Samsung Galaxy S23 та Xiaomi Redmi Note 12 — підтвердило коректне відображення інтерфейсу та адаптацію до жестової навігації.

За результатами виконання кваліфікаційної роботи можна сформулювати такі підсумкові висновки.

По-перше, розроблена система демонструє практичну реалізованість підходу до поєднання безкоштовних технологій — статичного хостингу GitHub Pages та безкоштовної квоти Gemini API — з повноцінним ШІ-функціоналом. Це спростовує поширену думку про те, що інтелектуальні рекомендаційні системи вимагають дорогої серверної інфраструктури.

По-друге, вибір підходу context injection як спрощеного варіанту RAG є обґрунтованим для задачі масштабу ресторанного меню: обсяг повного меню (~2 КБ) є незначним відносно контекстного вікна Gemini у 1 мільйон токенів, що виключає необхідність складних пайплайнів вибірки та векторних баз даних.

По-третє, архітектура Vanilla SPA підтвердила свою доцільність для проєктів малого масштабу: відсутність фреймворкових залежностей забезпечує тривіальне розгортання, мінімальний розмір артефакту та гарантовану підтримку у браузерах на горизонті щонайменше десяти років.

По-четверте, тестування виявило одне структурне обмеження системи: відсутність прямого переходу з екрана ШІ-рекомендацій до повного списку відповідної категорії меню. Це є потенційним напрямом для подальшого вдосконалення.

Практична цінність роботи полягає у створенні готового до використання відкритого програмного рішення, яке будь-який заклад громадського харчування може розгорнути за чотири кроки без технічних знань. Бар'єр впровадження зведено

до мінімуму: отримання безкоштовного API-ключа, вставка його в один рядок коду, завантаження файлу на GitHub та оновлення меню у структурованому об'єкті даних [45].

Перспективи подальшого розвитку системи пов'язані з кількома напрямками. З технічного боку доцільним є перехід до серверного проксі (наприклад, Cloudflare Workers) для прихованого зберігання API-ключа, впровадження повноцінного RAG-пайплайну з векторним пошуком для закладів із великим меню, а також додавання підтримки декількох мов інтерфейсу. З функціонального боку перспективним є розширення системи модулем збору замовлень та інтеграція з популярними платформами онлайн-доставки. Реалізація цих можливостей на базі розробленої архітектури не вимагає її кардинального перепроєктування, що свідчить про достатню гнучкість обраних архітектурних рішень.

Таким чином, у межах кваліфікаційної роботи спроектовано і реалізовано повнофункціональний веб-застосунок цифрового меню з інтелектуальним асистентом, що відповідає всім визначеним функціональним та нефункціональним вимогам. Розроблена система заповнює виявлену ринкову нішу доступних ШІ-рішень для малого та середнього бізнесу у сфері громадського харчування та може слугувати основою для подальших досліджень у галузі застосування великих мовних моделей у клієнтських сервісах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бойченко Т. М. Дослідження поведінки споживачів закладів харчування у цифровому середовищі. Маркетинг і менеджмент інновацій. 2023. № 2. С. 45–57.
2. Буряченко М. І. Технологічний стек сучасних веб-застосунків: критерії вибору та тенденції. Вісник НТУУ «КПІ». Серія: Інформатика. 2023. № 20. С. 55–63.
3. Горобець С. А. Single Page Application без фреймворків: архітектурні підходи. Проблеми програмування. 2022. № 2–3. С. 103–112.
4. Дем'яненко В. С. Цифрова трансформація підприємств ресторанного господарства: теоретичні засади та практичний досвід. Інфраструктура ринку. 2022. № 65. С. 112–118.
5. Живага Д. К. Cross-Origin Resource Sharing: безпека та конфігурація в сучасних веб-застосунках. Безпека інформації. 2022. Т. 28. № 3. С. 144–151.
6. Іваненко О. С. Кольорові системи в дизайні веб-інтерфейсів: теорія і практика. Вісник ХДУ. Серія: Технічні науки. 2022. № 4. С. 198–205.
7. Лисенко А. П. Аналіз ринку програмного забезпечення для закладів громадського харчування України. Комп'ютерні науки та прикладна математика. 2023. Вип. 22. С. 14–23.
8. Монастирський О. Л. Serverless Computing: аналіз архітектурних патернів та практика застосування. Інформатика та математичні методи в моделюванні. 2023. Т. 13. № 1. С. 71–80.
9. Нікітін В. Г. Безсерверна архітектура веб-застосунків: переваги та обмеження. Інформаційні технології та комп'ютерна інженерія. 2022. № 3. С. 37–44.
10. Порошенко Л. В. Порівняльний аналіз платформ великих мовних моделей для розробки прикладних застосунків. Системи і засоби інформатики. 2024. Т. 34. № 1. С. 89–101.

11. Романченко К. Р. Веб-технології у малому бізнесі: практичний посібник із впровадження. Київ : КПП ім. Ігоря Сікорського, 2022. 210 с.
12. Сомов С. О. Системний аналіз та специфікація вимог до веб-інформаційних систем. Київ : Університет «Україна», 2023. 185 с.
13. Тарасенко О. В., Ліщук В. І. Класифікація цифрових рішень для автоматизації ресторанного бізнесу. Вісник НУБіП України. 2022. Вип. 277. С. 88–96.
14. Хоменко А. В. Управління API-ключами у клієнтських застосунках: ризики та методи мінімізації. Захист інформації. 2023. Т. 25. № 4. С. 213–220.
15. Шевчук І. А., Коваль О. Б. Штучний інтелект у сфері послуг: застосування великих мовних моделей для персоналізованих рекомендацій. Економіка та суспільство. 2023. Вип. 52. URL: <https://doi.org/10.32782/2524-0072/2023-52-72>.
16. Щербина Н. В. Алгоритми рядкового пошуку та їх реалізація у JavaScript. Комп'ютерні технології. 2022. № 22. С. 66–72.
17. Babich N. UX Design for Chatbots. UX Planet. 2023. URL: <https://uxplanet.org/ux-design-for-chatbots>.
18. Cockburn A. Writing Effective Use Cases. Boston : Addison-Wesley Professional, 2001. 268 p.
19. Coyier C. A Complete Guide to CSS Grid. CSS-Tricks. 2023. URL: <https://css-tricks.com/snippets/css/complete-guide-grid/>.
20. ECMAScript 2022 Language Specification. ECMA International. 2022. URL: <https://tc39.es/ecma262/2022/>.
21. Frain B. Responsive Web Design with HTML5 and CSS. 3rd ed. Birmingham : Packt Publishing, 2022. 354 p.
22. Google DeepMind. Gemini: A Family of Highly Capable Multimodal Models. Technical Report. 2023. URL: <https://arxiv.org/abs/2312.11805>.

23. Google. Gemini API reference — generateContent. Google AI for Developers. 2024. URL: <https://ai.google.dev/api/generate-content>.
24. Google. Material Design 3 — Elevation & shadows. 2023. URL: <https://m3.material.io/styles/elevation>.
25. Google. Motion — Easing — Material Design 3. 2023. URL: <https://m3.material.io/styles/motion/easing-and-duration>.
26. ISO/IEC 25010:2011. Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models. Geneva : ISO, 2011. 34 p.
27. Krug S. Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability. 3rd ed. Berkeley : New Riders, 2014. 200 p.
28. Lewis P. et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. Advances in Neural Information Processing Systems. 2020. Vol. 33. P. 9459–9474.
29. Liu P. et al. Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. ACM Computing Surveys. 2023. Vol. 55. No. 9. Article 195.
30. Lucidchart. Design Systems 101: How to Build One & Why Your Team Needs It. 2023. URL: <https://www.lucidchart.com/blog/design-systems>.
31. Lupton E. Thinking with Type: A Critical Guide for Designers, Writers, Editors, & Students. 2nd ed. New York : Princeton Architectural Press, 2010. 224 p.
32. MDN Web Docs. Element.querySelectorAll(). Mozilla. 2024. URL: <https://developer.mozilla.org/en-US/docs/Web/API/Element/querySelectorAll>.
33. MDN Web Docs. The loading attribute. Mozilla. 2024. URL: <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/img#loading>.
34. MDN Web Docs. Using data attributes. Mozilla. 2024. URL: https://developer.mozilla.org/en-US/docs/Learn/HTML/Howto/Use_data_attributes.

35. Nielsen J., Loranger H. Prioritizing Web Usability. Berkeley : New Riders, 2006. 406 p.
36. Norman D. The Design of Everyday Things: Revised and Expanded Edition. New York : Basic Books, 2013. 368 p.
37. Petrov D., Kovalenko L. AI-powered chatbots in the hospitality sector: systematic review. International Journal of Hospitality Management. 2023. Vol. 114. P. 103–112. DOI: 10.1016/j.ijhm.2023.103549.
38. Vaswani A. et al. Attention Is All You Need. Advances in Neural Information Processing Systems. 2017. Vol. 30. P. 5998–6008.
39. W3C. CSS Custom Properties for Cascading Variables Module Level 1. 2022. URL: <https://www.w3.org/TR/css-variables-1/>.
40. W3C. CSS Grid Layout Module Level 2. W3C Candidate Recommendation. 2023. URL: <https://www.w3.org/TR/css-grid-2/>.
41. W3C. CSS Transitions Module Level 1. W3C Working Draft. 2023. URL: <https://www.w3.org/TR/css-transitions-1/>.
42. W3C. HTML Accessibility API Mappings 1.0. 2021. URL: <https://www.w3.org/TR/html-aam-1.0/>.
43. W3C. Web Content Accessibility Guidelines (WCAG) 2.1. 2018. URL: <https://www.w3.org/TR/WCAG21/>.
44. WHATWG. HTML Living Standard. 2024. URL: <https://html.spec.whatwg.org/>.
- White J. et al. A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT. arXiv. 2023. arXiv:2302.11382.
- 45.

ДОДАТКИ

Додаток 1

Лістинг 3.2 — Формування menuCtx та системного промπτ

```
// Лістинг 3.2 – Побудова плоского масиву ALL та формування menuCtx

// Плоский масив усіх страв для семантичного пошуку ШІ
const ALL = [];
Object.entries(MENU).forEach(([cid, cat]) =>
  cat.items.forEach(i => ALL.push({ ...i, cid, cname: cat.name }))
);

// Рядок контексту (~2 KB) – вбудовується у системний промπτ
const menuCtx = ALL
  .map(i => `• ${i.name}: ${i.desc} (теги: ${i.tags}) – ${i.price} грн`)
  .join('\n');

// Системний промπτ із роллю та повним контекстом меню
const SYS = `Ти – ввічливий ШІ-бариста кав'ярні Blossom Café в Ірпені.
МЕНЮ:
${menuCtx}
ПРАВИЛА: відповідай ЛИШЕ українською, рекомендуй конкретні назви
із меню, 2-4 речення тепло, не вигадуй страв.`;
```

Додаток 2

Лістинг 3.3 — Функція askAI(): інтеграція з Gemini API

```
// Лістинг 3.3 – Функція askAI(): запит до Gemini API
async function askAI() {
  const inp = document.getElementById('ai-inp');
  const btn = document.getElementById('ai-btn');
  const txt = document.getElementById('ai-ans-txt');
  const hits = document.getElementById('ai-hits');
  const q = inp.value.trim();

  // Рівень 1: валідація вводу
  if (!q) { inp.focus(); return; }

  // Рівень 2: перевірка наявності API-ключа
  if (!GEMINI_KEY || GEMINI_KEY === 'ВАШ_КЛЮЧ_ТУТ') {
    txt.innerHTML = 'Вставте Gemini API ключ у data.js.';
    return;
  }

  // UI → стан завантаження
  btn.disabled = true; btn.textContent = '...';
```

```

txt.innerHTML = '<span class="dots">...</span>';
hits.innerHTML = '';

try {
  // Рівень 3: мережевий запит із async/await
  const r = await fetch(
    `https://generativelanguage.googleapis.com/v1beta/models/
    gemini-2.5-flash-lite:generateContent?key=${GEMINI_KEY}`,
    { method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({
        systemInstruction: { parts: [{ text: SYS }] },
        contents: [{ role: 'user', parts: [{ text: q }] }],
        generationConfig: {
          temperature: 0.72,
          maxOutputTokens: 350,
          topP: 0.9
        }
      })
    }
  );
  const d = await r.json();

  // Optional chaining – захист від нестандартної структури відповіді
  const a = d?.candidates?.[0]?.content?.parts?.[0]?.text
    || 'Вибачте, спробуйте ще раз.';

  txt.textContent = a; // textContent замість innerHTML – захист від XSS

  // Пошук згаданих страв та рендеринг карток
  const found = ALL.filter(i =>
    a.toLowerCase().includes(i.name.toLowerCase())
  );
  if (found.length) {
    hits.innerHTML = found.map(i => `
      <div class="ai-hit-card" onclick="openItem('${i.id}')">
        
        <div class="ai-hit-body">
          <div class="ai-hit-name">${i.name}</div>
          <div class="ai-hit-price">${i.price} грн</div>
        </div>
      </div>`).join('');
  }
} catch (e) {
  txt.textContent = 'Не вдалося з\'єднатися з сервером.';
}

btn.disabled = false; btn.textContent = 'Знайти'; // UI → звичайний стан

```

ДОДАТОК 3

Таблиця 3.3 — Результати тестування запитів до ШІ-асистента

№	Тип запиту	Запит	Результат	Оцінка	Час
1	Описовий	Щось тепле та заспокійливе	Трав'яний чай, Латте	Коректно	1.2 с
2	Бюджетний	Що найдешевше у меню?	Еспreso (45 грн)	Коректно	1.1 с
3	Дієтичний	Без кофеїну і без молока	Трав'яний чай, Лимонад	Коректно	1.4 с
4	Конкретний	Є капучіно?	Так, 65 грн + опис	Коректно	0.9 с
5	Дитячий	Що замовити дитині?	Лимонад, Ягідний чай	Коректно	1.3 с
6	Ситний	Хочу щось ситне на обід	Клаб-сендвіч, Омлет	Коректно	1.5 с
7	Вегетаріанський	Веганські варіанти	Матча-латте, Авокадо-тост	Коректно	1.2 с
8	Сезонний	Щось холодне і освіжаюче	Айс-латте, Лимонад	Коректно	1.1 с
9	Солодкий	Хочу десерт до кави	Круасан, Чізкейк	Коректно	1.6 с
10	Порційний	Щось легке, не важке	Салат Цезар, Авокадо-тост	Коректно	1.3 с
11	Ранковий	Хороший сніданок	Омлет з сиром, Капучіно	Коректно	1.4 с
12	Енергетичний	Треба бадьоритися, але без кофеїну	Матча-латте	Коректно	1.2 с
13	Груповий	Що взяти на двох із різними смаками?	Латте + Трав'яний чай	Коректно	1.7 с

№	Тип запиту	Запит	Результат	Оцінка	Час
14	Специфічний	Що зроблено з ефіопської кави?	Еспресо (Single Origin)	Коректно	1.8 с
15	Нетиповий	Рекомендуй за настроєм дощового дня	Загальна відповідь без назв	Частково	2.1 с