Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук

Бакалаврська кваліфікаційна робота на тему:

РОЗРОБКА WEB-РЕСУРСУ «SCHOOL KITCHEN»

Виконала: студентка 4 курсу, групи 3КН-21б
спеціальності 122 Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Соколова К.А.
(прізвище та ініціали)

Керівник: асистент каф. КН
Ваховська Л.М.
(прізвище та ініціали)
«04» червня 2025 р.

Рецензент: доцент каф. АІІТ
Барабан М.В.
(прізвище та ініціали)
«05» червня 2025 р.

Допущено до захисту
Зав. кафедри Яровий А.А.
«06» червня 2025 р.

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти перший (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 122 Комп'ютерні науки
Освітньо - професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
проф., д.т.н. Яровий А.А.
«05» травня 2025 р.

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ
Соколовій Катерині Андріївні

1. Тема роботи: Розробка WEB-ресурсу «School kitchen».
Керівник роботи: Ваховська Любов Михайлівна, асистент.
затверджені наказом вищого навчального закладу "20" березня 2025 року №97

2. Термін подання студентом роботи 30 травня 2025 р

3. Вихідні дані до роботи :
Мова програмування JavaScript, Середовище розробки Visual Studio Code;
Формати графічних файлів – JPEG,PNG;
Підтримка загального інтерфейсу баз даних ODBC.

4. Зміст текстової частини (перелік питань, які потрібно розробити):
Дослідити та проаналізувати методи та технології для розробки вебресурсів за вимогами до функціональності та дизайну; розробити загальну структурну схему функціонування вебресурсу; розробити алгоритми функціонування основних модулів системи вебресурсу; програмно реалізувати серверну та клієнтську часнити вебресурс «School kitchen»; виконати тестування та проаналізувати отримані результати.

5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень):
Діаграма класів вебдодатку
Загальна структурна схема функціонування вебресурсу
«Usecase» – діаграма вебдодатку
Діаграми класів та компонентів проекту
Загальний вигляд інтерфейсу користувача
Приклад роботи програми

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-3	Ваховська Л. М. каф. КН		

7. Дата видачі завдання: 05 травня 2025 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз предметної області та відомих реалізацій WEB-ресурсів оформлення замовлень у громадських закладах харчування	05.05.25	07.05.25	
2	Вибір архітектури WEB-ресурсу «School kitchen»	08.05.25	10.05.25	
3	Проектування WEB-ресурсу «School kitchen» із застосуванням UML-діаграм	11.05.25	15.05.25	
4	Розробка бази даних для WEB-ресурсу «School kitchen»	16.05.25	18.05.25	
5	Розробка схеми алгоритму роботи WEB-ресурсу «School kitchen»	19.05.25	21.05.25	
6	Обґрунтування вибору інструментів технічної реалізації WEB-ресурсу «School kitchen»	22.05.25	23.05.25	
7	Програмна реалізація WEB-ресурсу «School kitchen»	24.05.25	30.05.25	
8	Тестування розробленого WEB-ресурсу «School kitchen»	30.05.25	01.06.25	
9	Оформлення матеріалів до захисту БКР	02.06.25	04.06.25	

Студентка _____ (підпис) **Соколова К** (прізвище та ініці

Керівник проекту (роботи) _____ (підпис) **Ваховська** (прізвище та ініц

АНОТАЦІЯ

Бакалаврська кваліфікаційна робота складається з 80 сторінок формату А4, на яких є 19 рисунків, список використаних джерел містить 23 найменувань.

Робота присвячена розробці вебресурсу «School kitchen», який призначений для автоматизації процесу замовлення та обліку харчування у шкільних їдальнях. У роботі розглянуто етапи аналізу предметної області, проектування архітектури системи, вибору технологій та реалізації функціональних можливостей. Основний акцент зроблено на практичному створенні вебресурсу з використанням сучасних технологій веброзробки, таких як HTML, CSS, JavaScript та бібліотека React.js.

Ключові слова: вебресурс, автоматизація, шкільна їдальня, замовлення харчування, React.js, JavaScript, управління меню, адміністрування системи, тестування.

ABSTRACT

The bachelor's thesis consists of 80 A4-format pages, including 19 figures, and the list of references contains 23 sources.

The work is dedicated to the development of the "School kitchen" web resource, which is designed to automate the process of ordering and managing meals in school canteens. The study examines the stages of domain analysis, system architecture design, technology selection, and implementation of functional features. The main focus is on the practical development of the web resource using modern web development technologies such as HTML, CSS, JavaScript, and the React.js library.

Keywords: web resource, automation, school canteen, meal ordering, React.js, JavaScript, menu management, system administration, testing.

ЗМІСТ

ВСТУП

Актуальність досліджень. Розвиток інформаційних технологій значно впливає на різні сфери життя, зокрема на автоматизацію процесів у сфері громадського харчування. Одним із важливих напрямів є впровадження цифрових рішень у шкільних їдальнях, що дозволяє оптимізувати процес замовлення та обліку харчування, зменшити черги та покращити контроль за якістю послуг.

Виникає необхідність автоматизації процесу замовлення страв у шкільних їдальнях для підвищення ефективності роботи персоналу, мінімізації помилок при обробці замовлень та покращення користувацького досвіду.

Традиційні методи управління харчуванням у навчальних закладах часто є застарілими та вимагають значних витрат часу як для персоналу, так і для учнів та їхніх батьків. Впровадження вебресурсу дозволяє вирішити ці проблеми шляхом спрощення процесу вибору, замовлення та оплати страв.

Сучасний стан розвитку технічних задач, що їх належить розв’язати в бакалаврській кваліфікаційній роботі, визначається активним впровадженням цифрових сервісів у повсякденне життя та високими очікуваннями користувачів щодо зручності, доступності та інтерактивності вебрішень. В умовах навчальних закладів це означає необхідність створення простих, надійних і ефективних інструментів, які легко впроваджуються, мають зрозумілий інтерфейс і не потребують значних ресурсів для підтримки.

Метою дослідження є розширення функціональності користувацько-орієнтованого вебресурсу у вигляді POS-системи для автоматизації замовлення та обліку харчування у шкільних їдальнях, що забезпечить зручний доступ до меню, ефективне адміністрування замовлень та покращений контроль процесу харчування [1].

Об’єкт дослідження – процес автоматизації управління замовленнями харчування у шкільних їдальнях.

Предмет дослідження – програмна реалізація вебресурсу для автоматизації замовлення та обліку харчування, зокрема використання HTML, CSS, JavaScript та бібліотеки React.js.

Практична цінність полягає у тому, що розроблені алгоритми та методи можуть бути використані при створенні подібних WEB-ресурсів.

Для досягнення цієї мети необхідно вирішити такі **задачі дослідження:**

1. Провести аналіз предметної області та існуючих аналогів систем автоматизації замовлення харчування.
2. Розробити архітектуру вебресурсу та визначити основні компоненти системи.
3. Спроектувати структуру вебресурсу та логіку взаємодії між компонентами.
4. Розробити базу даних для вебресурсу, визначити сутності та зв’язки між ними.
5. Програмно реалізувати функціональність вебресурсу, включаючи модулі замовлення страв, адміністрування меню та управління замовленнями.
6. Провести тестування розробленого вебресурсу для оцінки його працездатності, стабільності та відповідності вимогам користувачів.

Апробація роботи. Результати досліджень було апробовано на LIV Всеукраїнській науково-технічній конференції факультету інтелектуальних інформаційних технологій та автоматизації Вінницького національного технічного університету (НТКП ВНТУ - 2025) м. Вінниці у 2025 р.

Публікації. За основними результатами досліджень бакалаврської кваліфікаційної роботи було опубліковано тези доповіді на науково-технічній конференції [1].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ВІДОМИХ РЕАЛІЗАЦІЙ POS-СИСТЕМ

1.1 Обґрунтування доцільності розробки вебресурсу «School kitchen» у вигляді POS-системи

Сучасні цифрові технології відіграють важливу роль у процесах автоматизації та оптимізації бізнес-процесів у різних сферах діяльності. Однією з таких сфер є громадське харчування, де ефективне управління ресурсами та сервісами є ключовим завданням для забезпечення якісного обслуговування споживачів. Впровадження автоматизованих систем замовлення та обліку харчування дозволяє значно підвищити ефективність роботи їдалень та мінімізувати людський фактор у процесі обробки замовлень [1].

На сьогодні більшість шкільних їдалень в Україні працюють за застарілими методами обліку та управління замовленнями. Основні проблеми, що виникають при використанні традиційних підходів, включають:

- часові затримки. У години найбільшої активності, наприклад під час перерв, учні змушені витрачати значний час у чергах, що призводить до скорочення часу на прийом їжі та створює дискомфорт;
- помилки при обробці замовлень. Через ручний спосіб реєстрації замовлень можуть виникати неточності, що призводить до плутанини в замовленнях та незадоволення з боку учнів;
- відсутність персоналізації. Шкільні їдальні часто не мають механізмів для обліку дієтичних уподобань учнів, алергічних реакцій чи спеціальних потреб, що може негативно впливати на якість харчування;
- необхідність урахування фінансових аспектів. Відсутність централізованого контролю витрат на харчування ускладнює управління бюджетом сімей учнів, особливо якщо йдеться про пільгові категорії або передоплачені обіди.

З огляду на ці проблеми, необхідність цифровізації процесу замовлення та обліку харчування є очевидною. Впровадження сучасного вебресурсу дозволяє розв’язати більшість із зазначених питань та створити зручну платформу для взаємодії учнів, батьків, персоналу їдальні та адміністрації навчального закладу.

Одним із найефективніших інструментів управління продажами, обліком товарів та обслуговуванням є POS-системи (Point of Sale – точка продажу).

POS-система – це програмно-апаратний комплекс, який використовується для проведення фінансових операцій, обліку продажів, автоматизації процесу оформлення замовлень, керування запасами та збору аналітичних даних. Такі системи широко застосовуються у сфері роздрібної торгівлі, громадського харчування, готельного бізнесу та сферах, пов’язаних із наданням послуг.

POS-система – це ключовий компонент сучасного роздрібного та громадського харчування. Для ефективного впровадження POS-системи в шкільну їдальню, необхідно вивчити специфіку даної предметної області.

POS-системи можна поділити на кілька основних типів:

1. Класичні POS-системи – включають касові апарати, підключені до спеціалізованого програмного забезпечення, що дозволяє здійснювати продаж товарів і послуг.

2. Хмарні POS-системи – працюють на основі вебтехнологій, що дозволяє зберігати дані у віддалених серверах та отримувати доступ до них з будь-якого пристрою.

3. Мобільні POS-системи – використовуються на планшетах або смартфонах, що дає можливість працювати з продажами у будь-якому місці.

POS-системи для громадського харчування – спеціалізовані рішення, які автоматизують процес приймання замовлень, контролю залишків, ведення аналітики та оптимізації роботи персоналу в ресторанах, кафе та їдальнях.

Основними складовими POS-систем є: програмне забезпечення – платформи для ведення обліку продажів, управління замовленнями та аналізу фінансових показників, обладнання – касові апарати, платіжні термінали, сенсорні екрани, принтери чеків тощо, база даних – сховище інформації про

товари, клієнтів, продажі та аналітику, інтерфейси взаємодії – мобільні та веб-додатки для зручного доступу до POS-функціоналу [1].

У контексті шкільної їдальні, вебдодаток у вигляді POS-системи повинен враховувати наступні ключові аспекти:

- меню. Шкільні їдальні мають регулярно оновлюване меню, часто на тиждень або місяць вперед. Веб-додаток повинен мати зручну систему для оновлення меню та відображення його для користувачі;
- замовлення та підтвердження. Система повинна дозволяти користувачам легко замовляти їжу з меню, вибираючи порції та вказуючи час для видачі. Також система повинна надсилати підтвердження замовлення для зручності користувачів;
- час видачі. У шкільних їдальнях часто є певний час для обіду, і цей час повинен бути врахований при замовленні та видачі замовлень;
- безпека платежів. Оскільки система включає фінансові транзакції, важливо, щоб вона була безпечною та надійною;
- інтеграція з іншими системами. Вебдодаток може бути інтегрований з іншими шкільними системами, такими як системи управління учнями, бухгалтерські системи тощо, що дозволить автоматизувати процеси та полегшити управління;
- користувацький досвід. Важливо створити зручний та інтуїтивно зрозумілий інтерфейс, який був би доступний для різних вікових категорій користувачів;
- масштабованість. Шкільні їдальні можуть відрізнятися за розміром і потребами, тому система повинна бути гнучкою і масштабованою, щоб задовольнити різні потреби.

Оглядаючи ці аспекти, можна розробити веб-додаток «School kitchen», що відповідає специфічним вимогам шкільних їдальнь та забезпечує ефективне управління процесом замовлення та видачі їжі.

POS-система для шкільної їдальні має враховувати особливості середовища школи, а також вимоги учнів, персоналу та батьків. Деякі основні особливості, які слід врахувати, включають:

- структуровані години обіду. У школах є визначений графік для обідніх перерв, і це вимагає точного планування та координації для забезпечення ефективного обслуговування учнів;
- індивідуальні дієтичні потреби. У деяких учнів можуть бути спеціальні дієтичні вимоги або алергії на продукти, і система повинна мати можливість враховувати такі вимоги при замовленні;
- контроль батьків. Батьки мають можливість контролювати замовлення своїх дітей, а також їх харчовий баланс;

Для вдалих POS-систем важливо розуміти, чого саме хочуть користувачі. До основних груп користувачів POS-системи шкільної їдальні відносяться учні, персонал шкільної їдальні та батьки учнів.

Учні хочуть замовляти їжу швидко та легко, а також отримувати її у вказаний час. Також для них важливо мати змогу переглянути меню та скласти замовлення завчасно. Для персоналу важливо мати можливість легко оновлювати меню, враховувати спеціальні дієтичні вимоги та ефективно планувати розподіл порцій.

Батьки хочуть мати контроль над тим, що їх діти їдять в школі. Вони також можуть хотіти здійснювати поповнення рахунків для оплати обідів своїх дітей.

Розробка вебресурсу «School kitchen» у вигляді POS-системи є доцільним рішенням для автоматизації процесів управління харчуванням у шкільних їдальнях. Така система дозволяє покращити обслуговування, підвищити контроль за замовленнями та забезпечити зручний доступ до інформації для всіх учасників процесу.

Використання вебтехнологій у розробці системи забезпечить високу продуктивність, зручність у використанні та можливість подальшого розширення функціональності. Таким чином, впровадження «School kitchen» як

POS-системи сприятиме покращенню організації шкільного харчування та оптимізації витрат навчальних закладів.

1.2 Огляд відомих реалізації POS-системи

POS-системи (Point of Sale) активно використовуються в різних сферах бізнесу, включаючи ресторанний бізнес та шкільні їдальні. Деякі з найвідоміших реалізацій POS-систем у світі включають Square POS, Shopify, Clover, та Lightspeed.

Square POS визнаний своєю простотою використання та доступністю. Ця система пропонує безкоштовний мобільний POS-додаток, що дозволяє приймати платежі будь-де. Square також пропонує аналітичні інструменти, інтеграцію з онлайн-платформами та можливість управління інвентарем.

Shopify є ще одним популярним рішенням, особливо для онлайн-бізнесу. Shopify POS дозволяє продавцям приймати платежі через мобільні пристрої та синхронізувати продажі онлайн та офлайн.

Clover це гнучка POS-система, яка пропонує множину функцій, таких як управління інвентарем, стеження за тим, як ваш бізнес працює, та можливість підключати зовнішні додатки.

Lightspeed пропонує рішення, спеціально розроблені для ресторанів, роздрібних магазинів та електронної комерції. Їх система POS включає інтеграцію з управлінням інвентарем, CRM та інші корисні інструменти.

Ці та інші POS-системи є відмінними прикладами того, як можна ефективно інтегрувати технологію в процес замовлення та видачі їжі. Однак, у шкільних їдальнях потребуються особливі функції, такі як можливість замовляти їжу завчасно, вказувати час видачі порцій, та інше. На основі цих відомих реалізацій, було розроблено власний додаток «School kitchen», який враховує ці специфічні потреби.

Розглядаючи додаткові відомі POS-системи, які пов’язані зі сферою громадського харчування, можна виділити такі рішення як Vend, TouchBistro та Toast.

Vend – це хмарна POS-система, зорієнтована на роздрібну торгівлю, але її функціонал також придатний для сектора громадського харчування. Vend надає можливість працювати зі складом, управляти працівниками, слідкувати за продажами та збирати деталізовану інформацію про клієнтів.

Vend розроблена для роздрібної торгівлі, яка забезпечує ефективне управління продажами, запасами та взаємодією з клієнтами. Заснована у 2010 році в Окленді, Нова Зеландія, компанія швидко здобула популярність серед роздрібних продавців по всьому світу.

Головну сторінку цього вебресурсу можна побачити на рисунку 1.1 [2].

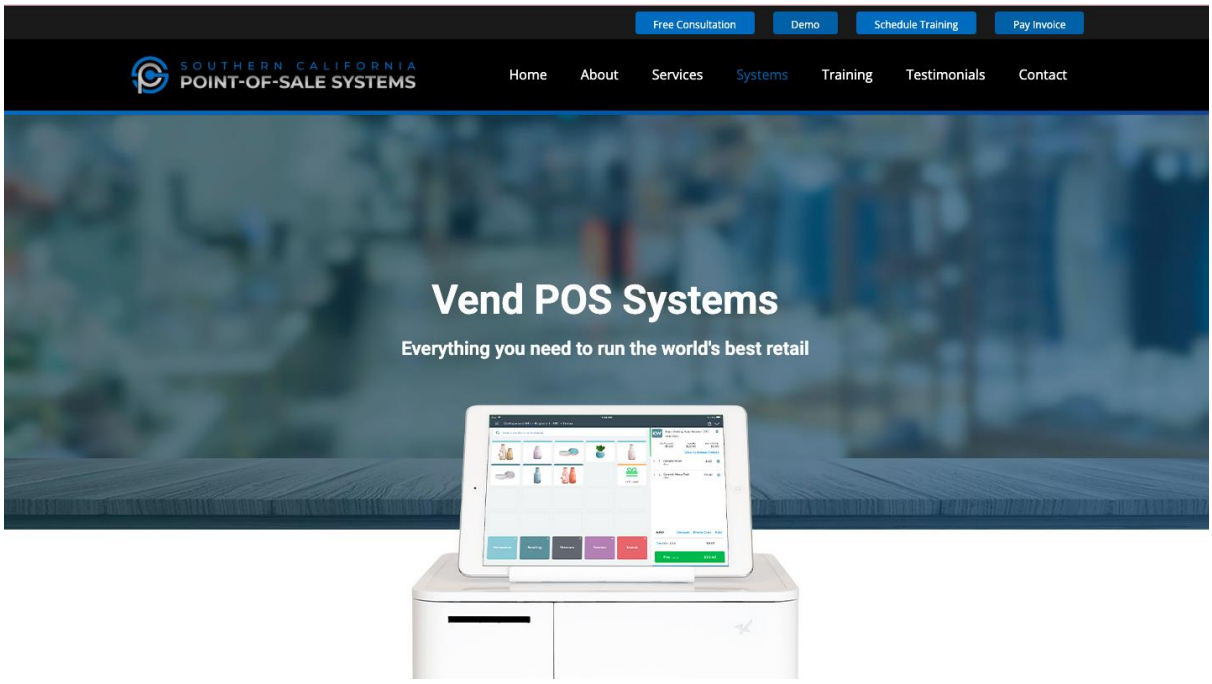

Рисунок 1.1 – Головна сторінка вебресурсу Vend

Основні характеристики Vend:

- хмарна архітектура: Vend працює на основі хмарних технологій, що дозволяє отримувати доступ до системи з будь-якого пристрою з інтернет-з’єднанням. Це забезпечує гнучкість та мобільність у роботі;

- підтримка офлайн-режиму: у разі втрати інтернет-з’єднання Vend продовжує працювати в офлайн-режимі, синхронізуючи дані після відновлення підключення. Це гарантує безперебійність роботи навіть за умов нестабільного інтернету;
- інтеграція з обладнанням: Система підтримує роботу на iPad, Mac або PC, а також інтегрується з різноманітним POS-обладнанням, таким як касові апарати, сканери штрих-кодів та принтери чеків;
- управління запасами: Vend надає інструменти для відстеження запасів у режимі реального часу, управління постачаннями та автоматичного поповнення товарів, що сприяє оптимізації складських процесів;
- інтеграція з платіжними системами: Vend підтримує інтеграцію з різними платіжними шлюзами, що забезпечує швидкі та безпечні транзакції.

У 2021 році Vend була придбана компанією Lightspeed, що дозволило розширити функціональні можливості та інтеграцію з іншими продуктами екосистеми Lightspeed.

Переваги Vend:

1. Хмарна архітектура – завдяки зберіганню даних у хмарі, користувачі можуть отримати доступ до системи з будь-якого пристрою через інтернет. Це робить систему зручною для роботи з різних точок продажу.
2. Підтримка офлайн-режиму – навіть у разі втрати інтернет-з'єднання POS-система продовжує працювати, а після відновлення підключення всі дані синхронізуються автоматично.
3. Широка інтеграція – система підтримує підключення до Shopify, Xero, PayPal та інших платформ, що робить її гнучкою для бізнесу різного масштабу.
4. Гнучкість та масштабованість – підходить як для малого, так і для середнього бізнесу, забезпечуючи легке розширення мережі магазинів.
5. Зручне управління запасами – система дозволяє автоматично відстежувати залишки товарів, створювати замовлення та прогнозувати потреби складу.

6. Безпека даних – хмарне зберігання даних гарантує захист від втрати інформації та збоїв у роботі локальних пристроїв.

Недоліки Vend:

1. Залежність від інтернету – хоча є офлайн-режим, для доступу до всіх функцій необхідне стабільне з'єднання.

2. Висока вартість підписки – для малого бізнесу вартість підключення та щомісячного обслуговування може бути значною.

3. Неоптимізована для ресторанів – система більше підходить для роздрібної торгівлі, а не для закладів громадського харчування.

TouchBistro – це POS-система, спеціально створена для ресторанного бізнесу. Вона пропонує повний набір функцій, включаючи управління меню, замовленнями, запасами, працівниками, лояльністю клієнтів та аналітикою. Заснована у 2010 році в Торонто, Канада, компанія швидко стала лідером у сфері ресторанних технологій. Має дуже приємний дизайн головного сайту, який зображено на рисунку 1.2 [3].

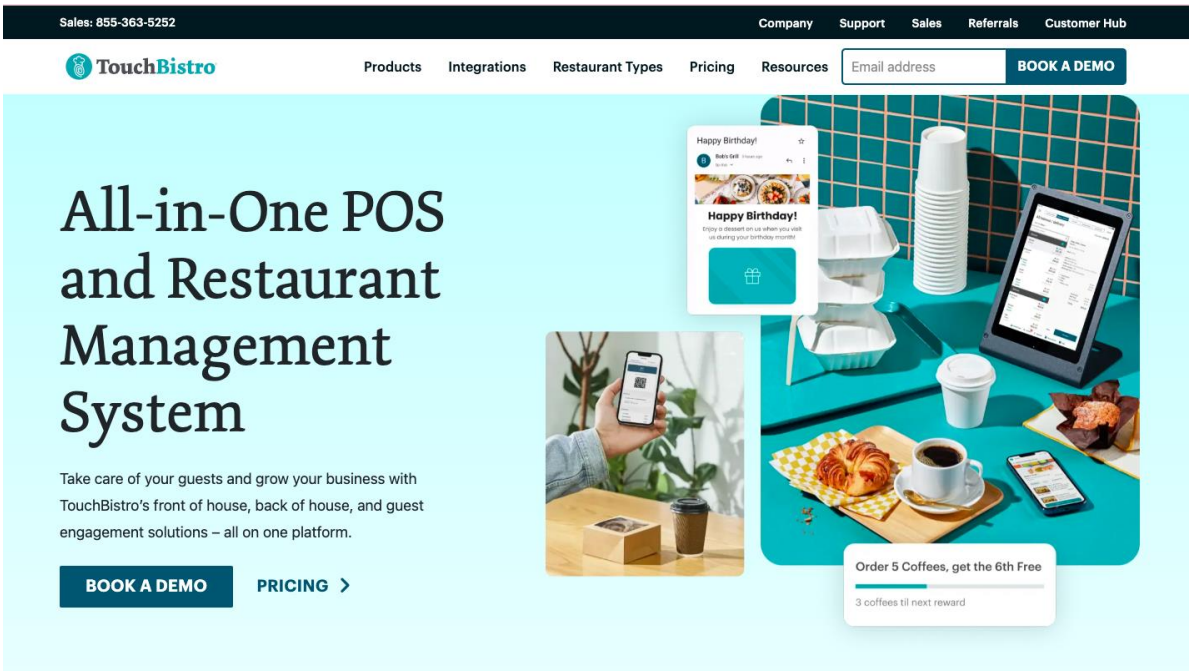

Рисунок 1.2 – Головна сторінка вебресурсу TouchBistro

Основні характеристики TouchBistro:

- орієнтація на ресторанний бізнес: система розроблена з урахуванням специфіки роботи ресторанів, кафе та барів, забезпечуючи функціональність, необхідну для ефективного обслуговування клієнтів;
- підтримка офлайн-режиму. TouchBistro продовжує працювати навіть при відсутності інтернет-з’єднання, забезпечуючи безперебійність обслуговування клієнтів;
- інтуїтивно зрозумілий інтерфейс: система має зручний інтерфейс, що спрощує навчання персоналу та підвищує ефективність роботи;
- управління меню та замовленнями: TouchBistro дозволяє легко оновлювати меню, відстежувати замовлення та контролювати процес приготування страв, що сприяє підвищенню якості обслуговування;
- інтеграція з платіжними системами: система підтримує інтеграцію з різними платіжними шлюзами, що забезпечує швидкі та безпечні транзакції;
- аналітика та звітність: TouchBistro надає детальні звіти про продажі, продуктивність персоналу та вподобання клієнтів, що допомагає приймати обґрунтовані бізнес-рішення.

Завдяки своїй спеціалізації та функціональності, TouchBistro є популярним вибором серед закладів громадського харчування, які прагнуть оптимізувати свої операційні процеси.

Переваги TouchBistro:

1. Оптимізація для ресторанів – система спеціально розроблена для управління замовленнями, обслуговуванням клієнтів та контролем персоналу у закладах громадського харчування.
2. Локальне зберігання даних – система працює без потреби у постійному інтернет-з'єднанні, що забезпечує стабільність роботи.
3. Зручне управління столами – інтерактивна карта залу дозволяє персоналу ефективно керувати розміщенням гостей та обробкою замовлень.
4. Гнучке управління меню – легке оновлення та зміна меню, включаючи акції та сезонні пропозиції.

5. Мобільність – можливість використовувати систему на iPad дозволяє персоналу швидко приймати замовлення безпосередньо за столами.

6. Аналітика та звітність – система збирає та аналізує дані про замовлення, що допомагає оцінювати ефективність бізнесу.

Недоліки TouchBistro:

1. Обмеження щодо обладнання – система працює лише на iPad, що може бути незручним для деяких користувачів.

2. Висока вартість – придбання обладнання та підписка можуть бути дорогими для малих закладів.

3. Менше інтеграцій, ніж у хмарних систем – підтримка зовнішніх додатків є обмеженою.

4. Необхідність регулярного оновлення – оновлення через App Store є обов’язковим для стабільної роботи.

Toast – це ще одна хмарна POS-система, спеціалізована на громадському харчуванні. Вона надає широкий спектр функцій, таких як онлайн-замовлення, управління інвентарем, репортинг та аналітика.

Система також пропонує рішення для податкового обліку та бухгалтерії, що може бути корисним для шкільних їдальнь. Заснована у 2012 році в Бостоні, США, компанія швидко здобула популярність серед ресторанів різного типу та масштабу.

Toast поєднує функціонал для управління замовленнями, оплатами, персоналом, інвентарем та аналітикою в єдиній зручній платформі. Toast підтримує як стаціонарні, так і мобільні термінали, дозволяє налаштовувати програми лояльності, приймати онлайн-замовлення, інтегрується з кухонними дисплеями та сторонніми сервісами доставки.

Система працює на базі Android і здатна функціонувати навіть офлайн. Хоча Toast здебільшого орієнтована на ринок США, її функціональність робить її однією з найпотужніших POS-систем для ресторанного бізнесу.

Вигляд головної сторінки вебресурсу можна побачити на рисунку 1.3 [4].

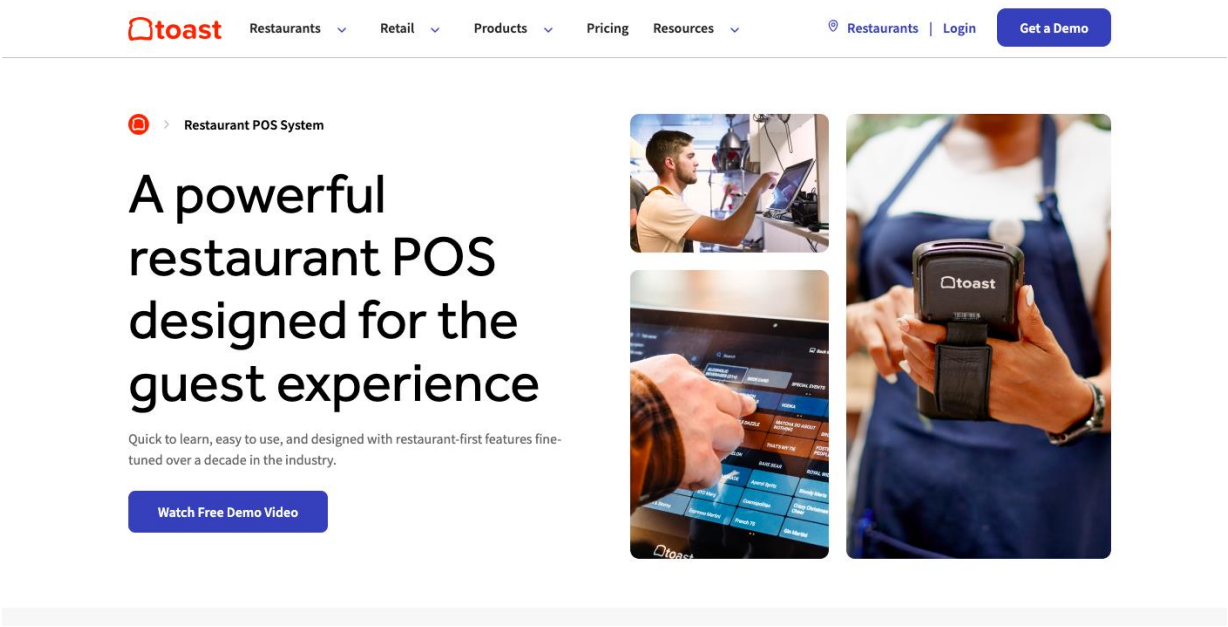

Рисунок 1.3 – Головна сторінка вебресурсу Toast

Основні характеристики Toast:

- хмарна архітектура: Toast працює на базі хмарних технологій, що дозволяє отримувати доступ до системи з будь-якого пристрою з інтернет-з'єднанням. Це забезпечує гнучкість та мобільність у роботі;
- управління замовленнями: система дозволяє офіціантам приймати замовлення, надсилати їх на кухню та обробляти платежі безпосередньо за столом за допомогою портативних планшетів Toast;
- управління меню: Toast надає можливість легко змінювати та оновлювати меню, включаючи створення нових позицій та відстеження популярності страв;
- управління запасами: система забезпечує відстеження запасів у реальному часі, що допомагає уникнути нестачі продуктів та оптимізувати процеси постачання;
- аналітика та звітність: Toast генерує детальні звіти про продажі, продуктивність персоналу та інші ключові показники, що сприяє прийняттю обґрунтованих бізнес-рішень.

Переваги Toast:

1. Повноцінна система для громадського харчування – підтримує ресторани, кафе, бари та пекарні.
2. Портативність – використання планшетів дозволяє офіціантам швидко приймати замовлення та передавати їх на кухню.
3. Розширене управління запасами – автоматизований контроль наявності інгредієнтів та відстеження використання продуктів.
4. Потужна аналітика – система надає звіти про продуктивність персоналу, популярність страв та ефективність продажів.
5. Інтеграція з онлайн-замовленнями – можливість обробки онлайн-замовлень та доставки.
6. Гнучкість у редагуванні меню – швидке внесення змін та оновлення інформації.
7. Програми лояльності – підтримка бонусних програм та акцій для клієнтів.

Недоліки Toast:

1. Залежність від інтернету – для повноцінної роботи потрібне стабільне підключення.
2. Висока вартість – обслуговування та підключення можуть бути дорогими для невеликих закладів.
3. Обмежена підтримка поза США – система найбільш ефективна у США, що може бути недоліком для міжнародних ринків.

Залежно від потреб бізнесу, кожна з розглянутих POS-систем має свої переваги та обмеження. Загальну картину можна побачити розглянувши таблицю 1.1.

Таблиця 1.1 – Переваги та недоліки розглянутих POS-систем

POS-система	**Переваги**	**Недоліки**
Vend	Хмарна архітектура, підтримка офлайн-режиму, інтеграція з платформами, управління запасами	Залежність від інтернету, висока вартість підписки, не оптимізована для ресторанів
TouchBistro	Оптимізована для ресторанів, зручне управління столами, мобільність, аналітика	Працює лише на iPad, висока вартість, обмежена кількість інтеграцій
Toast	Підходить для ресторанів, онлайн-замовлення, аналітика, управління запасами	Залежність від інтернету, висока вартість, обмежена підтримка поза США

Vend є найкращим варіантом для роздрібних магазинів, оскільки пропонує зручне управління запасами та інтеграцію з популярними платформами.

TouchBistro – чудовий вибір для ресторанів, які хочуть ефективно керувати столами, меню та персоналом, але система працює лише на iPad.

Toast ідеально підходить для закладів громадського харчування, що шукають потужну платформу з розширеними можливостями аналітики, управління запасами та програмами лояльності.

Для розробки вебресурсу «School kitchen» у вигляді POS-системи доцільно врахувати функціональні можливості TouchBistro та Toast, оскільки вони оптимізовані для управління замовленнями, інтеграції з платіжними системами та роботи у сфері громадського харчування.

Ці та інші POS-системи показують, як технології можуть допомогти управлінню процесами замовлення та обслуговування в секторі громадського харчування. Однак, необхідно зосередитися на створенні спеціалізованого рішення для шкільних їдальнь, яке враховує їхні особливості та потреби.

Є ряд ключових особливостей, які спільні для багатьох POS-систем:

- управління інвентарем та меню. POS-системи дозволяють операторам легко оновлювати та керувати меню, а також слідкувати за запасами інгредієнтів або товарів;

- обробка транзакцій. POS-системи виконують платіжні транзакції, включаючи кредитні та дебетові картки, мобільні платежі та інші форми електронної оплати;
- звітність та аналітика. Більшість POS-систем надає детальну звітність та аналітику продажів, що допомагає бізнесу зрозуміти тренди та використовувати цю інформацію для прийняття рішень;
- управління персоналом. POS-системи часто мають функції для керування графіками працівників, відслідковування продуктивності та ведення обліку часу роботи;
- інтеграція з іншими системам. Багато POS-систем можуть інтегруватися з іншими системами, такими як системи управління відносинами з клієнтами (CRM), системи управління товарами (Inventory Management Systems), системи бухгалтерського обліку та інші;
- мобільність. Багато сучасних POS-систем можуть працювати на мобільних пристроях, таких як планшети або смартфони, що дозволяє обслуговувати клієнтів поза традиційною касою;
- безпека. POS-системи повинні відповідати стандартам безпеки для захисту платіжних даних користувачів.

Ці особливості формують основу POS-системи, але конкретна реалізація може варіюватися в залежності від специфічних потреб бізнесу або простору, де система буде використовуватися.

1.3 Формулювання вимог та постановка задачі

Вебресурс «School kitchen» призначений для автоматизації процесу замовлення та обліку харчування у шкільних їдальнях. Основна мета його розробки – спрощення взаємодії між учнями, батьками та адміністрацією їдальні шляхом впровадження електронної системи замовлень.

Для коректного функціонування система повинна підтримувати реєстрацію користувачів, що передбачає різні рівні доступу, зокрема для учнів,

батьків та адміністрації. Меню повинно формуватися адміністраторами, які матимуть змогу додавати, змінювати та видаляти страви. Користувачі повинні мати можливість обирати страви з доступного меню, оформлювати замовлення та вказувати час їх отримання.

Важливою складовою є збереження історії замовлень, щоб користувачі та адміністратори могли переглядати інформацію про здійснені замовлення. Система повинна підтримувати інтеграцію з платіжними сервісами або мати можливість передплачених розрахунків. Крім того, необхідно забезпечити можливість формування звітності та аналізу популярності страв.

До нефункціональних вимог належить забезпечення зручного користувацького інтерфейсу, що буде адаптивним і простим у використанні. Безпека даних повинна гарантувати захист персональної інформації та контроль доступу. Вебресурс повинен працювати швидко, забезпечуючи мінімальні затримки під час обробки замовлень.

Важливою характеристикою є масштабованість, що дозволить розширювати функціональність та підтримувати декілька навчальних закладів одночасно. Доступність системи передбачає її кросбраузерність та підтримку мобільних пристроїв.

Для реалізації вебресурсу необхідно вирішити такі основні задачі:

1. Розробити архітектуру системи, визначивши взаємодію між користувачами, базою даних та серверною частиною.
2. Створити інтерфейс користувача, реалізувавши сторінки для перегляду меню, оформлення замовлень та адміністративного управління.
3. Реалізувати серверну частину для обробки запитів, роботи з базою даних та забезпечення безпеки.
4. Розробити модуль аналітики для збору та обробки даних щодо замовлень.
5. Провести тестування вебресурсу для перевірки функціональності та виправлення можливих помилок.

Впровадження «School kitchen» дозволить значно спростити процес замовлення та обліку харчування в шкільних їдальнях, покращити контроль за якістю обслуговування та зменшити ризики помилок при реєстрації замовлень.

1.4 Висновок до розділу 1

Аналіз предметної області показав, що сучасні шкільні їдальні стикаються з проблемами неефективного управління замовленнями, великими чергами та незручностями для учнів і персоналу. Впровадження вебресурсу «School kitchen» дозволить розв’язати ці питання, забезпечуючи автоматизацію основних процесів. На основі огляду існуючих POS-систем було визначено основні функціональні можливості, які необхідно реалізувати у проєкті. Це включає систему реєстрації користувачів, управління меню, формування замовлень та їх адміністрування.

Формулювання вимог до вебресурсу дозволило окреслити ключові функціональні та нефункціональні характеристики системи. Було визначено необхідність інтеграції засобів безпечної автентифікації користувачів, підтримки платіжних сервісів та аналітичних інструментів для збору статистики. Також було розглянуто сучасні технології веброзробки, які забезпечать високу швидкість роботи та зручний користувацький інтерфейс. Використання React.js дозволяє створити гнучкий та масштабований вебресурс, а Firebase забезпечить ефективне збереження та обробку даних.

Таким чином, виконаний аналіз та формулювання вимог заклали основу для подальшої розробки вебресурсу «School kitchen». Очікується, що цей продукт значно спростить процес замовлення та обліку харчування, покращить контроль за якістю обслуговування та підвищить ефективність роботи персоналу шкільних їдалень. Подальші етапи розробки включатимуть детальне проєктування, реалізацію функціональності та тестування системи для забезпечення її надійності та відповідності визначеним вимогам.

2 ПРОЕКТУВАННЯ СТУРКТУРИ ТА КОМПОНЕНТІВ ВЕБРЕСУРСУ «SCHOOL KITCHEN»

2.1 Обґрунтування вибору архітектури для вебресурсу

Розробка вебресурсу «School kitchen» вимагає вибору відповідної архітектури, яка забезпечить високу продуктивність, масштабованість та безпеку. Вебресурси можуть використовувати різні архітектурні підходи, серед яких можна виокремити три основні: монолітну архітектуру, мікросервісну архітектуру та архітектуру клієнт-сервер (зокрема, SPA — Single Page Application) [5-6].

Монолітна архітектура передбачає створення єдиної програми, де клієнтська і серверна частини тісно взаємопов’язані. Основними перевагами такого підходу є простота розгортання, цілісність коду та легкість тестування.

Однак, при розширенні функціональності монолітна архітектура може ускладнити підтримку, масштабування та оновлення окремих компонентів. Схематичне зображення цієї архітектури на рисунку 2.1.

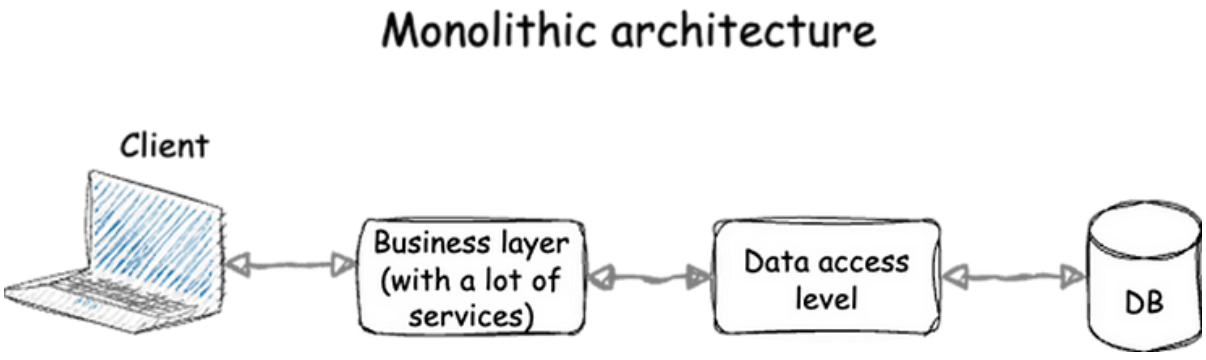

Рисунок 2.1 – Схематичне зображення монолітної архітектури

Мікросервісна архітектура, навпаки, дозволяє розділити систему на незалежні сервіси, що взаємодіють між собою через API. Це покращує масштабованість і надійність, оскільки кожен сервіс можна оновлювати та

змінювати незалежно від інших. Також це надає простоту в підтримці великих команд розробників, кожна з яких може працювати над своїм мікросервісом.

Мікросервіси можуть розгортатися, масштабуватися та оновлюватися окремо, що забезпечує високу гнучкість, стійкість до збоїв і швидкість впровадження змін. Така архітектура особливо ефективна для великих і складних веб-додатків, дозволяє різним командам працювати автономно над окремими компонентами системи, використовуючи різні мови програмування та технології.

Основним недоліком мікросервісної архітектури є складність управління великою кількістю сервісів та потреба в ефективній організації комунікації між ними. Потрібно добре продумувати тестування і безпеку, оскільки система складається з багатьох частин.

Схему можна побачити на рисунку 2.2.

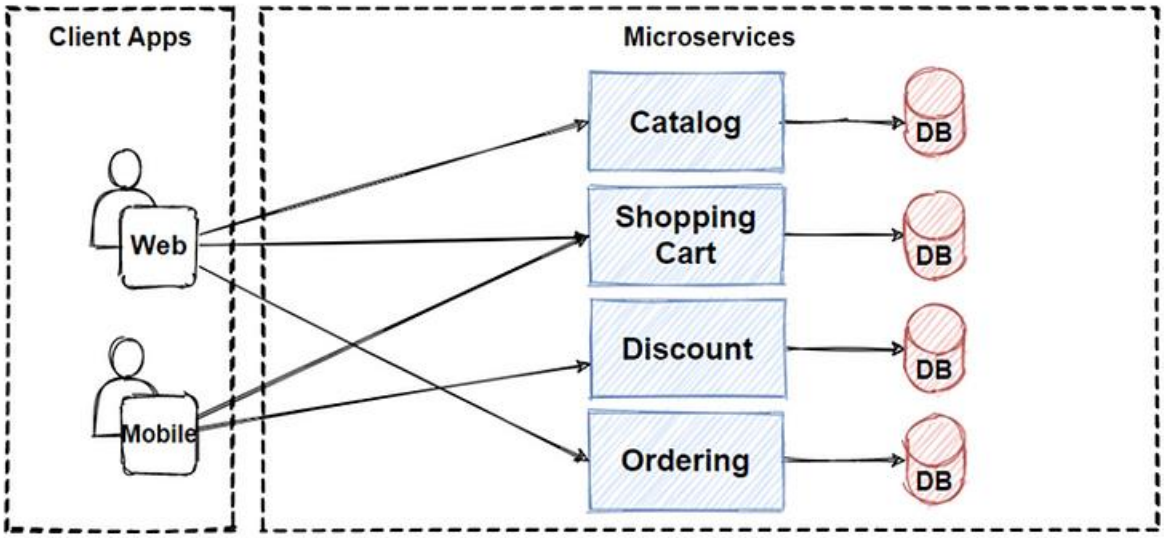

Рисунок 2.2 – Схематичне зображення мікросервісної архітектури

Архітектура клієнт-сервер (SPA) забезпечує розділення клієнтської та серверної логіки, дозволяючи фронтенду взаємодіяти із сервером через API. SPA-додатки використовують єдину сторінку, яка динамічно оновлюється при взаємодії користувача.

Схема зображена на рисунку 2.3.

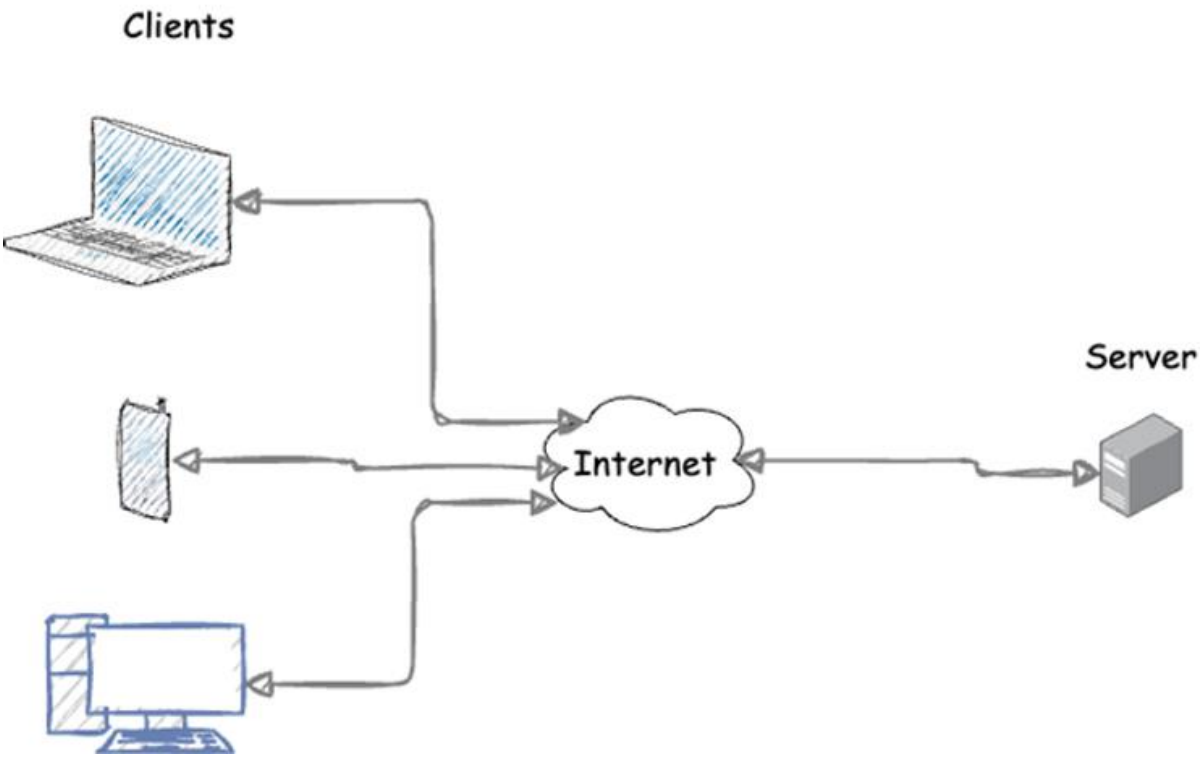

Рисунок 2.3 – Схематичне зображення архітектури клієнт-сервер (SPA)

Основні переваги цього підходу включають швидку взаємодію без перезавантаження сторінок, лише оновлення і змінювання серверу, не зачіпаючи клієнтської частини, зниження навантаження на сервер та покращений користувацький досвід.

Недоліком може бути початкове завантаження всієї програми, що може збільшити час першого відкриття сторінки. Клієнт і сервер повинні постійно взаємодіяти: якщо сервер виходить з ладу, клієнт не може працювати.

Аналізуючи вищезазначені варіанти, для вебресурсу «School kitchen» було обрано архітектуру Single Page Application (SPA) у поєднанні з «Backend as a Service» (BaaS) на основі Firebase. Це рішення дозволяє ефективно реалізувати динамічний користувацький інтерфейс, швидке оновлення контенту та зручне управління даними.

React.js використовується для фронтенду, що дозволяє розробити гнучку та масштабовану компонентну систему. Firebase надає хмарне зберігання даних,

механізми автентифікації та обробки запитів, що спрощує адміністрування серверної частини.

Серверна логіка реалізується через Cloud Functions у Firebase, що дозволяє мінімізувати витрати на підтримку серверного середовища та легко масштабувати систему залежно від навантаження. Такий підхід забезпечує високу продуктивність, зручність використання, безпеку та можливість подальшого розширення функціональності [6].

Впровадження SPA та використання Firebase як бекенд-сервісу дозволить створити ефективний, інтерактивний та надійний вебресурс для автоматизації процесу харчування у школах.

2.2 Проектування вебресурсу із застосуванням UML-діаграм

Для ефективного проектування вебресурсу «School kitchen» використовується мова UML (Unified Modeling Language), яка дозволяє моделювати структуру системи, її поведінку та взаємодію між компонентами. UML-діаграми допомагають формалізувати вимоги, візуалізувати основні модулі вебресурсу та їхню взаємодію, що сприяє кращому розумінню системи під час розробки.

Універсальна мова моделювання (UML) є одним із ключових інструментів, який допомагає в процесі проектування програмного забезпечення, зокрема вебдодатку, застосовувати об'єктно-орієнтований підхід. UML надає зручний і стандартизований набір засобів та нотацій для моделювання системи, дозволяючи аналізувати, проектувати та документувати різноманітні аспекти системи.

Універсальна мова моделювання (UML) є стандартом для визначення, відображення та документування об'єктно-орієнтованих систем. Вона надає графічні нотації для створення діаграм, які представляють структуру, поведінку та взаємодії компонентів системи. UML дозволяє команді розробників зрозуміти систему, спілкуватися та спільно працювати над її розробкою [7].

У процесі проектування вебдодатку UML надає різні можливості для моделювання системи, дозволяючи розробникам описати структуру, поведінку та взаємодії компонентів. Нижче розглянуті деякі з найбільш використовуваних можливостей UML в контексті проектування вебдодатку:

1. Діаграма варіантів використання (Use Case Diagram). Діаграма варіантів використання в UML дозволяє визначити функціональність системи з точки зору користувачів. У контексті вебдодатку інтернет-магазину ця діаграма може відображати основні функції, які доступні покупцям та адміністраторам, наприклад, додавання товару в кошик, оформлення замовлення, редагування товарів тощо.

2. Діаграма класів (Class Diagram). Діаграма класів дозволяє моделювати структуру системи, включаючи класи, атрибути, методи та їхні взаємозв'язки. Ця діаграма може відображати класи, такі як «Товар», «Кошик», «Замовлення» та їхні взаємозв'язки, як-от асоціації, агрегації та спадкування.

3. Діаграма активностей (Activity Diagram). Діаграма активностей дозволяє моделювати послідовність дій або процеси в системі, може відображати процес оформлення замовлення або керування запасами товарів.

Ці можливості UML є лише кількома з великого спектру інструментів, які можна використовувати в процесі проектування вебдодатків. Комбінація цих діаграм дозволяє детально проаналізувати систему, зрозуміти її функціональність та взаємодії, а також документувати проект на рівні, зрозумілому для розробників та інших зацікавлених сторін.

Перш за все, варто визначити основні компоненти системи веб-додатку для замовлень в шкільній їдальні. Ці компоненти включатимуть, але не обмежуються, наступними елементами:

- клієнтський інтерфейс – вебсторінка або додаток, який користувачі шкільної їдальні використовуватимуть для замовлення їжі. Він повинен мати зручний та інтуїтивно зрозумілий інтерфейс, щоб користувачі могли легко здійснювати свої замовлення;

- база даних – компонент, який зберігатиме всю інформацію про страви, доступні для замовлення, інформацію про користувачів тощо. База даних буде використовуватись для збереження та управління даними, пов'язаними з процесом замовлень;
- адміністративний інтерфейс – вебсторінка, яка використовуватиметься адміністратором шкільної їдальні для керування системою. Він повинен надавати можливості для додавання та редагування страв, перегляду замовлень.

Після визначення основних компонентів необхідно розглянути їхні взаємозв'язки та взаємодію між собою. Основні взаємозв'язки вебдодатку для замовлень в шкільну їдальню можуть включати:

- клієнтський інтерфейс та база даних. Клієнтський інтерфейс буде звертатись до бази даних для отримання інформації про страви, розклади та інші деталі. Він також буде надсилати дані замовлення до бази даних для збереження;
- адміністративний інтерфейс та база даних. Адміністративний інтерфейс буде звертатись до бази даних для отримання даних про страви, розклади та звіти. Він також буде здійснювати зміни в базі даних, такі як додавання та редагування страв, управління розкладами тощо.

Однією з ключових UML-діаграм, що використовується для проектування, є діаграма варіантів використання (Use Case Diagram). Вона демонструє основних користувачів системи, їхні ролі та доступні дії.

Основними користувачами є учні, адміністратори їдальні, а також система керування замовленнями. Діаграма відображає сценарії взаємодії користувачів із системою, включаючи реєстрацію, перегляд меню, оформлення замовлень та адміністрування меню [8].

Наведені нижче алгоритми допоможуть зрозуміти логіку та послідовність операцій у кожному модулі, які забезпечують виконання відповідних функцій системи.

Модуль каталогу страв:

- користувач переглядає каталог доступних страв;
- система завантажує список страв з бази даних та відображає їх на сторінці;
- користувач може додати обрану страву до кошика.

Модуль кошика:

- користувач переглядає вміст свого кошика, де відображені обрані страви;
- користувач може змінювати кількість порцій страви або видаляти страви з кошика;
- система розраховує загальну суму замовлення на основі обраних страв та їх кількості.

Модуль замовлення:

- користувач вводить необхідну інформацію для замовлення;
- система перевіряє наявність страв у кошику та їх доступність на вказаний час доставки;
- якщо, система створює нове замовлення з відповідною інформацією;
- замовлення додається до бази даних та маркується як непідтверджене;
- адміністратор системи отримує повідомлення про нове замовлення.

Ці алгоритми визначають основну логіку функціонування системи веб-додатку «School kitchen».

На рисунку 2.4 наведено діаграму активностей вебресурсу зі сторони користувача, яка показує алгоритм створення замовлення користувачем. Вона показує потік управління або даних між діями, рішеннями, розгалуженнями та паралельними процесами. Така діаграма допомагає зрозуміти логіку процесу, виявити неефективності та спростити комунікацію між членами команди.

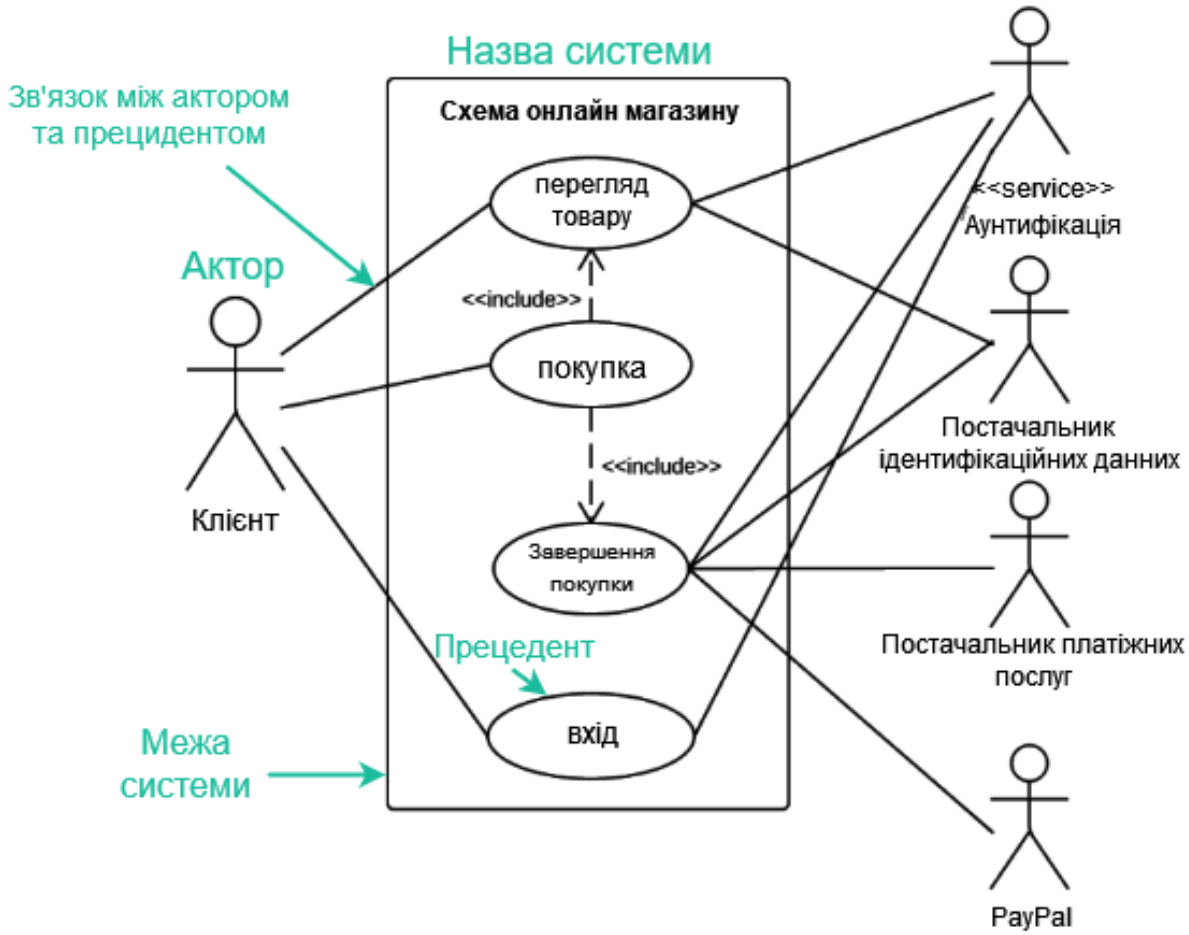

Рисунок 2.4 – «Use Case» – діаграма вебдодатку

Загальна структурна схема функціонування вебдодатку допомагає зрозуміти основні компоненти системи та їхні взаємозв'язки. Вона є важливим етапом проектування, що допомагає уникнути непорозумінь та помилок у подальшій розробці.

Діаграма класів дозволяє моделювати структуру системи, вказуючи класи, їх атрибути, методи та взаємозв'язки між ними. Вона відображає основні сутності системи та їх взаємозв'язки [9].

Діаграма класів є однією з найважливіших у процесі об'єктно-орієнтованого проєктування, оскільки вона відображає основні елементи системи, їх взаємодію та структуру.

Діаграма класів допомагає зрозуміти логіку організації коду, сприяє правильному розподілу відповідальностей між класами та слугує основою для реалізації програмного забезпечення.

В цьому випадку вебдодаток має дві сторінки для різного призначення, які мають свої окремі класи, атрибути та методи, що в них використовуються. На рисунку 2.5 наведено діаграму класів сторінки користувачів вебдодатку «School kitchen».

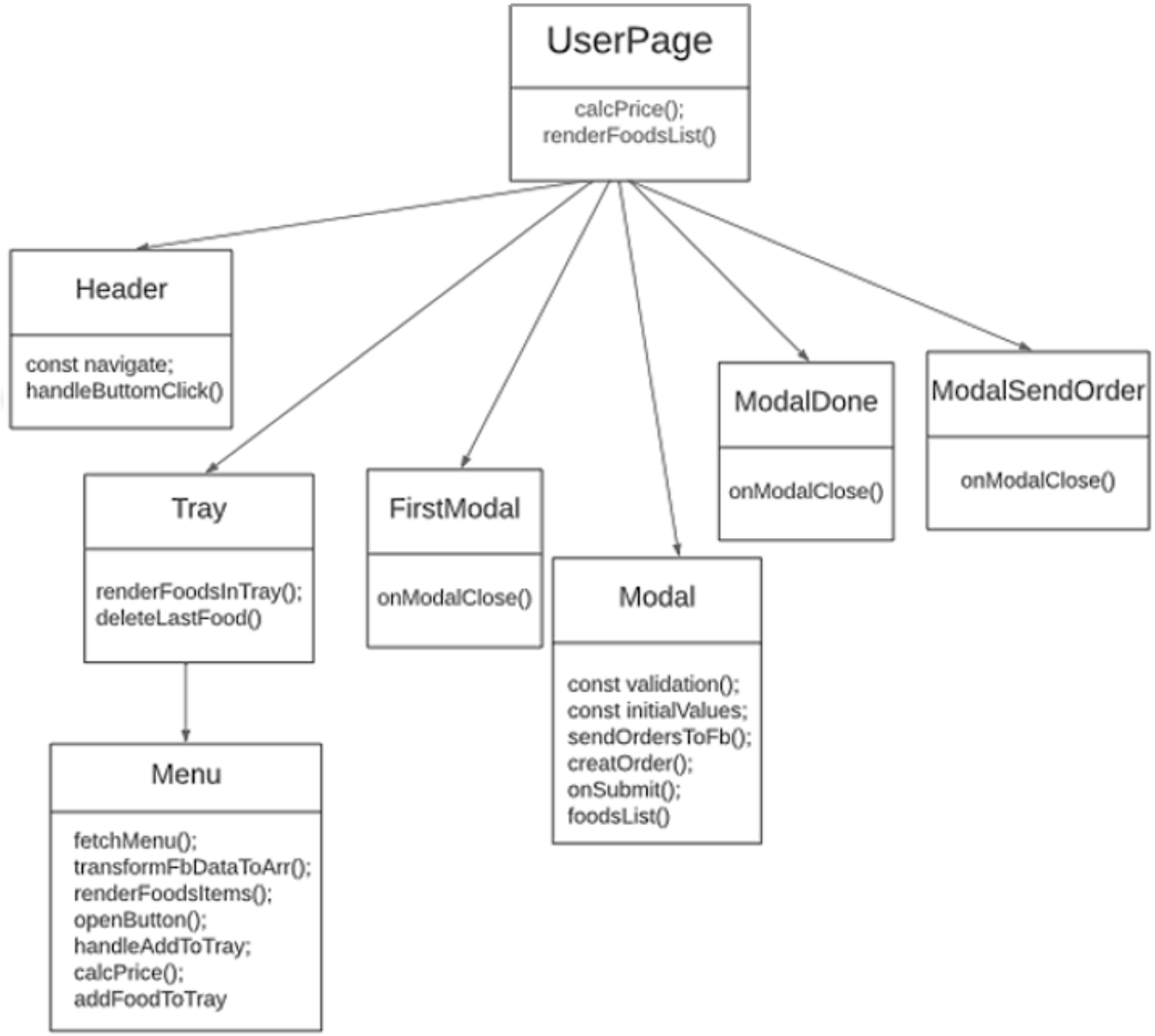

Рисунок 2.5 – Діаграма класів сторінки користувачів вебресурсу

На рисунку 2.6 наведено діаграму класів сторінки адміністрації їдальні вебдодатку «School kitchen».

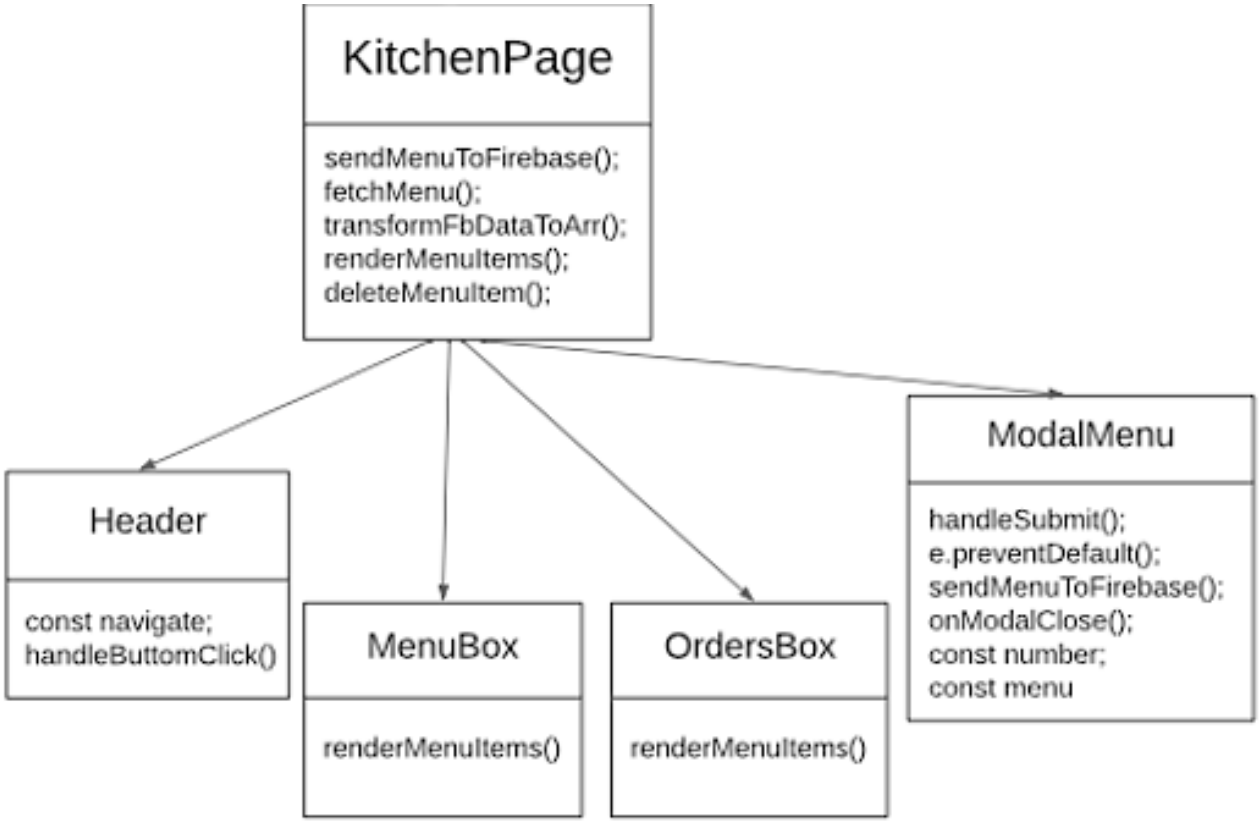

Рисунок 2.6 – Діаграма класів сторінки адміністрації їдальні вебресурсу

Для розробки загальної структурної схеми створено діаграму активностей, тобто послідовність дій і процесів цього вебдодатку. На рисунку 2.7 зображено функціональність системи з точки зору користувача та адміністратора кухні.

Кожна колонка відповідає за різні сторінки вебдодатку та їхні можливості у використанні. Діаграма активностей — це вид поведінкової діаграми в UML, яка використовується для моделювання динаміки процесів у системі. Вона відображає послідовність дій, переходів між ними та можливі умови чи паралельні гілки виконання.

Рисунок 2.7 – Діаграма активностей вебдодатку

Застосування UML-діаграм у процесі проектування вебресурсу «School kitchen» дозволяє формалізувати архітектуру системи, спрощує її розуміння та подальшу реалізацію.

Використання цих моделей допомагає оптимізувати процес розробки, зменшити кількість помилок та забезпечити відповідність системи визначеним вимогам.

2.3 Розробка бази даних для вебресурсу

Розробка бази даних є ключовим етапом створення вебресурсу «School kitchen», оскільки саме вона забезпечує збереження та управління всією інформацією, необхідною для функціонування системи. Вибір правильної моделі бази даних впливає на швидкість обробки запитів, цілісність даних та можливість їх масштабування.

Бази даних поділяються на дві основні категорії: реляційні та нереляційні.

Реляційні бази даних (SQL) організовують дані у вигляді таблиць, між якими встановлюються зв’язки. Вони використовують мову SQL (Structured Query Language) для маніпуляції даними. Такі бази забезпечують високу структурованість, підтримку транзакцій, забезпечення цілісності даних та їхню нормалізацію. Прикладами реляційних СУБД є MySQL, PostgreSQL, Microsoft SQL Server [10-12].

Нереляційні бази даних (NoSQL) використовують різні моделі зберігання, такі як документи, графи, стовпці або ключ-значення. Вони забезпечують гнучкість, горизонтальне масштабування та швидку обробку великих обсягів неструктурованих даних. До популярних NoSQL рішень належать MongoDB, Firebase Firestore, Cassandra.

Для вебресурсу «School kitchen» була обрана реляційна база даних через необхідність забезпечення структурованих зв’язків між користувачами, замовленнями та меню. Це дозволить підтримувати складні запити, забезпечувати цілісність даних та ефективно працювати з великою кількістю користувачів.

Основні сутності бази даних «School kitchen»:

- користувачі (users): містить інформацію про всіх зареєстрованих користувачів системи, включаючи учнів, батьків та адміністраторів. Кожен користувач має унікальний ідентифікатор;
- меню (menu): містить перелік доступних страв, їхні назви, описи, ціни та категорії;

- замовлення (orders): містить інформацію про кожне замовлення, його статус, дату створення, пов’язаного користувача та страви;
- страви у замовленні (order_Items): таблиця зв’язку, що зберігає інформацію про страви, які входять до певного замовлення.

Атрибути та ключі сутностей:

- users (user_id, name, email, role, password_hash);
- menu (dish_id, name, description, price, category);
- orders (order_id, user_id, order_date, status);
- order_Items (order_id, dish_id, quantity).

user_id, dish_id, order_id – це первинні ключі, які однозначно ідентифікують записи.

order_id та dish_id у таблиці Order_Items є зовнішніми ключами, що встановлюють зв’язки між таблицями.

Сформуємо універсальне відношення та визначимо його ступінь. Для цього перерахуємо усі атрибути, що описують сутності. Універсальне відношення для цієї бази даних буде мати наступний вигляд:

R (user_id, name, email, role, password_hash, dish_id, name, description, price, category, order_id, user_id, order_date, status, order_id, dish_id, quantity).

Отже, ступінь універсального відношення – 17.

ER-модель – це модель даних, яка дозволяє описати деяку предметну область бази даних за допомогою узагальнених конструкцій блоків. В ER-моделі дані представляються у вигляді компонентів (сутностей), які пов'язані між собою певними зв'язками. ER-модель бази даних для вебресурсу «School kitchen » можна побачити на рисунку 2.8.

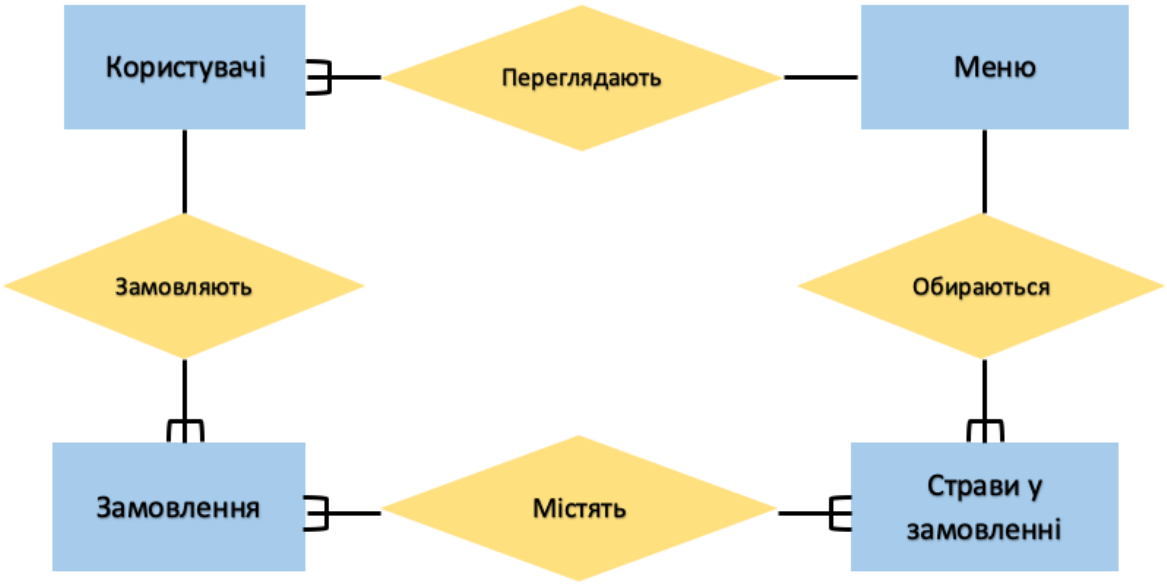

Рисунок 2.8 – ER- модель бази даних вебресурсу

У цій моделі можна побачити наступні зв’язки між сутностями:

1. Користувачі – Меню. Зв’язок багато до одного. Багато користувачів можуть переглядати меню, проте лише одне меню може переглядатися багатьма користувачами.

2. Меню – Страви у замовленні. Зв’язок один до багатьох. Меню може вміщати у собі багато страв у замовленні, проте страви у замовленні містяться лише в одному меню.

3. Замовлення – Страви у замовленні. Зв’язок багато до багатьох. Багато замовлень можуть містити страву, так само як і страва може міститися в багатьох замовленнях.

4. Користувачі – Замовлення. Зв’язок один до багатьох. Користувачі можуть робити багато замовлень, проте замовлення може належати одному конкретному користувачеві.

Для забезпечення відсутності аномалій оновлення та видалення база даних повинна відповідати Другій нормальній формі (2НФ).

Перша нормальна форма (1НФ) – усі атрибути містять лише атомарні значення (немає списків чи множинних значень у комірках).

Друга нормальна форма (2НФ) – виконується 1НФ, і всі неключові атрибути залежать тільки від первинного ключа.

У нашій моделі всі неключові атрибути в таблиці Orders (order_date, status) залежать від order_id, а в Menu (name, description, price, category) – від dish_id. У таблиці Order_Items всі атрибути залежать від комбінації order_id та dish_id, що забезпечує відповідність 2НФ.

Таким чином, спроєктована реляційна база даних «School kitchen» відповідає всім вимогам до збереження структурованих даних, підтримує зв'язки між сутностями та відповідає Другій нормальній формі для забезпечення цілісності.

2.4 Розробка схеми алгоритму роботи вебресурсу «School kitchen»

Схема алгоритму – це графічне зображення послідовності дій, що виконуються в рамках певного процесу. Вона дозволяє наочно представити логіку роботи системи, визначити можливі помилки або неефективності, а також оптимізувати робочий процес [13-14].

Створення схеми алгоритму відбувається у кілька етапів:

1. Визначення основних етапів процесу.
2. Визначення взаємозв'язків між діями.
3. Графічне відображення послідовності виконання за допомогою блок-схем.
4. Аналіз і тестування алгоритму для виявлення потенційних покращень.

Основними елементами схеми алгоритму є:

- початок та кінець процесу – відображаються у вигляді овалів;
- дії або процеси – прямокутники, що позначають операції або функції;
- рішення або перевірки умов – ромби, що вказують на необхідність ухвалення рішення;
- стрілки – позначають послідовність виконання операцій.

Алгоритм роботи вебресурсу включає наступні основні етапи:

1. Реєстрація та автентифікація користувачів:

- користувач заходить на сайт і вибирає варіант реєстрації або авторизації;

- якщо це новий користувач, він заповнює реєстраційну форму, вводить електронну пошту, пароль, клас (для учнів) або інші ідентифікаційні дані;

- вхідні дані перевіряються, після чого створюється обліковий запис у базі даних;

- після успішної автентифікації користувач потрапляє в особистий кабінет.

2. Перегляд меню та вибір страв:

- користувач переходить до меню, яке відображає доступні страви;

- меню оновлюється в реальному часі згідно з даними адміністратора кухні;

- користувач обирає бажані страви та додає їх у кошик.

3. Формування та підтвердження замовлення:

- користувач переходить до кошика, переглядає список обраних страв, їхню вартість та загальну суму;

- вибирає час отримання замовлення;

- підтверджує замовлення;

- замовлення надходить у базу даних та передається адміністратору кухні.

4. Обробка замовлення адміністратором:

- адміністратор отримує список замовлень у спеціальному інтерфейсі;

- відмічає замовлення як виконане або вносить зміни у статус;

- користувач отримує сповіщення про статус замовлення.

5. Отримання замовлення та фіналізація процесу:

- користувач приходить у шкільну їдальню у зазначений час;

- адміністратор видає замовлення;

- у базі даних оновлюється статус замовлення.

На рисунку 2.9 зображена блок-схема роботи вебресурсу «School kitchen».

Рисунок 2.9 – Блок-схема роботи вебресурсу «School kitchen»

2.5 Висновок до розділу 2

У цьому розділі було здійснено проєктування структури та компонентів вебресурсу «School kitchen», що включало вибір архітектури, створення UML-діаграм, розробку бази даних та розробку алгоритму роботи системи.

Було обґрунтовано вибір архітектури вебресурсу, який базується на клієнт-серверній моделі. Це забезпечує гнучкість, масштабованість та зручність використання вебресурсу. Було визначено основні технології, які будуть використані для реалізації, а також їх переваги.

У межах розділу було створено UML-діаграми, що дозволили наочно представити структуру вебресурсу, його основні компоненти та взаємодію між ними. Діаграми випадків використання, послідовностей та класів допомогли деталізувати роботу системи та полегшили процес її подальшої реалізації.

Також було розроблено структуру бази даних для вебресурсу. Були визначені сутності, зв’язки між ними, універсальне відношення та його ступінь, створена ER-модель та структура для зберігання інформації про користувачів, замовлення та меню. Це забезпечує ефективне збереження та обробку даних.

Далі було розроблено алгоритм функціонування системи. Він охоплює ключові етапи роботи користувачів та адміністратора кухні, включаючи реєстрацію, вибір страв, оформлення замовлення, його обробку та отримання страв у їдальні. Використання блок-схем дозволило чітко структурувати процеси та оптимізувати логіку роботи сервісу.

Таким чином, виконане проєктування стало фундаментом для подальшої реалізації вебресурсу «School kitchen», забезпечуючи ефективне функціонування системи та автоматизацію процесів замовлення та видачі страв у шкільних їдальнях.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ВЕБДОДАТКУ «SCHOOL KITCHEN»

У цьому розділі розглянуто програмну реалізацію та процес тестування вебресурсу «School kitchen» з використанням об'єктно-орієнтованого підходу та фреймворку React.js. Наведені підрозділи обґрунтують вибір мови програмування та описують основні оператори мови програмування JavaScript та бібліотеки React.js, що використовуються у реалізації проекту.

3.1 Обґрунтування вибору мови програмування

Для створення вебресурсу розглядалися кілька мов програмування, зокрема Python і Java, кожна з яких має свої переваги та недоліки.

Python є однією з найпопулярніших мов програмування, що використовується у різних галузях, включаючи веброзробку. Його головна перевага полягає у простоті синтаксису, що робить його легким для навчання. Крім того, мова має велику стандартну бібліотеку, що дозволяє швидко розробляти різні типи додатків. Python часто використовується у бекенд-розробці, аналітиці даних та машинному навчанні [15-16].

Однак, Python має і суттєві недоліки. Він поступається у швидкості виконання мовам зі статичною типізацією, таким як Java. Крім того, можливості Python у фронтенд-розробці обмежені, оскільки браузери безпосередньо не підтримують цю мову. Для використання Python у вебдодатках необхідно застосовувати додаткові технології, такі як Django чи Flask для бекенду.

Java є потужною мовою програмування, яка широко використовується у корпоративному секторі та розробці великих систем. Її основною перевагою є висока продуктивність та можливість кросплатформного використання завдяки віртуальній машині Java (JVM). Java також відома своєю стабільністю та безпекою, що робить її привабливою для розробки складних програмних продуктів.

Проте Java має більш складний синтаксис порівняно з іншими мовами, що ускладнює її вивчення та застосування. Крім того, вона не є найкращим вибором для фронтенд-розробки, оскільки для створення вебінтерфейсів потрібно використовувати додаткові технології, такі як JavaServer Faces або Vaadin. У сфері веброзробки Java частіше застосовується для серверної частини, ніж для клієнтської.

Беручи до уваги вимоги до вебресурсу, було прийнято рішення використовувати JavaScript як основну мову програмування, а саме її бібліотеку React.js.

JavaScript є мовою програмування, яка використовується для розробки як клієнтської, так і серверної частини веб-додатків. Вона використовується практично в усіх браузерах і є однією з найпоширеніших мов для веб-розробки. JavaScript є мовою з високим рівнем динамічності, що дозволяє розробникам легко маніпулювати елементами веб-сторінок і створювати взаємодію з користувачем.

Вона підтримує об'єктно-орієнтований підхід до програмування, що дозволяє організовувати код у вигляді об'єктів, методів і властивостей, що полегшує розробку складних систем. Також ця мова дозволяє створювати динамічні та інтерактивні вебсторінки, що робить її ідеальним вибором для фронтенд-розробки.

Наведемо основні переваги JavaScript:

- універсальність. JavaScript може використовуватися як на стороні клієнта, так і на стороні сервера. Це дозволяє розробникам використовувати одну мову програмування для обох частин веб-додатку, що спрощує розробку та обмін даними між ними;
- швидкість розробки. JavaScript має простий синтаксис і добре документований, що дозволяє розробникам швидко писати код і проводити тестування;

- багата екосистема. JavaScript має велику кількість бібліотек і фреймворків, таких як React.js, Angular.js, Vue.js, що допомагають розробникам прискорити розробку, забезпечити більшу функціональність і покращити продуктивність.

Недоліки JavaScript:

- кросс-браузерність. JavaScript може працювати по-різному на різних браузерах, іноді призводячи до непередбачуваної поведінки. Це може створювати проблеми з розробкою і тестуванням;
- відсутність типізації. JavaScript є слабко типізованою мовою, що може призводити до помилок, пов'язаних з типами даних;
- відсутність підтримки великих проектів. JavaScript не має вбудованих механізмів для роботи з великими проектами, що може призводити до проблем з обслуговуванням і розширенням коду.

JavaScript має велику кількість бібліотек і фреймворків, які допомагають розробникам прискорити процес розробки, надати більшу функціональність та забезпечити високу якість коду.

React.js є популярним фреймворком JavaScript для розробки користувацьких інтерфейсів. Він базується на компонентній архітектурі, що дозволяє розбити інтерфейс на невеликі незалежні компоненти, що полегшує розробку, тестування та обслуговування коду. React.js використовує віртуальний DOM, що забезпечує швидку і ефективну роботу з інтерфейсом користувача [17].

React.js має велику спільноту розробників, яка надає багато готових компонентів, бібліотек і інструментів для прискорення розробки та полегшення роботи з React-додатками.

Основні переваги React.js:

- швидкість та ефективність. Використання віртуального DOM дозволяє React.js швидко оновлювати інтерфейс користувача і мінімізувати кількість зайвих операцій з DOM;

- перевикористання компонентів. React.js підтримує компонентну архітектуру, що дозволяє легко перевикористовувати компоненти і створювати складні інтерфейси з невеликою кількістю коду;
- спільнота розробників. Широка спільнота розробників React.js надає підтримку, документацію, готові рішення та багато корисних ресурсів для розробників.

Недоліки React.js:

- велика кількість концепцій. React.js вимагає знання і розуміння деяких додаткових концепцій, таких як JSX і життєві цикли компонентів;
- вимоги до вивчення. Розробка з використанням React.js може вимагати деякого часу для вивчення і освоєння його основних принципів та практик.

Обґрунтовуючи вибір JavaScript і React.js для розробки веб-додатку «School kitchen», враховується їх широка популярність, універсальність, велика спільнота розробників, зручний синтаксис та можливості перевикористання компонентів. Використання JavaScript і React.js дозволить реалізувати потрібну функціональність, забезпечити ефективну розробку та високу якість веб-додатку.

3.2 Обґрунтування вибору середовищ розробки

Одним із варіантів вибору середовища для програмування було IntelliJ IDEA, яке є потужним середовищем для розробки на багатьох мовах програмування, включаючи JavaScript. Воно забезпечує автоматичне доповнення коду, інтегровані засоби для роботи з системами контролю версій та багатий функціонал для дебагінгу. Проте IntelliJ IDEA має досить високу вимогливість до системних ресурсів, що може ускладнити роботу на менш потужних пристроях.

Ще одним варіантом було WebStorm, яке спеціалізується на розробці JavaScript-додатків. Це середовище має зручні інструменти для роботи з фреймворками, такими як React.js, Angular та Vue.js. WebStorm підтримує

розширене автодоповнення коду та має вбудований налагоджувач. Проте головним недоліком цього середовища є його комерційна модель – повноцінне використання потребує придбання ліцензії, що може бути недоцільним для індивідуальних розробників або малих команд [18-19].

Зрештою, було обрано Visual Studio Code – легке, швидке та функціональне середовище розробки, яке має такі переваги:

- безкоштовність – VS Code є відкритим і доступним для всіх користувачів;
- легка вага – займає менше ресурсів, ніж WebStorm чи IntelliJ IDEA;
- гнучкість – має велику кількість розширень, що дозволяють додавати підтримку нових технологій та бібліотек;
- інтеграція з Git – спрощує контроль версій та командну розробку;
- зручне автодоповнення – покращує швидкість написання коду, особливо при роботі з React.js.

Для зберігання даних вебдодатку необхідно було обрати надійну платформу для роботи з базами даних. Було також розглянуто кілька варіантів.

Одним із можливих рішень було використання MySQL – реляційної бази даних, яка забезпечує високу продуктивність та підтримує складні SQL-запити. Проте для інтеграції з вебдодатком необхідно додатково налаштовувати сервер баз даних, що ускладнює розгортання проєкту.

Іншим варіантом було MongoDB, що є нереляційною базою даних і добре підходить для гнучких вебпроєктів. MongoDB дозволяє швидко масштабувати застосунок, проте потребує окремого сервера або хмарного середовища для розміщення [20-21].

Оптимальним варіантом виявився Firebase – хмарна база даних від Google, яка забезпечує легку інтеграцію з вебдодатками та має такі переваги:

- реальний час оновлення – дозволяє автоматично оновлювати дані на сторінці без потреби у запитах;
- хмарне зберігання – не потребує окремого сервера та спрощує розгортання додатку;

- зручний API – забезпечує легке підключення через JavaScript;
- безпека – підтримує гнучку систему доступу до даних [22].

Таким чином, було обрано Visual Studio Code як основне середовище розробки та Firebase як платформу для бази даних, що дозволяє забезпечити швидку розробку, ефективне тестування та масштабованість вебресурсу «School kitchen».

3.3 Опис програмних рішень

Програмна реалізація вебресурсу створена на функціональних компонентах та хуках. Хуки це спеціальні функції, які дозволяють використовувати стан (state) та інші функціональності React в функціональних компонентах.

Так як в нашому вебресурсі є дві сторінки (користувацька та адміністраторська), програмний код вміщає дві компоненти UserPage і KitchenPage, які є батьківськими для цих сторінок відповідно.

Зупинимося спочатку на компоненті UserPage. Вона вміщає в собі усю верстку сторінки та дочірні компоненти: Header, Tray, FirstModal, Modal, ModalOrders.

Header – компонента, яка являє собою шапку сторінки, до речі, вона є однаковою в обох сторінках цього додатку.

Tray вміщає в себе корзину, кнопку видалення товару, підрахунок суми обраних товарів.

FirstModal – компонета модального вікна, яка відображає відео-інструкцію для користувача, який вперше відвідав додаток.

Modal – модальне вікно з формою для оформлення замовлення. ModalOrders – модальне вікно, яке відображає відправлене замовлення користувача на його сторінці.

В Tray є дочірня компонента Menu, яка дуже важливою, адже саме в ній прописаний функіонал витягування даних з бази даних Firebase та формування товарів в каталозі. Далі розглянемо деякі функції цієї компоненти.

fetchMenu() використовує асинхронний підхід для отримання меню з вказаного URL і зберігає його в стані компонента.

transformFbDataToArr(fbData) приймає об'єкт fbData, який містить дані з сервера Firebase, і перетворює його в масив об'єктів з додаванням ідентифікатора до кожного об'єкта:

```
function transformFbDataToArr(fbData) {
 if (fbData) {
  return Object.keys(fbData).map((key) => {
   const item = fbData[key];
   item.id = key;
   return item;
  });
 } else{
  return
 }
```

TrayItem та TrayList, відповідають за відображення елементів їжі на сторінці веб-додатку "School Kitchen". Подамо ці функції таким чином:

```
const TrayItem = ({ item }) => {
 return (
  <div className="app__tray-item-box">
   <img src={item.imgUrl} alt="food" className="app__food-img" />
   <span className="app__tray-count">x {item.count}</span>
  </div>
 );
};
const TrayList = ({ foods }) => {
 return (
```

```
      <div className="app__tray">
        {foods.length > 0 ? (
          foods.map((item, index) => <TrayItem key={index} item={item} />)
        ) : (
          <p className="no-items">Лоток порожній</p>
        )}
      </div>
    );
  };
```

CalcPrice(arr) функція обчислює загальну вартість страв у кошику. Вона отримує масив arr з елементами їжі та обчислює загальну вартість, перебираючи кожен елемент і додаючи до загальної суми ціну помножену на кількість:

```
  function calcPrice(arr) {
    let totalPrice = 0;
    arr.forEach((item) => {
      totalPrice += +item.price * item.count;
    });
    totalPriceNode.innerHTML = totalPrice;
    totalPriceForm.innerHTML = totalPrice;
  }
```

addFoodToTray(selectedItem) функція додає обраний елемент їжі до лотка замовлення. Функція отримує необхідну інформацію про елемент їжі (id, назва, ціна, зображення) та створює новий об'єкт newFood з цією інформацією. Потім перевіряється, чи вже існує елемент з таким самим id у кошику. Якщо так, збільшується значення count для цього елементу, якщо ні, то елемент додається до кошика:

```
  function addFoodToTray(dragEl) {
    let id = dragEl.getAttribute("data-id");
    let name = dragEl.querySelector(".app__food-name").textContent;
    let price = dragEl.querySelector(".app__food-price-num").textContent;
```

```
  let imgUrl = dragEl.querySelector(".app__food-img").getAttribute("src");
  let newFood = {
    id: id,
    name: name,
    price: price,
    imgUrl: imgUrl,
    count: 1,
  };

  if (trayArr.length == 0) {
    trayArr.push(newFood);
    return;
  }

  let flag = false;

  trayArr.forEach((obj) => {
    if (obj.id == newFood.id) {
      obj.count += 1;
      flag = true;
    }
  });

  if (flag == false) {
    trayArr.push(newFood);
  }
}
```

Розглянемо компоненту KitchenPage, яка відповідає за іншу сторінку. Вона в свою чергу також містить усю необхідну верстку сторінки та компоненти: Header, MenuBox, OrdersBox, ModalMenu. Як зазначено вище, компонента

Header є однаковою в обох сторінках. MenuBox – відображає блок зі списком доданих страв в меню, а OrdersBox – зі списком замовлень від користувачів. ModalMenu відкриває модальне вікно з формою для додавання нових страв в меню та надсилання їх у базу даних. Серед основних функції маємо:

- компонент OrderItem рендерить кожне замовлення;
- використано useState для статусу виконання замовлення;
- використано map для створення списків;
- OrdersList рендерить список або повідомлення, якщо замовлень немає.

У програмному вигляді напишемо це наступним чином:

```
const OrderItem = ({ order }) => {
  const [isDone, setIsDone] = useState(
    localStorage.getItem(order.id) === "true"
  );
  const toggleStatus = () => {
    const newStatus = !isDone;
    setIsDone(newStatus);
    localStorage.setItem(order.id, newStatus.toString());
  };
  const orderPrice = order.list.reduce(
    (sum, food) => sum + food.count * +food.price,
    0
  );
  const deadline = new Date(order.deadline);
  const deadlineStr = `${deadline.toLocaleDateString()}
${deadline.getHours()}:${deadline.getMinutes()}`;
  return (
    <li className="panel__order-item panel__order-item-dark">
      <div className="panel__order-item-box">
        <div                    className="panel__order-timeout">Перерва:
{order.breakNum}</div>
        <div className="panel__order-timeout panel__order-date">
          Дата: {order.date}
        </div>
        <div className="panel__order-top">
          <p className="panel__order-info">
            {order.name} {order.surname} Клас - {order.classNum}
```

```
            </p>
            <button
              data-id={order.id}
              className={`panel__order-marker__btn ${isDone ? "done" : ""}`}
              onClick={toggleStatus}
            >
              {isDone ? "Виконано" : "Не виконано"}
            </button>
          </div>
          <ol className="panel__orders-list__items">
            {order.list.map((food, index) => (
              <li key={index} className="panel__orders-food">
                {food.name} x{food.count}
              </li>
            ))}
          </ol>
          <p className="panel__order-fullprice">
            Загальна сума замовлення: {orderPrice} UAH
          </p>
          <p className="panel__order-deadline">Дедлайн - {deadlineStr}</p>
        </div>
      </li>
    );
  };
  const OrdersList = ({ orders }) => {
    return (
      <ul>
        {orders.length > 0 ? (
          orders.map((order) => <OrderItem key={order.id} order={order} />)
        ) : (
          <p className="no-orders no-orders-dark">Замовлень немає</p>
        )}
      </ul>
    );
  };
```

Таким чином, розроблений вебдодаток забезпечує зручне управління замовленнями, зберігання інформації та обробку даних у реальному часі.

3.4 Тестування розробленого вебдодатку «School kitchen»

Під час виконання бакалаврської кваліфікаційної роботи було проведено тестування вебдодатку «School kitchen» за допомогою модуля тестування, що підтвердило коректність його роботи.

Всі дані правильно відображаються на вікнах (рис.3.1).

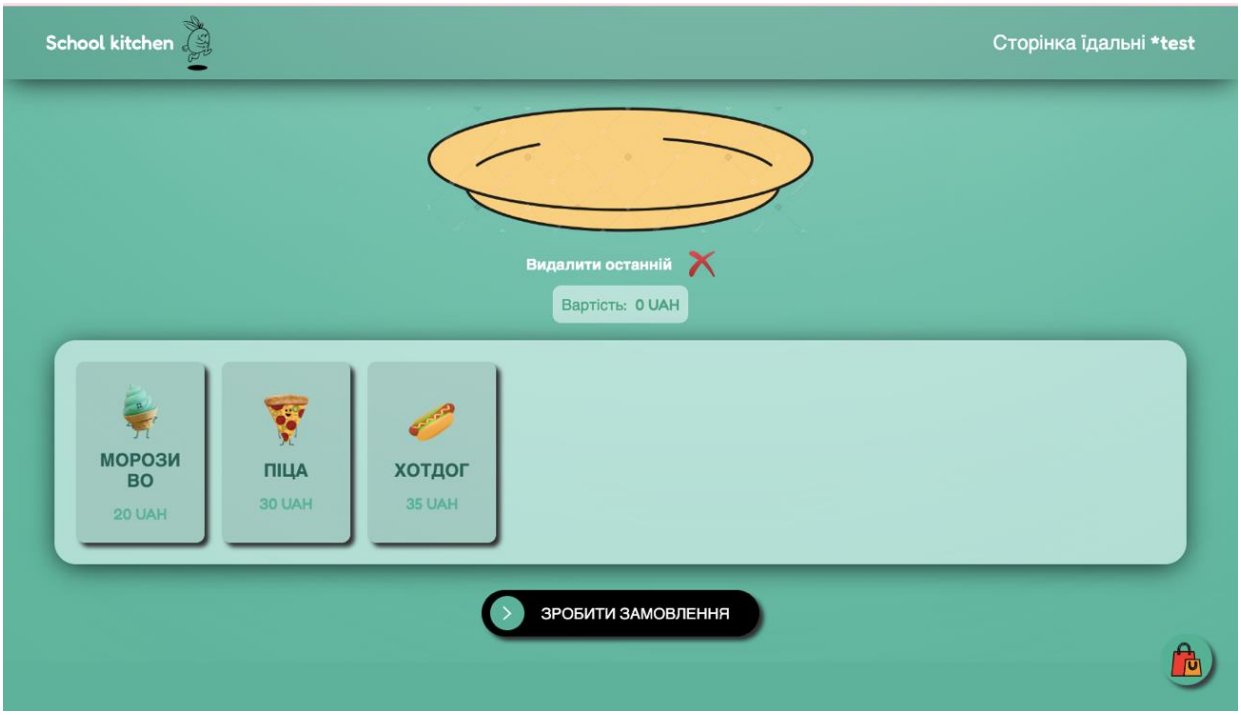

Рисунок 3.1 – Початкова активність

На головній сторінці користувача міститься простий та інтуїтивно зрозумілий інтерфейс. Є велика тарілка, на якій мають відображатися додані страви до замовлення. Нижче є кнопка для видалення останньої доданої страви та підраховується загальна вартість. Далі по центру розташована секція із наявних страв у меню.

При натисканні на товар з’являється кнопка «Додати», натиснувши на яку товар додається в тарілку з позначкою її кількості. На рисунку 3.2 зображено виконання цих дій.

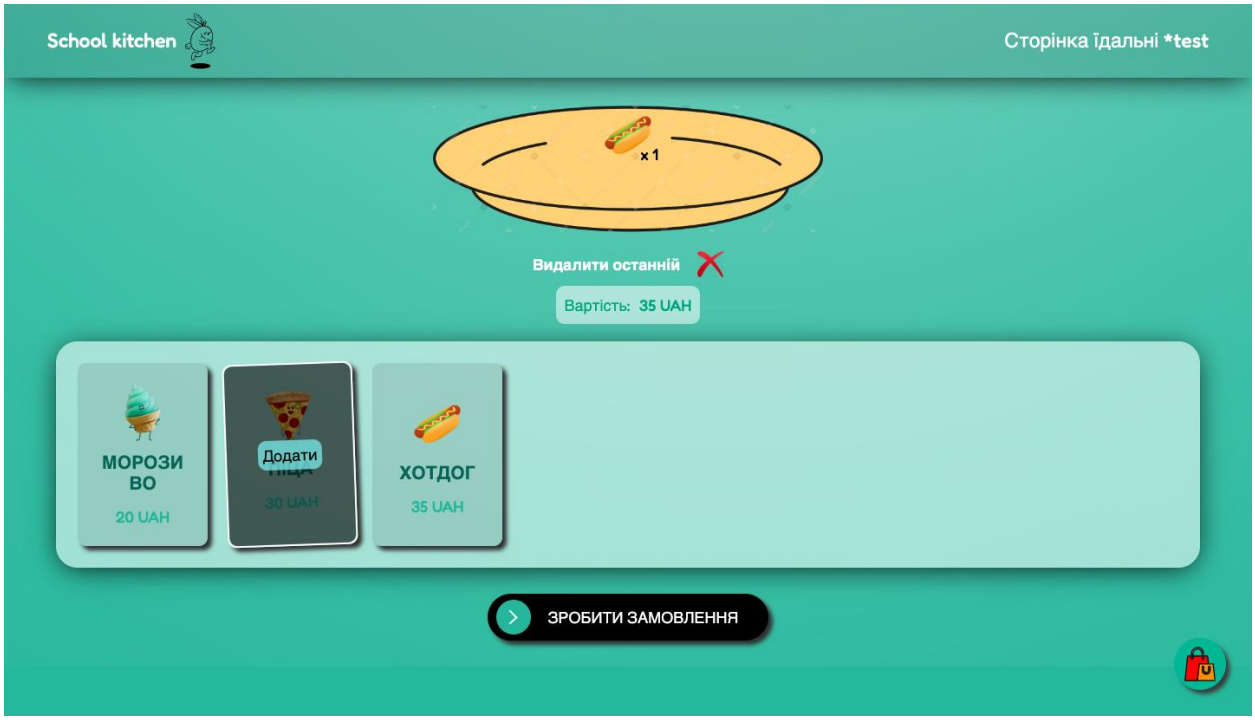

Рисунок 3.2 – Вибір товару

Далі при натисканні кнопки «Зробити замовлення», з’явиться форма для оформлення замовлення. У формі наданий коректно відображений список замовлення та суму.

При введені даних для заповнення форми (рис.3.3), спрацьовує валідація форми, яка в цьому випадку повідомляє про відсутність номеру класу.

Вказавши клас у формі слід натиснути на кнопку «Відправити». Після чого можна побачити замовлення, натиснувши кнопку усіх замовлень (рис.3.4). Також замовлення з’явиться на сторінці кухні.

Перейшовши на сторінку кухні, можна побачити, що замовлення відображається правильно для адміністрації цієї сторінки. Усі деталі, які були вказані користувачем при оформленні замовлення, з точністю передаються на сторінку кухні.

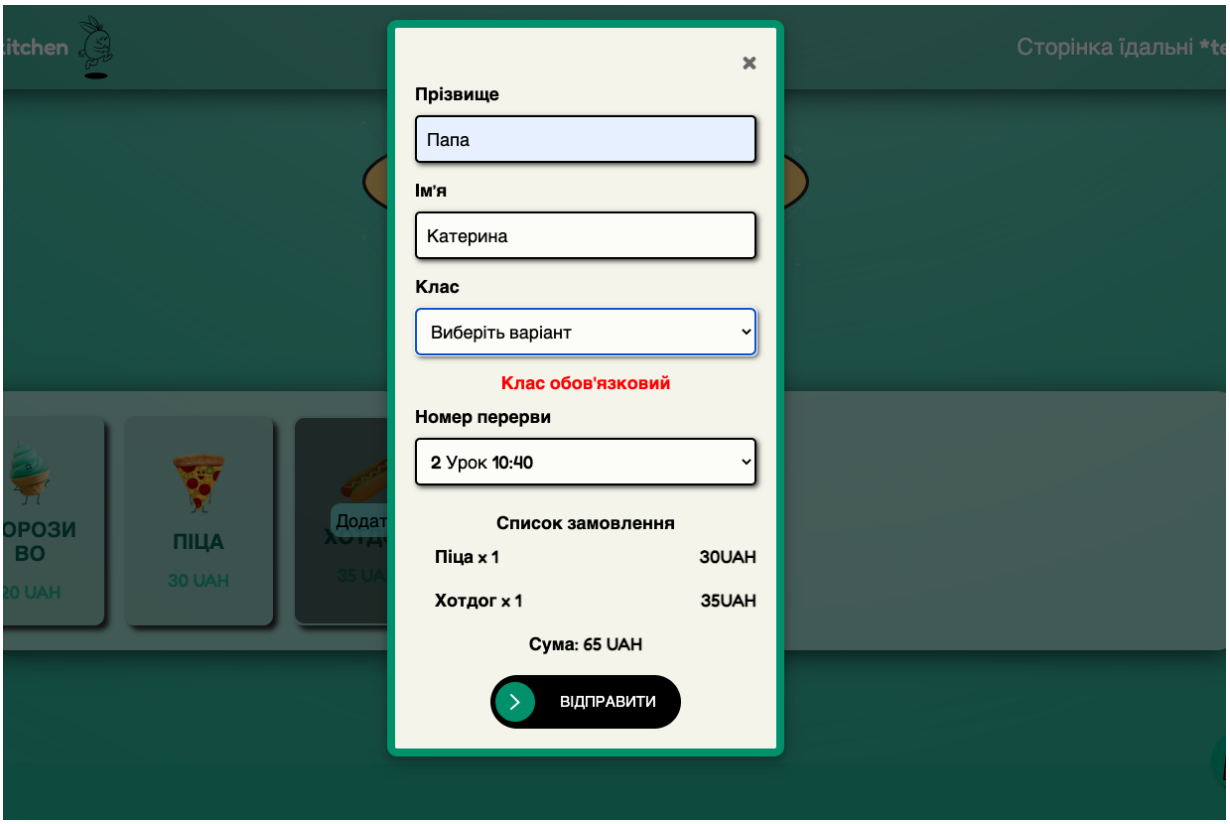

Рисунок 3.3 – Оформлення замовлення

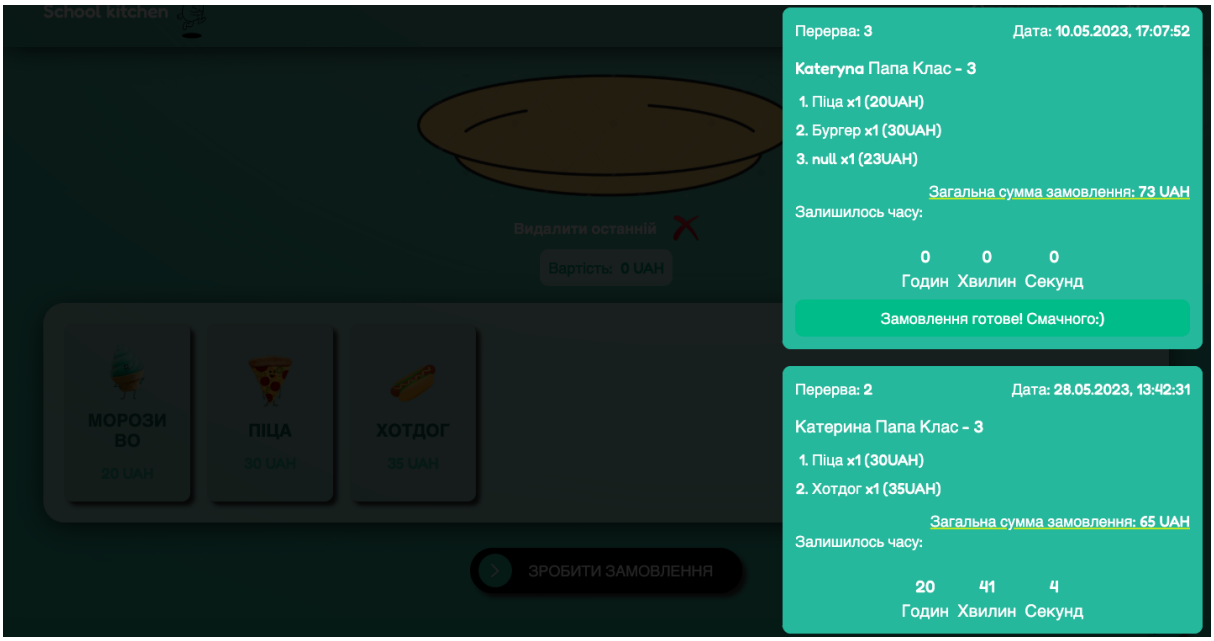

Рисунок 3.4 – Перегляд замовлень на сторінці користувача

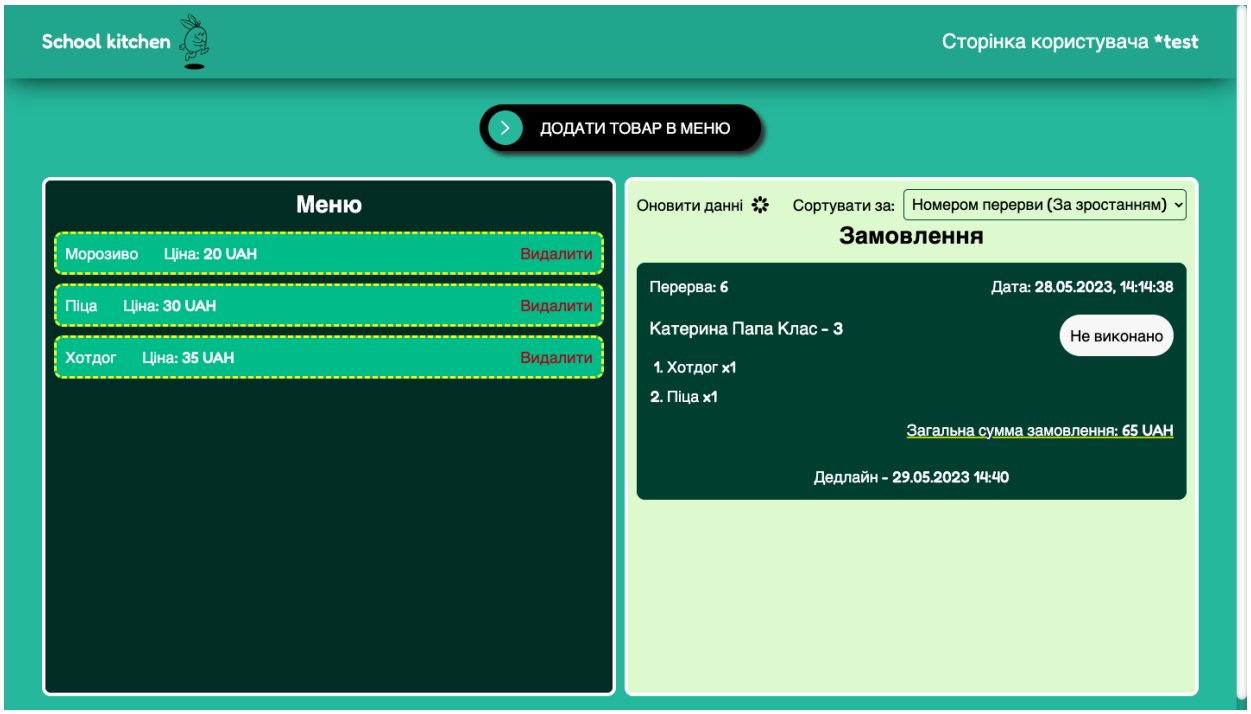

Рисунок 3.5 – Перегляд замовлень на сторінці кухні

На сторінці кухні протестовано також кнопку «Додати товар в меню». Натиснувши її, з’являється форма для додавання страв. На рисунку 3.6 зображена реалізована функція додавання страв до меню з боку адміністрації кухні. Після використання цієї функції, страва з’являється в секції «Меню» на сторінці кухні (рис.3.7) та у каталозі на сторінці користувача (рис.3.8).

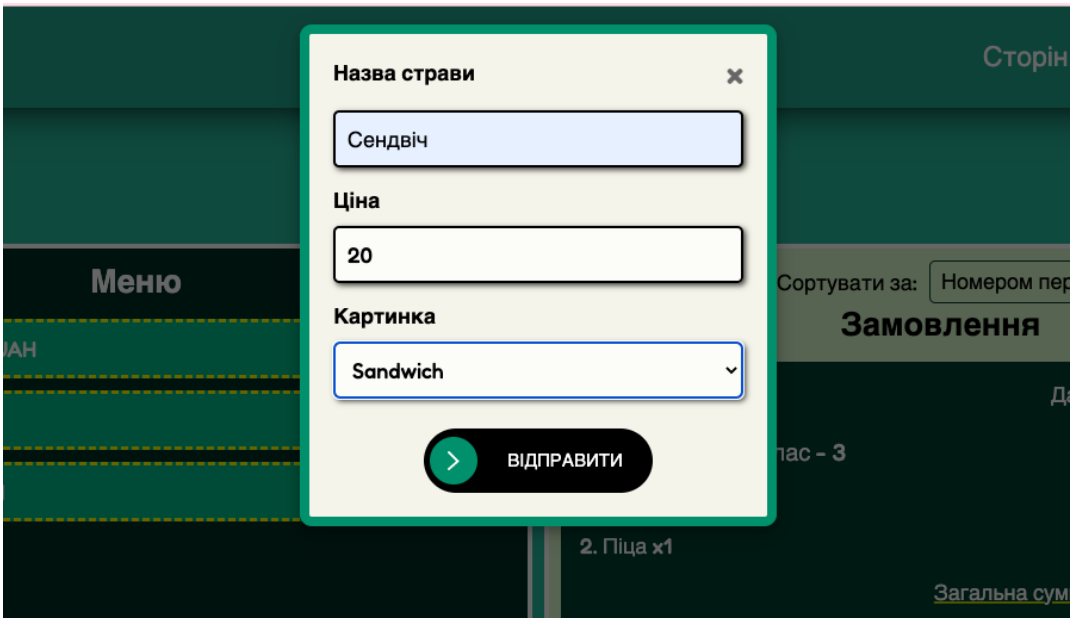

Рисунок 3.6 – Додавання страв в меню

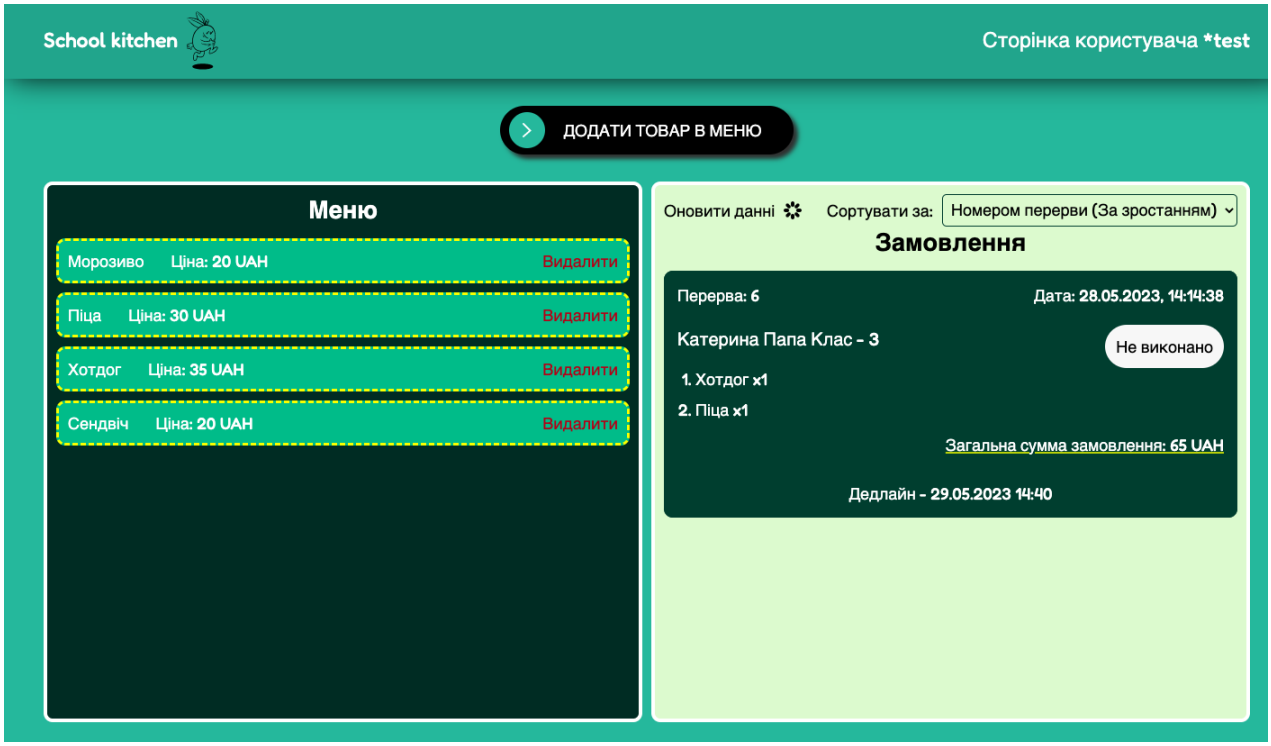

Рисунок 3.7 – Оновлення списку меню

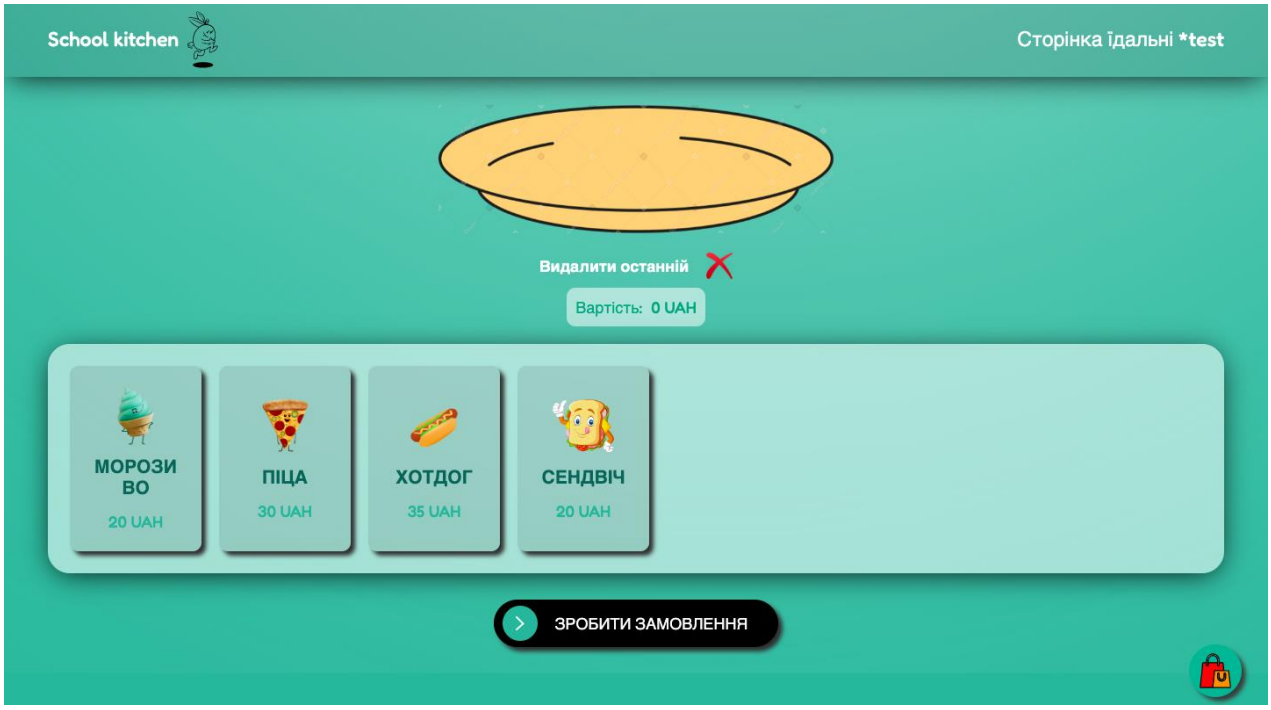

Рисунок 3.8 – Оновлення каталогу товарів

Результати тестування показали, що вебдодаток «School kitchen» працює коректно, без критичних помилок. Функціональність кошика та оформлення

замовлень працює стабільно. Валідація форм дозволяє уникнути некоректних даних. Дані у Firebase зберігаються та оновлюються без збоїв. Загалом, тестування підтвердило працездатність вебдодатку та його готовність до використання.

3.5 Висновок до розділу 3

У третьому розділі бакалаврської кваліфікаційної роботи було детально розглянуто процес програмної реалізації та тестування вебресурсу «School kitchen». Було проведено обґрунтування вибору мови програмування та бібліотек для розробки. Вибір зупинився на JavaScript та бібліотеці React.js, оскільки вони забезпечують високу продуктивність, компонентну архітектуру та зручність у розробці веб-додатків.

Також було обґрунтовано вибір середовища розробки. Серед можливих варіантів розглядалися IntelliJ IDEA, WebStorm та Visual Studio Code. Було прийнято рішення використовувати Visual Studio Code через його легкість, безкоштовність, гнучкість у налаштуванні та підтримку широкого спектра розширень

У підрозділі, присвяченому програмним рішенням, було описано використання компонентного підходу та хуків у React.js. Такий підхід забезпечує модульність, перевикористання коду та ефективну взаємодію між компонентами системи. Було проведено тестування вебдодатку, яке підтвердило його стабільну роботу. Всі функції, такі як додавання страв у кошик, оформлення замовлення, валідація форм та взаємодія з базою даних, працювали без збоїв. Дані зберігалися та оновлювалися у Firebase без втрат, що підтвердило надійність обраної архітектури та технологічного стеку

Таким чином, у цьому розділі було детально описано процес реалізації вебресурсу, підтверджено ефективність вибраного стеку технологій та виконано тестування, яке довело працездатність системи.

ВИСНОВКИ

У ході виконання бакалаврської кваліфікаційної роботи було розроблено вебресурс «School kitchen», який призначений для автоматизації процесу замовлення та обліку харчування у шкільних їдальнях. Робота включала аналіз предметної області, проектування архітектури системи, розробку бази даних, реалізацію програмного забезпечення та тестування отриманого рішення.

На першому етапі дослідження було проведено аналіз існуючих систем автоматизації шкільного харчування. Було виявлено, що більшість рішень не враховують специфіку замовлення їжі у навчальних закладах, не мають гнучкої системи управління меню та обліку замовлень. На основі цього було визначено вимоги до розроблюваного вебресурсу та сформульовано його основні функціональні можливості.

На етапі проектування було обрано архітектуру вебдодатку, яка ґрунтується на технології Single Page Application (SPA) з використанням React.js для клієнтської частини та Firebase для управління базою даних і автентифікації користувачів. Були створені UML-діаграми, що описують структуру системи та її компоненти, а також ER-модель бази даних, що забезпечує ефективне зберігання та управління інформацією.

Для збереження даних було обрано реляційну базу даних, у якій реалізовано зв'язки між основними сутностями: користувачами, замовленнями, стравами та їх категоріями. Було доведено, що структура бази відповідає Другій нормальній формі (2НФ), що забезпечує цілісність та узгодженість даних.

На етапі програмної реалізації було використано JavaScript у поєднанні з React.js, що дозволило створити динамічний і швидкий інтерфейс. Для зберігання даних і керування користувачами використовувалася Firebase Firestore, що забезпечує надійну взаємодію між компонентами системи. Було розроблено ключові програмні модулі, які відповідають за авторизацію користувачів, перегляд та управління меню, оформлення та обробку замовлень.

На завершальному етапі було проведено тестування вебресурсу, яке включало перевірку функціональності основних модулів. Під час тестування підтверджено, що система працює коректно, без критичних збоїв, а всі функції реалізовані відповідно до поставлених вимог.

Таким чином, у цій роботі було успішно вирішено поставлені задачі, реалізовано вебресурс «School kitchen» та підтверджено його ефективність. Отримане рішення сприяє автоматизації процесу замовлення харчування в школах, підвищенню зручності користувачів та оптимізації роботи шкільних їдалень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Соколова К.А., Ваховська Л.М. Аналіз передумов розробки вебресурсу «School Kitchen». LIV Всеукраїнська науково-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації (2025), Вінниця, 2025. [Електронний ресурс]. – Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2025/paper/view/23684

2. Що таке POS система та як вона працює [Електронний ресурс]. – Режим доступу:
https://magefan.com/ua/blog/shtcho-take-pos-systema

3. Vend POS. URL [Електронний ресурс]. – Режим доступу: https://www.vendhq.com/

4. TouchBistro POS. URL [Електронний ресурс]. – Режим доступу: https://www.touchbistro.com/

5. Toast POS. URL [Електронний ресурс]. – Режим доступу: https://pos.toasttab.com

6. Документація React.js: Офіційна документація бібліотеки React.js [Електронний ресурс]. – Режим доступу: https://reactjs.org/docs

7. Документація JavaScript: Офіційна документація мови програмування JavaScript [Електронний ресурс]. – Режим доступу:
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide

8. Книга "Eloquent JavaScript" автора Маріяна Хавербеке (Marijn Haverbeke) [Електронний ресурс]. – Режим доступу: https://eloquentjavascript.net

9. Stack Overflow [Електронний ресурс]. – Режим доступу: https://stackoverflow.com

10. Firebase Documentation [Електронний ресурс]. – Режим доступу:
https://firebase.google.com/docs

11. Документація HTML і CSS [Електронний ресурс]. – Режим доступу:
https://developer.mozilla.org/en-US/docs/Web/HTML
https://www.w3schools.com

12. Welling, L., & Thomson, L. (2009). "PHP and MySQL Web Development". Pearson Education.

13. Flanagan, D. (2020). "JavaScript: The Definitive Guide". O'Reilly Media.

14. Freeman, E., & Robson, E. (2014). "Head First Design Patterns". O'Reilly Media.

15. Sommerville, I. (2011). "Software Engineering" (9th ed.). Addison-Wesley.

16. Pressman, R. S. (2010). "Software Engineering: A Practitioner's Approach" (7th ed.). McGraw-Hill.

17. MDN Web Docs [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org

18. McFarland, D. (2014). "JavaScript & jQuery: The Missing Manual". O'Reilly Media.

19. Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). "Design Patterns: Elements of Reusable Object-Oriented Software". Addison-Wesley.

20. Visual Studio Code Documentation [Електронний ресурс]. – Режим доступу: https://code.visualstudio.com/docs

21. MySQL 8.0 Reference Manual [Електронний ресурс]. – Режим доступу: https://dev.mysql.com/doc/refman/8.0/en/

22. PostgreSQL Documentation [Електронний ресурс]. – Режим доступу: https://www.postgresql.org/docs/

23. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів спеціальності 122 «Комп’ютерні науки» [Електронний ресурс] / уклад.: А. А. Яровий, О. В. Сілагін, І. В. Богач. – Вінниця : ВНТУ, 2023. – (PDF, 54 с.).

ДОДАТКИ

ДОДАТОК А (ОБОВ'ЯЗКОВИЙ)

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Розробка WEB-ресурсу «School kitchen»

Тип роботи: бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра комп'ютерних наук
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 3,08 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

- ✓ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту
- ☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.
- ☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Яровий А.А., зав. каф. КН _____ (підпис)
(прізвище, ініціали, посада)

Колесницький О.К., проф. каф КН _____ (підпис)
(прізвище, ініціали, посада)

Особа, відповідальна за перевірку _____ Озеранський В.С.
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Ваховська Л.М.
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Соколова К.А.
(підпис) (прізвище, ініціали, посада)

ДОДАТОК Б (ОБОВ’ЯЗКОВИЙ)

ЛІСТИНГ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ ОСНОВНИХ МОДУЛІВ ВЕБДОДАТКУ «SCHOOL KITCHEN»

KitchenPage.js

```
import Header from "./components/KitchenPage/Header";
import MenuBox from "./components/KitchenPage/MenuBox";
import OrdersBox from "./components/KitchenPage/OrdersBox";
import ModalMenu from "./components/KitchenPage/ModalMenu";
import {useState} from "react";
import {useEffect} from "react";

function KitchenPage() {

  const [modal, setModal] = useState({
    modal1:false
  })

  const [menuArr, setMenuArr] = useState([]);

  const sendMenuToFirebase = async (menu) => {
    try {
      const response = await fetch(
        "https://school-kitchen-71370-default-rtdb.firebaseio.com/menu.json",
        {
          method: "POST",
          body: JSON.stringify(menu),
        }
      );
      const data = await response.json();
      const menuArray = transformFbDataToArr(data);
      renderMenuItems(menuArray);
      fetchMenu();
```

```
    } catch (error) {
        console.error(error);
    }
};

const [menuData, setMenuData] = useState([]);
const [isLoading, setIsLoading] = useState(false);

useEffect(() => {
    fetchMenu();
}, []);

const fetchMenu = async () => {
    setIsLoading(true);

    try {
        const response = await fetch(
            "https://school-kitchen-71370-default-rtdb.firebaseio.com/menu.json"
        );
        const data = await response.json();
        const transformedData = transformFbDataToArr(data);
        setIsLoading(false);
        setMenuData(transformedData);
        setMenuArr(transformedData);
        renderMenuItems(menuArr)
    } catch (error) {
        console.log(error);
    }
};

const transformFbDataToArr = (fbData) => {
    if (fbData) {
        return Object.keys(fbData).map((key) => {
            const item = fbData[key];
            item.id = key;
```

```
            return item;
          });
        } else {
          return [];
        }
      };

      function renderMenuItems() {
        return menuArr.map((item) => (
            <li key={item.id} className="panel__menu__item">
              <div className="panel__menu__item-box">
                <p className="panel__menu__item-name">{item.name}</p>
                <p className="panel__menu__item-price">Ціна: {item.price} UAH</p>
              </div>
              <button className="btn-dlt" onClick={() =>
deleteMenuItem(item.id)}>Видалити</button>
            </li>
          )
        );
      }

      const deleteMenuItem = async (itemId) => {
        try {
          // Видалення об'єкту з бази даних
          await fetch(
            `https://school-kitchen-71370-default-rtdb.firebaseio.com/menu/${itemId}.json`,
            {
              method: "DELETE",
            }
          );

          // Оновлення списку елементів
          const updatedMenuArr = menuArr.filter((item) => item.id !== itemId);
          setMenuArr(updatedMenuArr);
          renderMenuItems(updatedMenuArr);
```

```
    } catch (error) {
      console.error(error);
    }
  };

  return (
    <div className="Panel">
      <Header/>
      <button className="app__btn app__btn-panel" onClick={() => setModal({
        ...modal, modal1: true
      })}>
        <div className="app__decor">
          <img src="/images/arrow-right.svg" alt=""/>
        </div>
        <span>Додати товар в меню</span>
      </button>
      <section className="panel">
        <div className="container">
          <div className="panel__inner">
            <MenuBox renderMenuItems={renderMenuItems}/>
             <OrdersBox/>
          </div>
        </div>
      </section>
      <ModalMenu isOpened={modal.modal1}
                 onModalClose={() => setModal({
                   ...modal, modal1: false
                 })}
      sendMenuToFirebase={sendMenuToFirebase}/>
    </div>
  );
}

export default KitchenPage;
```

UserPage.js

```
import Header from './components/UserPage/Header';
import Notification from "./components/UserPage/Notification";
import Tray from "./components/UserPage/Tray";
import FirstModal from "./components/UserPage/FirstModal";
import Modal from "./components/UserPage/Modal";
import ModalDone from "./components/UserPage/ModalDone";
import ModalSendOrder from "./components/UserPage/ModalSendOrder";
import ModalOrders from "./components/UserPage/ModalOrders";
import {useState} from "react";

function UserPage() {

  const [modal, setModal] = useState({
    modal1:false,
    modal2:false
  })

  const [trayArr, setTrayArr] = useState([]);

  const calcPrice = (arr) => {
    let totalPrice = 0;
    arr.forEach((item) => {
      totalPrice += +item.price * item.count;
    });
    return totalPrice;
  };

  const renderFoodsList = (arr) => {
    if (arr.length === 0) {
      return (
        <div className="form__list-text" style={{ display: 'block' }}></div>
      );
```

```
    } else {
      return (
        <ul className="form__list">
          {arr.map((item) => (
            <li key={item.id} data-id={item.id}>
              <div className="price">
                <span>{`${item.name} x ${item.count}`}</span>
                <div>
                  <span>{item.price * item.count}</span>
                  <span>UAH</span>
                </div>
              </div>
            </li>
          ))}
        </ul>
      );
    }
};

return (
  <div className="App">
    <div className="wrapper">
      <Notification/>
      <Header/>
      <div className="app">
        <div className="container">
          <Tray
            trayArr={trayArr}
            setTrayArr={setTrayArr}
            calcPrice={calcPrice}
            renderFoodsList={renderFoodsList}/>
          <div className="app__btn-box animate__animated animate__fadeInUp">
            <button className="app__btn" onClick={() => setModal({
              ...modal, modal1: true
            })}>
```

```
                        <div className="app__decor">
                            <img src="/images/arrow-right.svg" alt=""/>
                        </div>
                        <span>Зробити замовлення</span>
                    </button>
                </div>
            </div>
        </div>
        <FirstModal/>
        <Modal isOpened={modal.modal1}
               onModalClose={() => setModal({
                  ...modal, modal1: false
               })}
               trayArr={trayArr}
               calcPrice={calcPrice}
               renderFoodsList={renderFoodsList}/>
        <ModalDone/>
        <ModalSendOrder/>
        <ModalOrders isOpened={modal.modal2}
                     onModalClose={() => setModal({
                        ...modal, modal2: false
                     })}/>
        <button className="app__orders-list-btn animate__animated" onClick={() => setModal({
           ...modal, modal2: true
        })}>
           <img className="app__orders-list-img" src="/images/ordersList.svg" alt="img"/>
        </button>
      </div>
   );
}
export default UserPage;
```

ДОДАТОК В (ОБОВ'ЯЗКОВИЙ)

ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА ВЕБРЕСУРСУ «SCHOOL KITCHEN»

Виконала: студентка 4 курсу, групи ЗКН-21б
спеціальності 122 Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Соколова К.А.
(прізвище та ініціали)
Керівник: асистент каф. КН
Ваховська Л.М.
(прізвище та ініціали)
«03» червня 2025 р.

Вінниця ВНТУ – 2025 рік

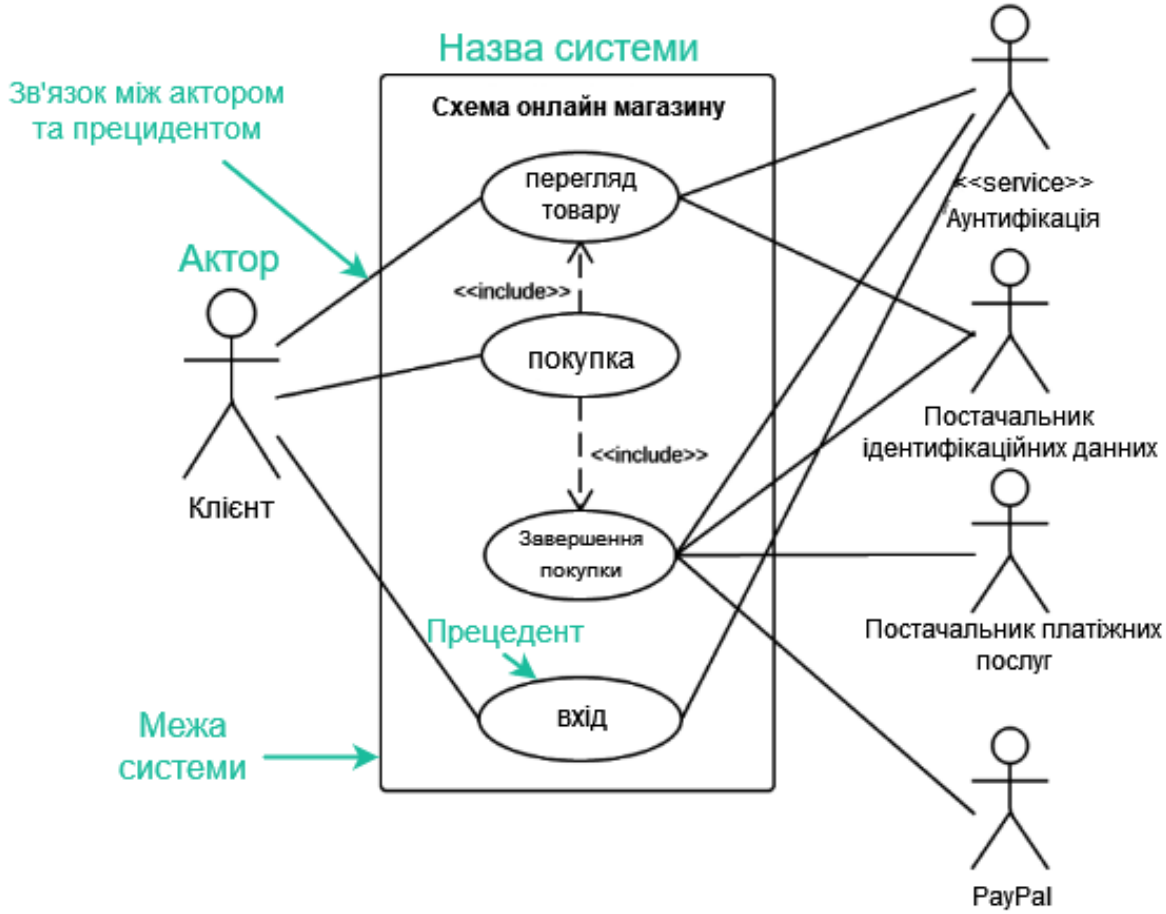

Рисунок В.1 – «Usecase» – діаграма вебдодатку

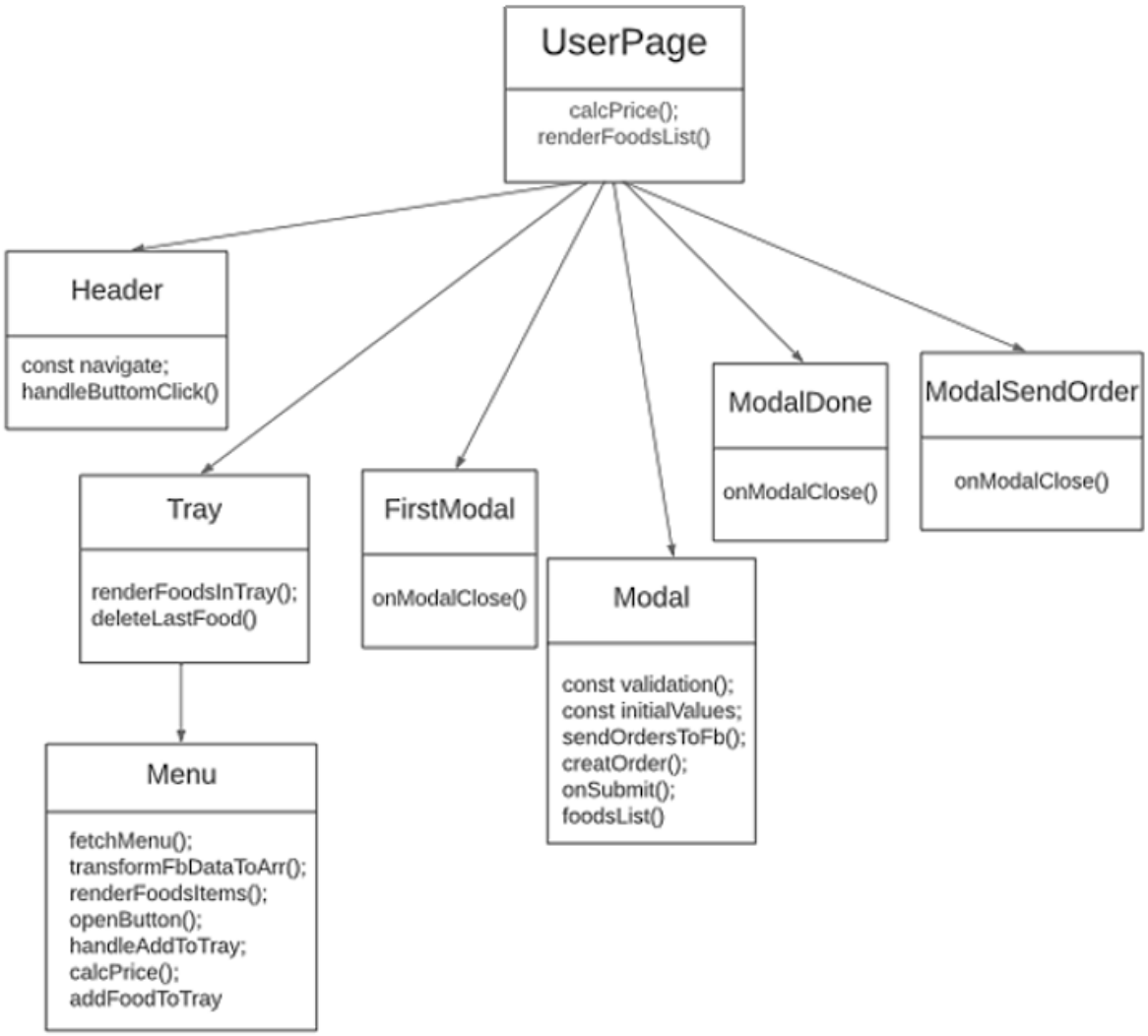

Рисунок В.2 – Діаграма класів сторінки користувачів вебресурсу

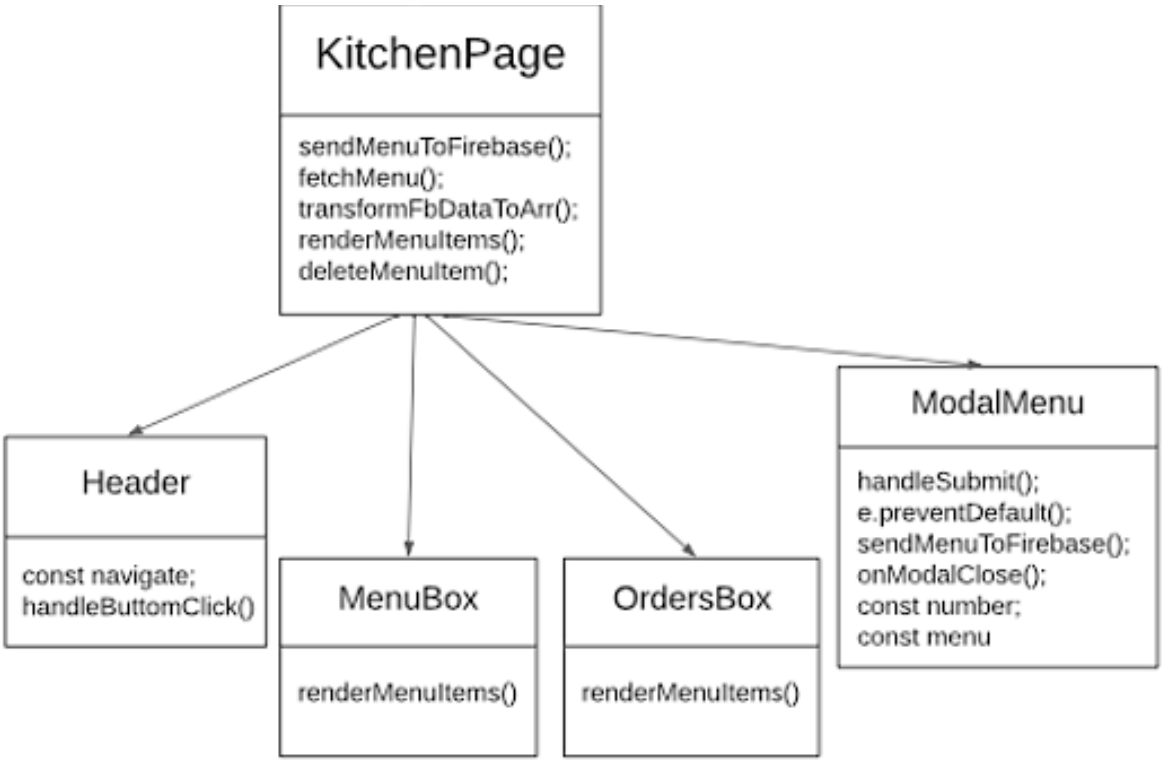

Рисунок В.3 – Діаграма класів сторінки адміністрації їдальні вебресурсу

Рисунок В.4 – Діаграма варіантів використання вебдодатку

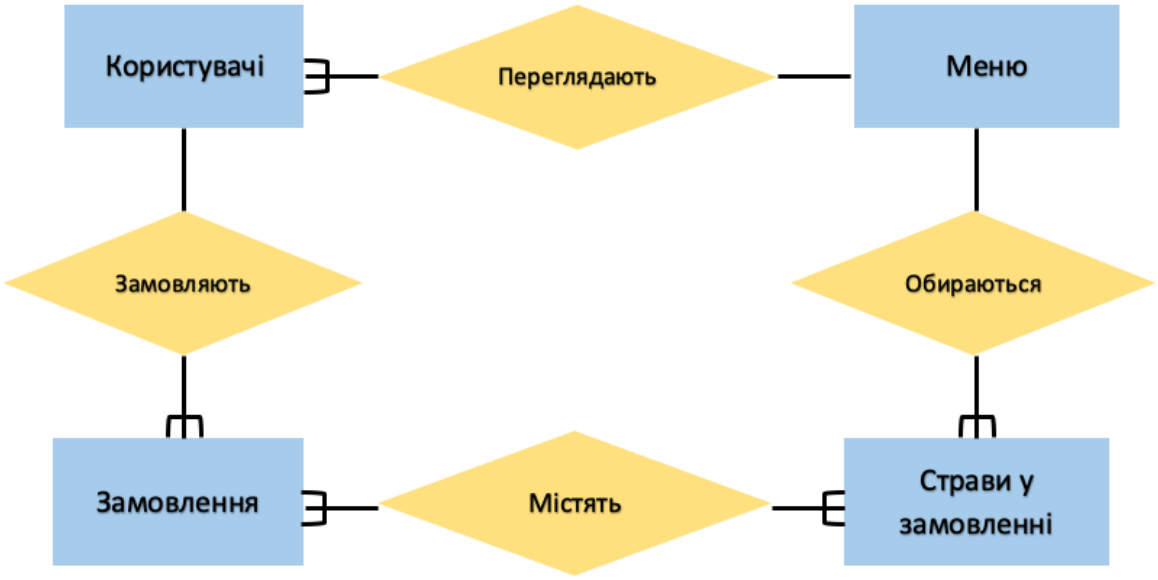

Рисунок В.5 – ER- модель бази даних вебресурсу

Рисунок В.6 – Блок-схема алгоритму роботи вебресурсу «School kitchen»

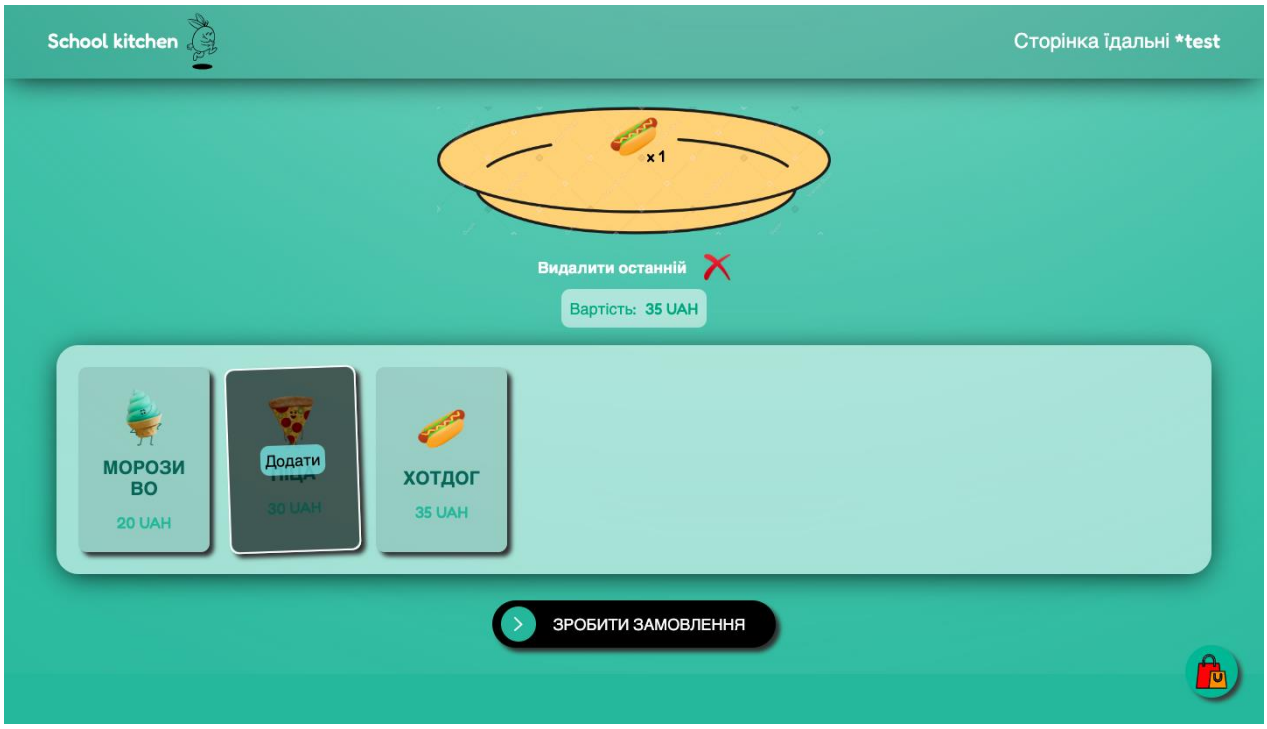

Рисунок В.7 – Вибір товару

ДОДАТОК Г (ДОВІДНИКОВИЙ)
ІНСТРУКЦІЯ КОРИСТУВАЧА ВЕБДОДАТКУ «SCHOOL KITCHEN»

Додатком можна користатися у будь-якому браузері, ввівши в пошук цю URL-адресу https://kateryna-papa.github.io/.

Для користувачів, які вперше відвідали сайт, відкриється відео з інструкцією додавання страв в тарілку. Після перегляду цього відео, можна вибирати страви, як показана на рисунку Г.1, натиснувши кнопку «Зробити замовлення», оформити замовлення ввівши усі потрібні дані у форму (рис.Г.2), яка відкриється, та натиснувши кнопку «Відправити».

Також на сторінці є кнопка для перегляду своїх замовлень, яка знаходиться знизу з правого боку екрану. Приклад зображено на рисунку Г.3.

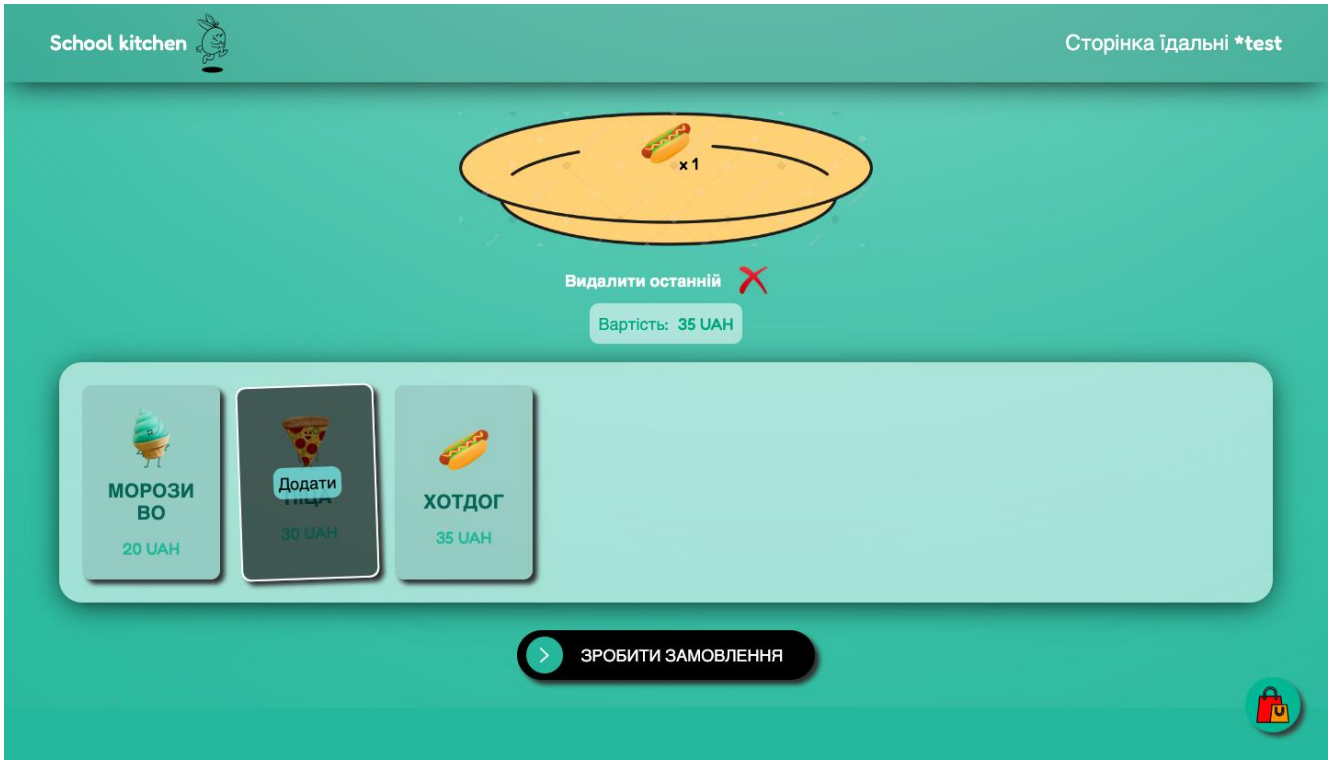

Рисунок Г.1 – Вибір товару

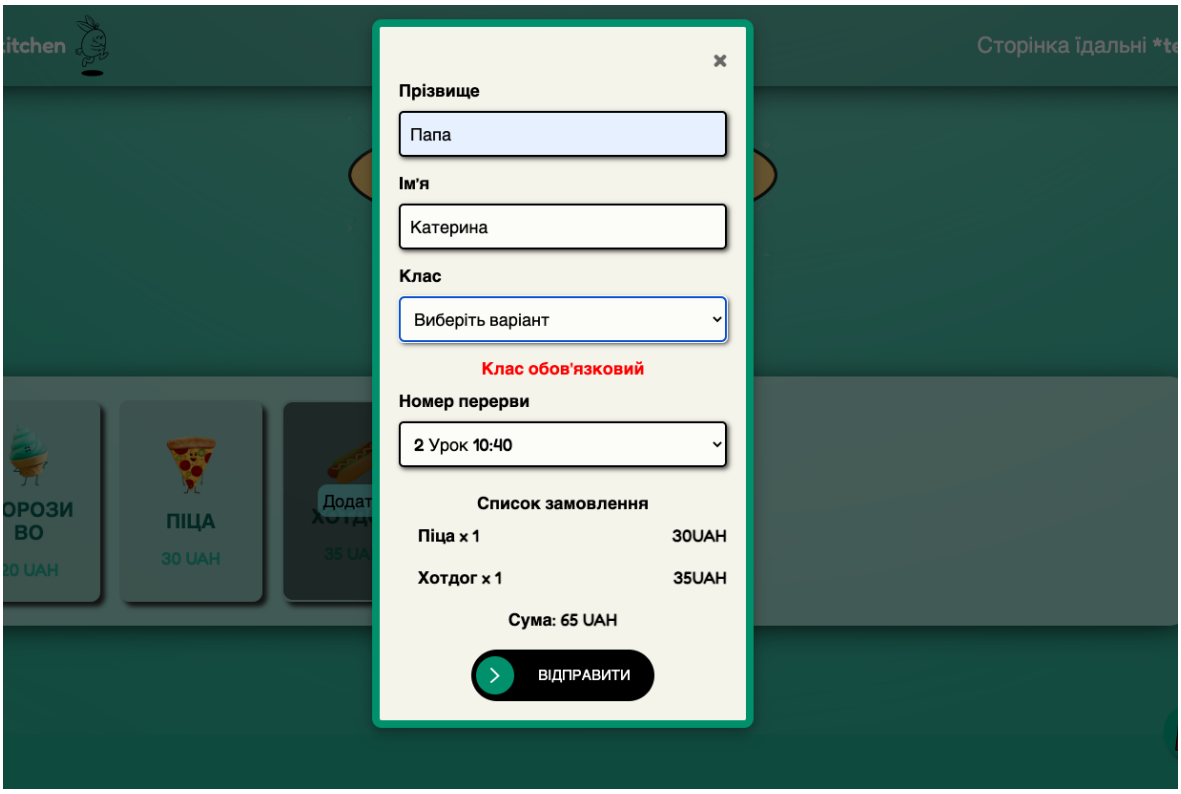

Рисунок Г.2 – Оформлення замовлення

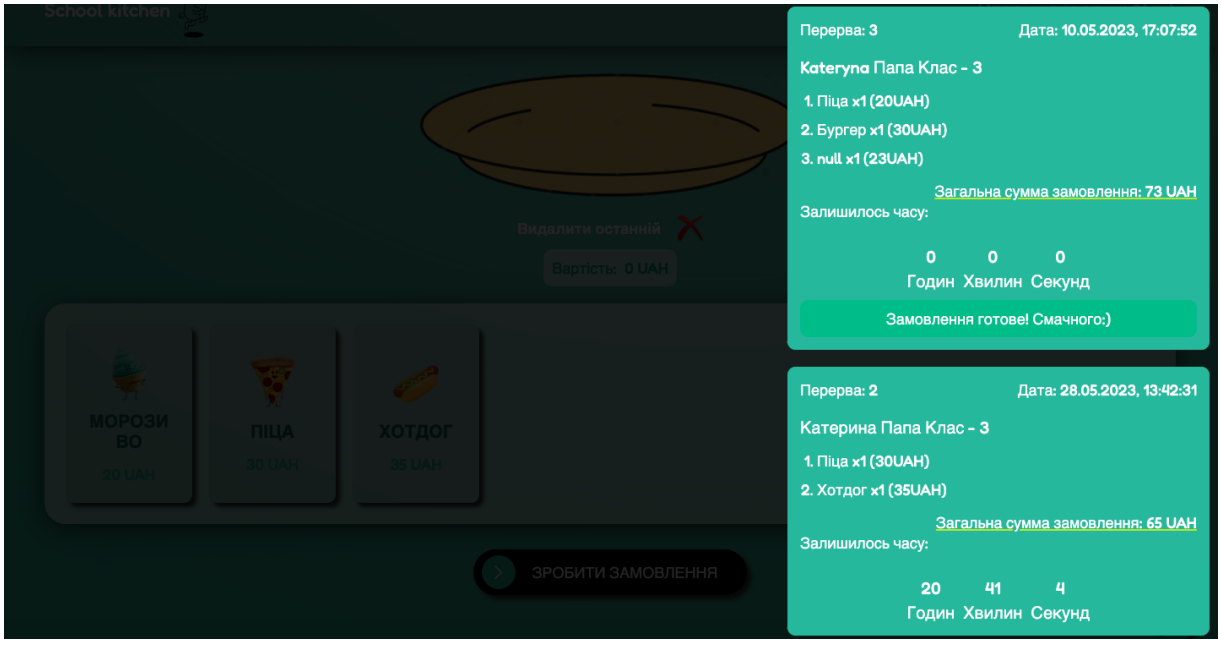

Рисунок Г.3 – Перегляд замовлень на сторінці користувача

З боку адміністратора на сторінці кухні є кнопка «Додати товар в меню», якою потрібно скористатися для формування меню (рис.Г.4). Також можна

переглядати списки меню та замовлень, які відразу відображаються на екрані. Приклад панелі адміністрації кухні зображений на рисунку Г.5.

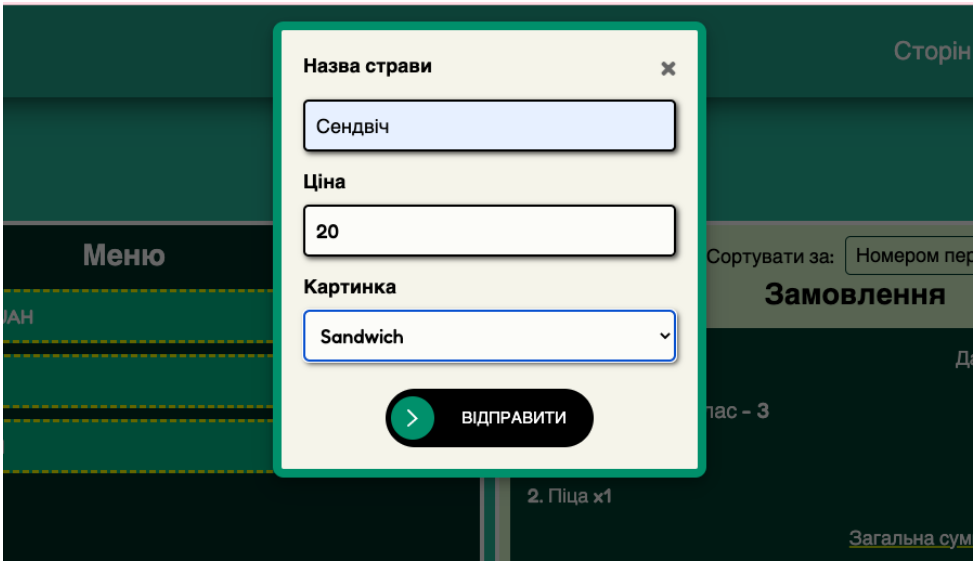

Рисунок Г.4 – Додавання страв в меню

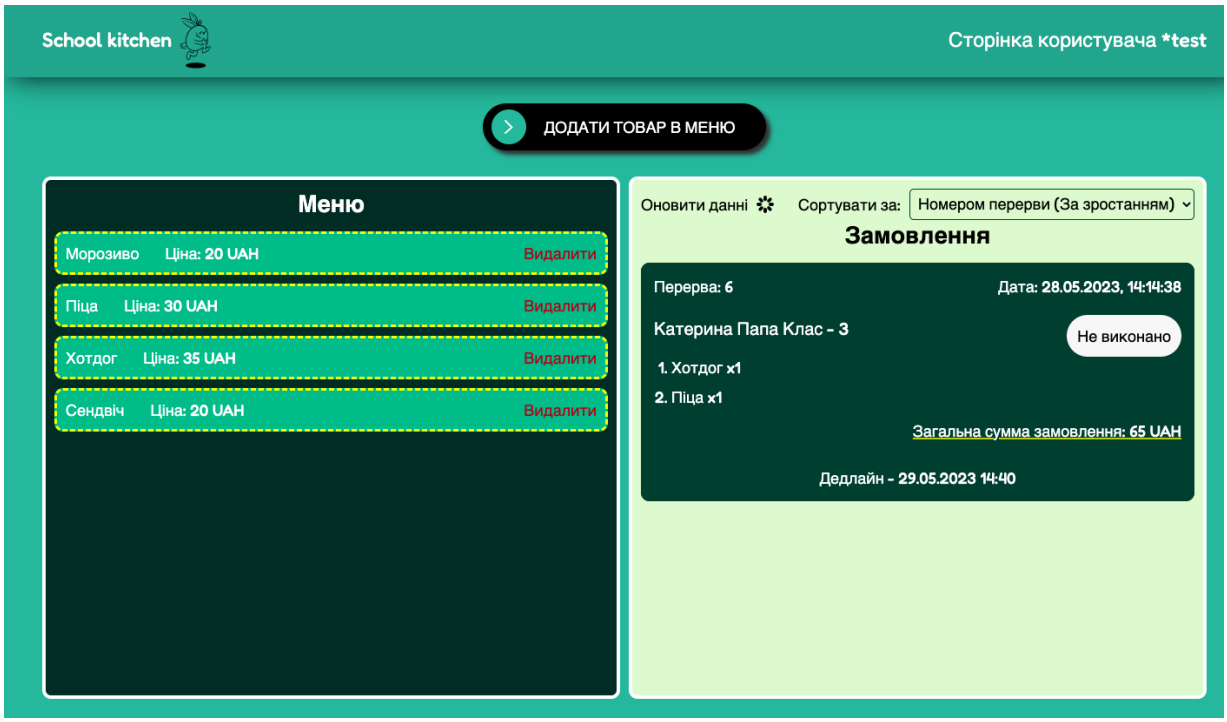

Рисунок Г.5 – Перегляд списків меню та замовлень