Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерної інженерії та інформаційних систем

КВАЛІФІКАЦІЙНА РОБОТА

бакалавр
Освітній рівень

Програмно-технічний комплекс для автоматизованого прийому та оповіщення про замовлення в ресторані східної кухні на базі мікроконтролера ESP32
Назва теми

КвРКІ 210367.21.03.38 ПЗ
Шифр

Галузь знань *12 «Інформаційні технології»*
Шифр, назва

Спеціальність *123 «Комп'ютерна інженерія»*
Шифр, назва

Освітня програма *«Комп'ютерна інженерія та програмування»*
Назва

Виконав: студент IV курсу, група КІ2-21-3 _____ Юрій МАРЦЕНЮК
Підпис Ініціали, прізвище

Керівник _____ Тетяна КИСІЛЬ
Підпис, дата Ініціали, прізвище

Нормоконтролер _____ Тетяна КИСІЛЬ
Підпис, дата Ініціали, прізвище

До захисту допускаю:
зав. кафедри комп'ютерної інженерії та інформаційних систем _____ Ольга ПАВЛОВА
Підпис Ініціали, прізвище

«12» червня 2025 р.

Хмельницький 2025

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

Освітній рівень БАКАЛАВР

Галузь знань 12 ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

Спеціальність 123 КОМП'ЮТЕРНА ІНЖЕНЕРІЯ

Освітня програма «КОМП'ЮТЕРНА ІНЖЕНЕРІЯ ТА ПРОГРАМУВАННЯ»

ЗАТВЕРДЖУЮ

Зав. кафедри Ольга ПАВЛОВА

" 10 " 01 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

Юрію МАРЦЕНЮКУ

Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Програмно-технічний комплекс для автоматизованого прийому та оповіщення про замовлення в ресторані східної кухні на базі мікроконтролера ESP32

Керівник проекту (роботи) Тетяна КИСІЛЬ, доцент кафедри КІІС.

Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 07.02.2025 р. № 23

2. Строк подання студентом проекту (роботи) на кафедру 01.06.2025 р.

3. Вихідні дані до проекту (роботи) Завдання на кваліфікаційну роботу

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Програмно-технічний комплекс для автоматизованого прийому та оповіщення про замовлення в ресторані східної кухні на базі мікроконтролера ESP32

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

Діаграма бази даних

Структура компонентів бекенду

Функціональна схема інтеграції головних компонентів апаратної частини

6. Консультанти розділів дипломного проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Тетяна КИСІЛЬ, доцент кафедри КІІС		
Антиплагіат	Андрій НІЧЕПОРУК, доцент кафедри КІІС		

7. Дата видачі завдання «_10_»_____01_____2025 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1	Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи з керівником	10.01.2025	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	01.02.2025	виконано
3	Робота над розділом 1 – дослідження предметної області та постановка задачі	01.03.2025	виконано
4	Робота над розділом 2 – проєктування програмно-технічного засобу	01.04.2025	виконано
5	Робота над розділом 3 – реалізація програмно-технічного засобу	29.04.2025	виконано
6	Оформлення пояснювальної записки згідно вимог	25.05.2025	виконано
7	Попередній захист ВКР	26.05.2025	виконано
8	Захист ВКР на засіданні ЕК	Червень 2025 року	

Студент _____ Підпис _____ Юрій МАРЦЕНЮК Ініціали, прізвище

Керівник роботи _____ Підпис _____ Тетяна КИСІЛЬ Ініціали, прізвище

№ рядка	формат	Позначення	Найменування	Кіл. листів	№ екз	Примітка
			Текстові документи			
1		КвРКІ 210367.21.03.38 ПЗ	Пояснювальна записка	74		
			Графічні матеріали			
2		КвРКІ 210367.21.03.38 Е8	Діаграма бази даних	1		
3		КвРКІ 210367.21.03.38 Е8	Структура компонентів бекенду	1		
4		КвРКІ 210367.21.03.38 Е8	Функціональна схема інтеграції головних компонентів апаратної частини	1		

КвРКІ 210367.21.03.38 ВП

Зм	Арк	№ докум	Підпис	Дата	Відомість проекту	Літера	Аркуш	Аркушів
Розробив		Марценюк		12.06.25		У	1	74
Перевір.		Кисіль		12.06.25				
Н. контр.		Кисіль		12.06.25		ХНУ, КІ2-21-3		
Затв.		Павлова		12.06.25				

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Програмно-технічний комплекс для автоматизованого прийому та оповіщення про замовлення в ресторані східної кухні на базі мікроконтролера ESP32».

Автор роботи: Юрій МАРЦЕНЮК.

Керівник роботи: Кисіль Тетяна Миколаївна.

Пояснювальна записка: 74 с., 22 рис., 1 табл., 3 дод., 50 джерел.

Графічна частина: 3 креслення.

АВТОМАТИЗАЦІЯ, КІБЕРФІЗИЧНА СИСТЕМА, Архітектура, МОНІТОРИНГ, БАЗА ДАНИХ, МІКРОКОНТРОЛЕР.

Метою дипломної роботи є визначення особиливостей та реалізація програмно-апаратного комплексу в діджиталізації фізичного ресторанного бізнесу.

Об'єктом дослідження є впровадження цифрової системи з апаратним пристроєм у ресторанний бізнес.

Предметом дослідження є оцінка ефективності діджиталізації процесів ресторанного бізнесу.

Під час проведення даного дослідження був використаний метод систематичного огляду літератури для вивчення і аналізу предметної області даного дослідження з текстових джерел інформації.

Підпис студента

30.05.2025

Дата

ЗМІСТ

ВСТУП

Актуальність дослідження. Кіберфізичні системи заповнюють практично всі сфери нашого життя. Це і розумні будинки, розумні виробництва та мережі, безпілотний транспорт та транспортні мережі і навіть розумні міста та ін. У даному дослідженні ми розглянемо один із типів кіберфізичних систем.

Проєкт, присвячений розробці вебсайту для замовлення їжі в ресторані східної кухні в поєднанні з фізичним пристроєм на базі мікроконтролера ESP32 і сенсорного екрана, орієнтований на створення інтегрованої системи для оптимізації робочих процесів закладу харчування. Дана робота з аналізом предметної області визначить головні проблеми, завдання та можливості покращення в цій сфері та реалізує це.

Сучасний ресторанний бізнес характеризується зростаючими вимогами до швидкості обслуговування, точності виконання замовлень та ефективності внутрішніх операційних процесів. Традиційні методи управління замовленнями, що базуються на паперових записах та вербальній комунікації між персоналом, демонструють суттєві обмеження в умовах високого навантаження та складності сучасного меню ресторанів східної кухні. Особливості азійської гастрономії, що включають різноманітні способи приготування, специфічні інгредієнти та складні комбінації страв, вимагають особливо ретельного підходу до передачі інформації між залом та кухнею.

Цифрова трансформація галузі громадського харчування набуває все більшого значення в контексті підвищення конкурентоспроможності закладів та забезпечення якісного сервісу для споживачів. Інтеграція інформаційних технологій у ресторанні процеси дозволяє не лише автоматизувати рутинні операції, але й створити нові можливості для аналізу продуктивності, оптимізації ресурсів та підвищення задоволеності клієнтів. Особливої актуальності набувають рішення, що поєднують веб-технології для взаємодії з клієнтами та embedded

системи для управління внутрішніми процесами, створюючи цілісну екосистему цифрового ресторану.

Мікроконтролери серії ESP32 представляють собою оптимальне рішення для створення IoT пристроїв у ресторанному середовищі завдяки поєднанню потужних обчислювальних можливостей, інтегрованих модулів бездротової комунікації та відносно низької вартості. Їх застосування в системах управління замовленнями дозволяє реалізувати розподілену архітектуру, де кожен кухонний пост може бути обладнаний автономним пристроєм для відображення та управління релевантною інформацією про замовлення. Така децентралізована структура забезпечує надійність системи та можливість масштабування відповідно до потреб конкретного закладу.

Специфіка ресторанів східної кухні вимагає особливого підходу до організації кухонних процесів через складність та різноманітність технологій приготування страв. Традиційні азійські кухні характеризуються використанням множини спеціалізованого обладнання, різних температурних режимів та точного дотримання часових інтервалів приготування. Ефективна координація цих процесів потребує чіткої та своєчасної передачі інформації про замовлення, що робить актуальним впровадження автоматизованих систем оповіщення та контролю.

Сучасні веб-технології, зокрема фреймворк NestJS та екосистема TypeScript, надають потужні інструменти для створення масштабованих серверних додатків, здатних обробляти значні обсяги даних про замовлення в режимі реального часу. Використання контейнерних технологій, таких як Docker, забезпечує гнучкість розгортання та управління серверною інфраструктурою, що особливо важливо для ресторанних мереж з множиною локацій. Інтеграція систем моніторингу та аналітики, реалізована засобами Grafana, створює можливості для глибокого аналізу операційної ефективності та виявлення вузьких місць у робочих процесах.

Проблематика забезпечення надійної та швидкої комунікації між веб-інтерфейсом прийому замовлень та фізичними пристроями на кухні вимагає

застосування сучасних протоколів передачі даних та архітектурних рішень. WebSocket технологія забезпечує двосторонню комунікацію в режимі реального часу, що критично важливо для своєчасного оповіщення персоналу кухні про нові замовлення та зміни їх статусів. Поєднання WebSocket для миттєвих повідомлень та HTTP для структурованого обміну даними створює оптимальний баланс між швидкістю відгуку системи та надійністю передачі інформації.

Ергономічні аспекти проектування користувацьких інтерфейсів для кухонного персоналу набувають особливого значення в контексті специфічних умов експлуатації ресторанного обладнання. Високі температури, підвищена вологість, необхідність швидкого прийняття рішень та часта робота у рукавичках вимагають створення інтуїтивних та стійких до зовнішніх впливів інтерфейсів. Використання сенсорних дисплеїв Waveshare з SPI інтерфейсом забезпечує необхідну швидкодію та надійність взаємодії користувача з системою в складних умовах кухонного середовища.

1 ТЕОРЕТИЧНІ ОСНОВИ ДОСЛІДЖУВАНОЇ ПРОБЛЕМИ

1.1 Аналіз предметної області і виявлення наявних проблем і завдань

Проєкт, присвячений розробленню вебсайту для замовлення їжі в ресторані східної кухні в поєднанні з фізичним пристроєм на базі мікроконтролера ESP32 і сенсорного резистивного TFT екрану, орієнтований на створення інтегрованої системи для оптимізації робочих процесів закладу харчування. Даний аналіз предметної області визначить головні проблеми, завдання та можливості покращення в цій сфері.

Сучасна індустрія громадського харчування характеризується інтенсивною конкуренцією та постійно зростаючими очікуваннями споживачів щодо якості сервісу та швидкості обслуговування. Специфіка ресторанів східної кухні додає додаткові виклики через складність меню, різноманітність способів приготування страв та необхідність точного дотримання автентичних рецептур. Аналіз поточного стану ринку демонструє критичну потребу в комплексних технологічних рішеннях, здатних забезпечити ефективну координацію між операціями прийому замовлень та процесами їх виконання.

Традиційна модель роботи ресторанів східної кухні базується на передачі інформації від офіціанта до кухні через паперові бланки або вербальну комунікацію. Такий підхід демонструє суттєві недоліки в умовах високого навантаження, створюючи передумови для помилок у передачі складних назв страв та специфічних побажань клієнтів.

Аналіз робочих процесів виявляє кілька критичних проблем. Першою є складність координації між різними кухонними постами, оскільки приготування замовлення часто вимагає синхронізації роботи холодних закусок, гарячих страв, wok-станції та суші-бару. Відсутність централізованої системи моніторингу призводить до ситуацій неодночасної готовності компонентів замовлення.

Другою проблемою є неефективність комунікації щодо змін у замовленнях або додаткових побажань клієнтів. Передача такої інформації через усні повідомлення створює ризики недопорозумінь, особливо при обслуговуванні великих компаній.

Третім проблемним аспектом є відсутність системи аналітики кухонних процесів. Без відстеження часу виконання замовлень та аналізу навантаження менеджмент не має достатньої інформації для оптимізації роботи закладу.

Специфічні виклики ресторанів східної кухні включають точне відображення складних назв страв, управління великою кількістю інгредієнтів та врахування культурних особливостей подачі. Традиційні POS системи часто не адаптовані до таких вимог, що створює додаткові труднощі для персоналу.

Дослідження поведінки споживачів показує зростаючу популярність онлайн замовлень та очікування щодо відстеження статусу в режимі реального часу. Клієнти ресторанів східної кухні особливо цінують можливість детального ознайомлення з інгредієнтами та кастомізації відповідно до індивідуальних переваг.

Технічний аналіз наявних рішень виявляє фрагментарність підходів до автоматизації ресторанних процесів. Більшість систем зосереджуються або на клієнтських інтерфейсах, або на управлінських системах, рідко забезпечуючи повну інтеграцію. Відсутність спеціалізованих рішень для кухонного персоналу створює значний розрив між можливостями технологій та реальними потребами галузі.

1.2 Сучасний стан замовлень у ресторанах

У ресторанному бізнесі системи управління замовленнями формують фундамент для забезпечення злагодженої роботи між клієнтами, залом обслуговування та кухнею. Традиційно такі системи передбачають використання

паперових замовлень або стандартного програмного забезпечення для прийому та обробки замовлень.

Однак у ресторанах східної кухні, зокрема, існують специфічні виклики:

– велика кількість складних страв із різними інгредієнтами потребує чіткої координації між кухнею та залом;

– висока швидкість обслуговування клієнтів є критичною, адже це безпосередньо впливає на рівень задоволеності клієнтів;

– точність передачі замовлень є важливою, адже помилки можуть призвести до втрат часу, інгредієнтів та клієнтів.

Використання мікроконтролера ESP32 у закладах харчування.

Мікроконтролери ESP32 завдяки своїй багатофункціональності широко застосовуються для створення IoT-рішень у сфері ресторанного бізнесу. Сенсорні екрани, підключені до таких пристроїв, забезпечують зручний інтерфейс для взаємодії між кухнею та системою управління замовленнями.

Ваш пристрій, що відображає активні замовлення на кухні та дозволяє змінювати їхній статус, має суттєвий потенціал для вдосконалення операційних процесів:

– виключення паперових носіїв для замовлень зменшує ризик помилок і втрат інформації;

– автоматизація зміни статусів замовлень підвищує ефективність і прозорість роботи кухні;

– інтеграція із вебсайтом дозволяє створити єдину систему для управління всіма аспектами процесу замовлення їжі.

1.3 Виявлення наявних проблем

Проблеми у замовленнях через вебсайт:

– відсутність інтеграції між вебсайтом і внутрішніми процесами кухні може спричинити затримки в обробці замовлень;

– клієнтський інтерфейс може бути недостатньо інтуїтивним, що може знижувати рівень задоволеності клієнтів;

– можливість помилок при передачі замовлень від вебсайту до кухні через людський фактор.

Індустрія активно використовує різні методи привернення уваги споживачів: унікальний дизайн, маскоти, оригінальні страви. Важливу роль відіграє і реалізація цих підходів у веб-застосунках.

Веб-дизайн безпосередньо впливає на користувацьку поведінку, сприйняття бренду та конверсію. Візуальні елементи формують перше враження, а ефективна навігація забезпечує комфортне користування. Коли відвідувач легко знаходить потрібну інформацію, підвищується ймовірність виконання цільових дій.

Удосконалення UX/UI-дизайну покращує користувацький досвід, скорочуючи час пошуку важливих елементів. Помітні кнопки заклику до дії, структурована інформація та продумана кольорова гама привертають увагу до ключових елементів. Якщо важливі кнопки губляться серед інших елементів, користувач може їх проігнорувати, що знижує конверсію.

Адаптивність дизайну та швидкість завантаження сайту критично важливі для утримання користувачів. Більшість замовлень здійснюється з мобільних пристроїв, тому сайт має бути оптимізований для смартфонів. Зручний пошук, доступ до кошика та спрощене оформлення замовлення забезпечують позитивний досвід.

Вдосконалення дизайну – це стратегічне бізнес-рішення, що впливає на клієнтську базу та прибутковість. Професійний UX/UI-дизайн знижує показники відмов, зміцнює довіру до бренду та заохочує клієнтів повертатися.

Зробимо аналіз сервісів онлайн-замовлення трьох тернопільських локальних ресторанів, а саме:

1. Guru Food;
2. Kilogramm;
3. Sushi Zoom.

Надамо оцінку за десяти-бальною шкалою за такими критеріями:

1. UI/UX-дизайн сайту (будуть наведені скриншоти, на основі яких проводитиметься аналіз);
2. мета-теги для пошуку в пошуковиках (для аналізу мета-тегів буде задіяно онлайн-сервіс SEOMATOR, який витягує з HTML мета-теги: Free Meta Tags Checker: Extract Meta Tags From Any Website);
3. шлях до оформлення замовлення (аналіз складності шляху).

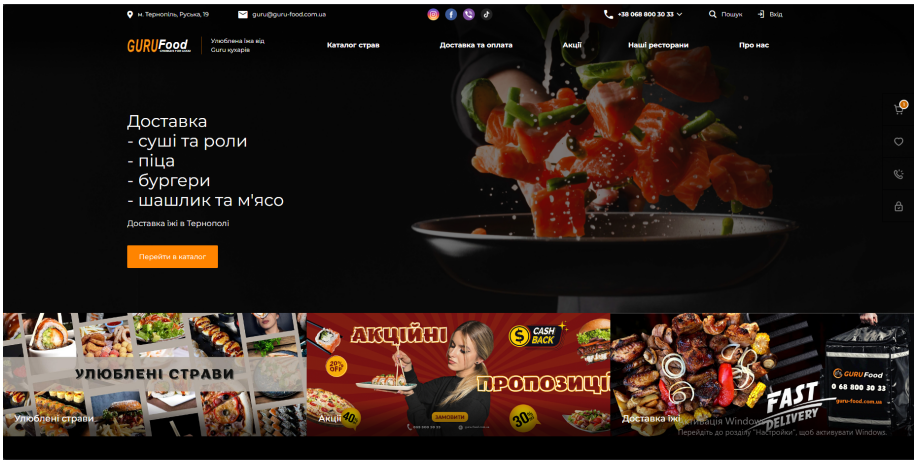


Рисунок 1.1 – Hero секція сайту GuruFood

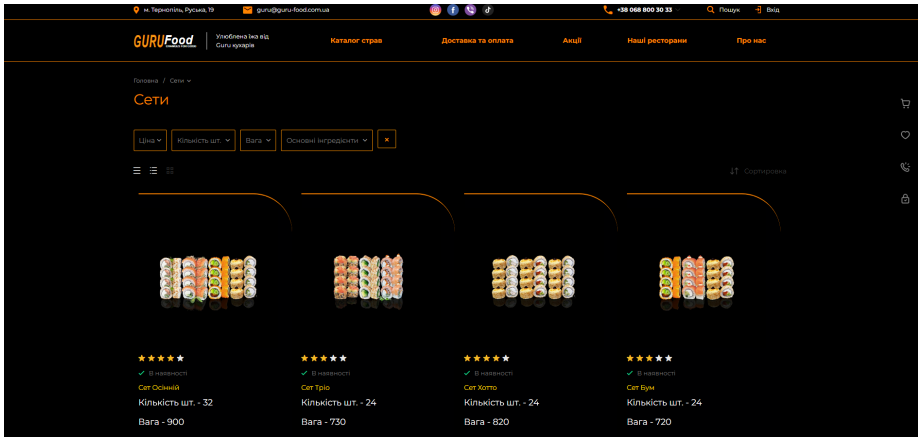
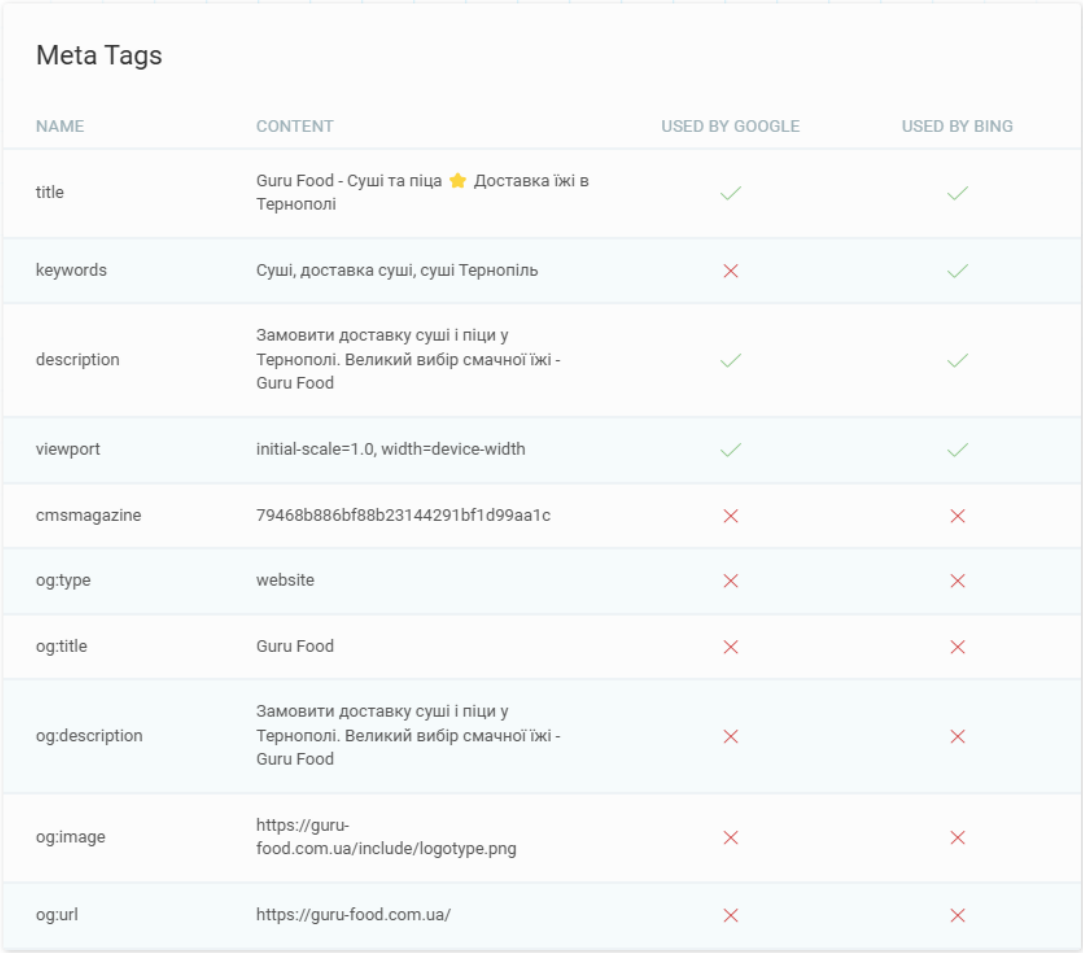


Рисунок 1.2 – Секція каталогу в категорії “Сети”

Дизайн веб-сайту викликає змішані враження: з одного боку, він містить усі необхідні елементи для зручної навігації та взаємодії користувача, проте з іншого – перевантаженість деталями, невдале розташування блоків і брак чіткої візуальної

ієрархії створюють певний хаос у сприйнятті. Важливо, щоб веб-дизайн не лише естетично виглядав, а й забезпечував комфортне та інтуїтивне користування, і наразі у цьому аспекті є низка недопрацювань.

Аналіз мета-тегів



Meta Tags

NAME	CONTENT	USED BY GOOGLE	USED BY BING
title	Guru Food - Суші та піца ⭐ Доставка їжі в Тернополі	✓	✓
keywords	Суші, доставка суші, суші Тернопіль	✗	✓
description	Замовити доставку суші і піци у Тернополі. Великий вибір смачної їжі - Guru Food	✓	✓
viewport	initial-scale=1.0, width=device-width	✓	✓
cmsmagazine	79468b886bf88b23144291bf1d99aa1c	✗	✗
og:type	website	✗	✗
og:title	Guru Food	✗	✗
og:description	Замовити доставку суші і піци у Тернополі. Великий вибір смачної їжі - Guru Food	✗	✗
og:image	https://guru-food.com.ua/include/logotype.png	✗	✗
og:url	https://guru-food.com.ua/	✗	✗

Рисунок 1.3 – Результати аналізу мета-тегів

Головний екран: слоган "Доставка їжі в Тернополі" губиться серед тексту. Перелік категорій займає забагато місця. CTA-кнопка недостатньо помітна.

Навігація: подвійний “хедер” розсіює увагу. Контакти розміщені нелогічно. Пошук потребує централізації. Номер телефону має бути доступнішим.

Структура: сайт перевантажений текстом. Функціональні блоки погано виділені. Ключові елементи треба розмістити у верхній частині сторінки. Блоки зливаються, створюючи візуальний хаос.

Типографіка: назви страв потребують збільшення. Візуальна ієрархія відсутня. Текст потребує кращого структурування.

Оцінка UI/UX-дизайну: 5/10 (потребує серйозних змін для покращення зручності користування).

За даними вихідних даних від SEOMATOR (рис. 1.3), нам відомо, що:

1. і Google, і Bing розпізнають:
   – Title tag ("Guru Food - Суші та піца Доставка їжі в Тернополі");
   – тег опису (про доставку їжі по Тернополю);
   – налаштування вікна перегляду.
2. лише пошуковик Bing розпізнає ключові теги ("Суші, доставка суші, суші Тернопіль");
3. жодна з пошукових систем не розпізнає:
   – ідентифікатор журналу CMS;
   – будь-який з тегів Open Graph (og:), які важливі для публікації в соціальних мережах.

Отже, при пошуку веб-сайту, Google і Bing можуть правильно відображати і заголовок і опис, але якщо хтось ділитиметься сторінкою у соціальних мережах, попередній перегляд (прев'ю) може бути некоректним, оскільки теги Open Graph не розпізнаються.

Оцінка ефективності мета-тегів: 6/10 (присутні проблеми із коректним відображенням сайту в пошуку, а також інших соц. мережах).

Не зважаючи на серйозні проблеми з UX-дизайном, користувач зможе розібратись із додаванням продукту у кошик, хоч він може і заплутатись у виборі категорії через складно читабельну сітку. Після цього автоматично відкриється пічна панель з кошиком, де можна відразу перейти до оформлення замовлення.

Сторінка "чекауту" може розсіяти увагу користувача через неоднорідний дизайн і яскраві лінії та жовтий текст.

Форма збирає усю необхідну інформацію для підтвердження замовлення і надає вибір між видом доставки і оплати, а також розраховує вартість доставки.

Оцінка шляху до оформлення замовлення: 6/10 (користувач може “загубитись по дорозі”, такий набір кольорової гами не підходить для UX)

Рисунок 1.4 – Головна секція сайту Kilogramm

Дизайн сайту Kilogramm (рис. 1.4) відповідає сучасним тенденціям, однак має певні недоліки, які впливають на зручність користування. Розглянемо основні аспекти оформлення та їхній вплив на сприйняття відвідувачем.

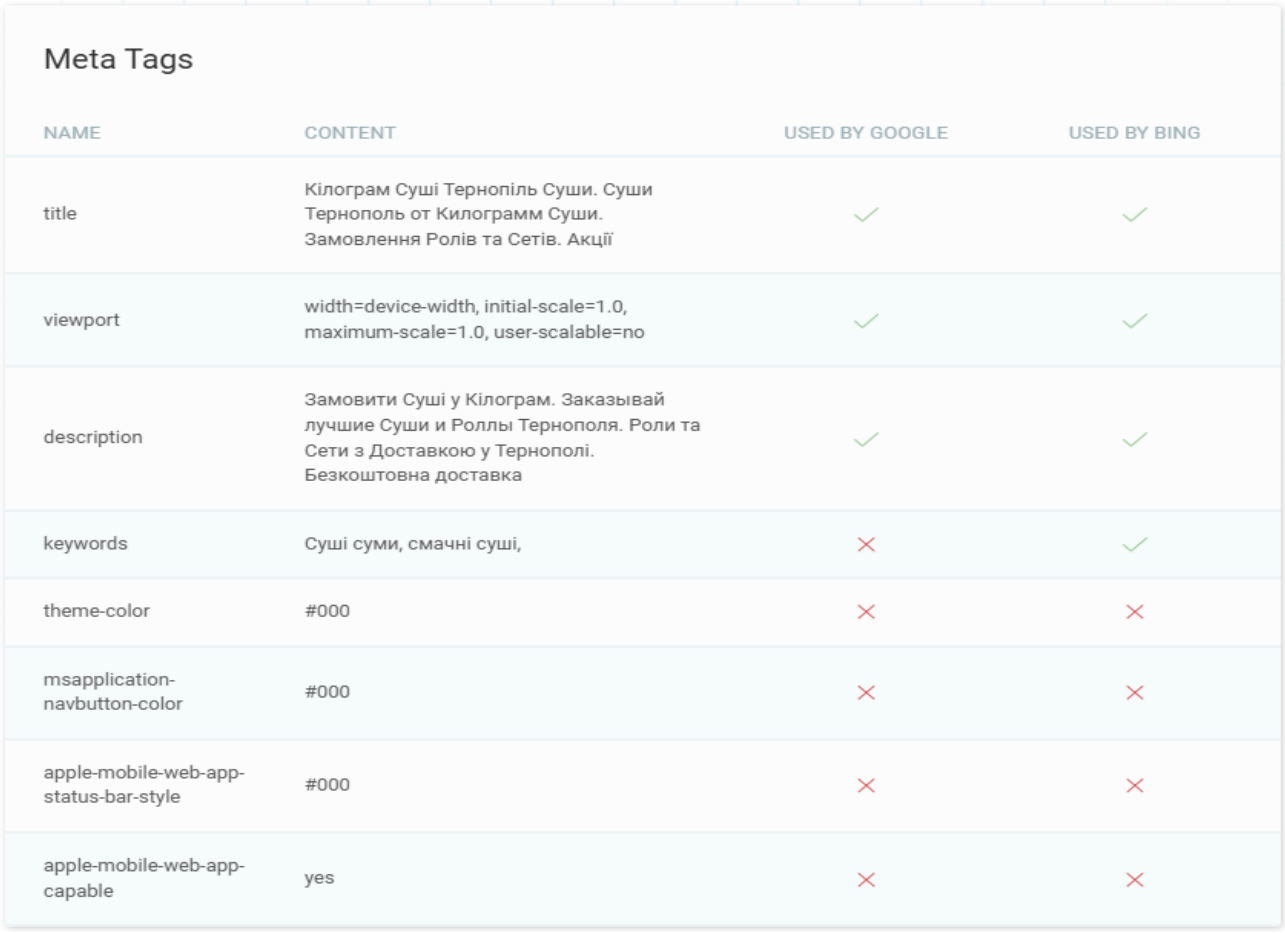
Зручність користування та навігація: використання прихованого меню на великому екрані ускладнює навігацію, змушуючи користувача витрачати більше часу на пошук потрібних розділів. Крім того, меню має непропорційно великі розміри, що робить його громіздким. Оптимальним рішенням було б відкриття частини меню для настільної версії.

Розташування елементів та використання простору: сайт дотримується принципу мінімалізму, але в деяких секціях надлишок порожнього простору створює відчуття незавершеності. Головна сторінка виглядає привабливо, проте подальше розташування елементів менш продумане, що ускладнює сприйняття інформації.

Візуальне оформлення та сприйняття: кольорова гама контрастна й запам’ятовується, проте надмірне використання яскравого помаранчевого може втомлювати. Варто збалансувати кольори більш спокійними відтінками. Для сайту

доставки їжі важливо, щоб зображення страв викликали бажання зробити замовлення, тому їхній візуальний пріоритет потрібно підсилити.

Оцінка UI/UX-дизайну: 6.5/10 (дизайн сайту сучасний і стильний, але має недопрацювання, що знижують зручність).



Meta Tags

NAME	CONTENT	USED BY GOOGLE	USED BY BING
title	Кілограм Суші Тернопіль Суши. Суши Тернополь от Килограмм Суши. Замовлення Ролів та Сетів. Акції	✓	✓
viewport	width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=no	✓	✓
description	Замовити Суші у Кілограм. Заказывай лучшие Суши и Роллы Тернополя. Роли та Сети з Доставкою у Тернополі. Безкоштовна доставка	✓	✓
keywords	Суші суми, смачні суші,	✗	✓
theme-color	#000	✗	✗
msapplication-navbutton-color	#000	✗	✗
apple-mobile-web-app-status-bar-style	#000	✗	✗
apple-mobile-web-app-capable	yes	✗	✗

Рисунок 1.5 – Результати аналізу мета-тегів

Завдяки даним, які зображені на рис. 1.5, ми можемо сказати, що:

1. І Google, і Bing розпізнають:

– Title tag ("Кілограм Суші Тернопіль. Замовлення Роли та Сети. Акції");

– налаштування вікна перегляду (для адаптивності на мобільних пристроях);

– мета-опис (про замовлення суші та ролів у Тернополі з безкоштовною доставкою).

2. жодна з пошукових систем не розпізнає:

– тег кольору теми (#000);
– колір кнопки навігації мобільного додатку (#000);
– стиль статусу мобільного веб-додатку Apple (#000);
– підтримка мобільних веб-додатків Apple (так).

Це вказує на те, що основні елементи SEO (заголовок і опис) коректно працюють для пошукових систем, але мета-теги, специфічні для мобільних пристроїв і теми, не розпізнаються жодною з пошукових систем.

Оцінка ефективності мета-тегів: 6/10 (основні аспекти мета-тегів працюють відмінно, та все ж є куди покращуватись у інших аспектах).

Незважаючи на негаразди на головній сторінці сторінці, меню виглядає доволі привабливо із суб'єктивної точки зору опитаних користувачів. Карточки із продукцією відділені, чітко видно основну інформацію про продукт – назва, актуальна ціна, вага і, що головне, фотографія вигляду страви. Кнопка "Кошик" веде прямо до оформлення замовлення. Сторінка загалом добре структурована, проте мапа доставки займає забагато місця, а блок з формою не дуже компактний і поля введення розкидані.

Оцінка шляху до оформлення замовлення: 7/10 (сторінка працює ефективно, але деякі елементи можна зробити зручнішими та естетичнішими).

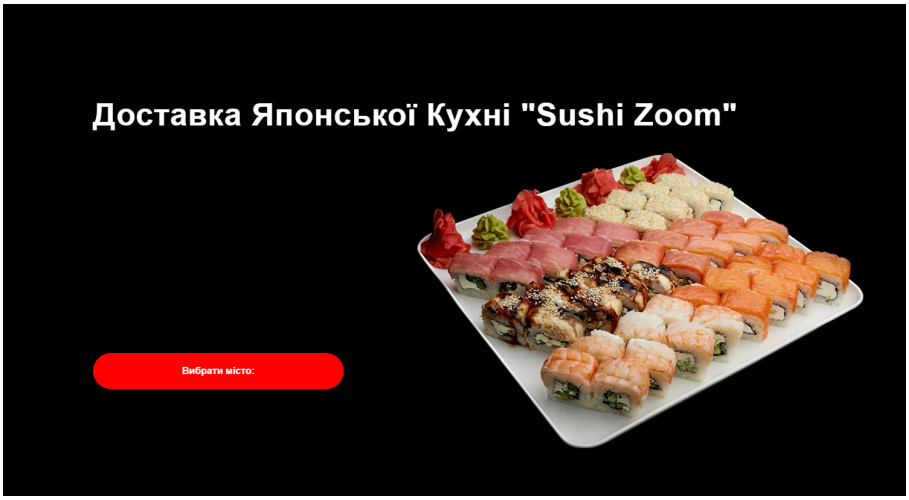


Рисунок 1.6 – Головна сторінка SushiZoom

Дизайн веб-сайту Sushi Zoom (рис. 1.6) має якісні фотографії продукції та структуровану систему меню, але демонструє суттєві проблеми з юзабіліті. Темне тло створює контраст із вмістом, проте загальна композиція не сприяє інтуїтивній навігації. Розглянемо основні аспекти інтерфейсу та їхній вплив на користувацький досвід.

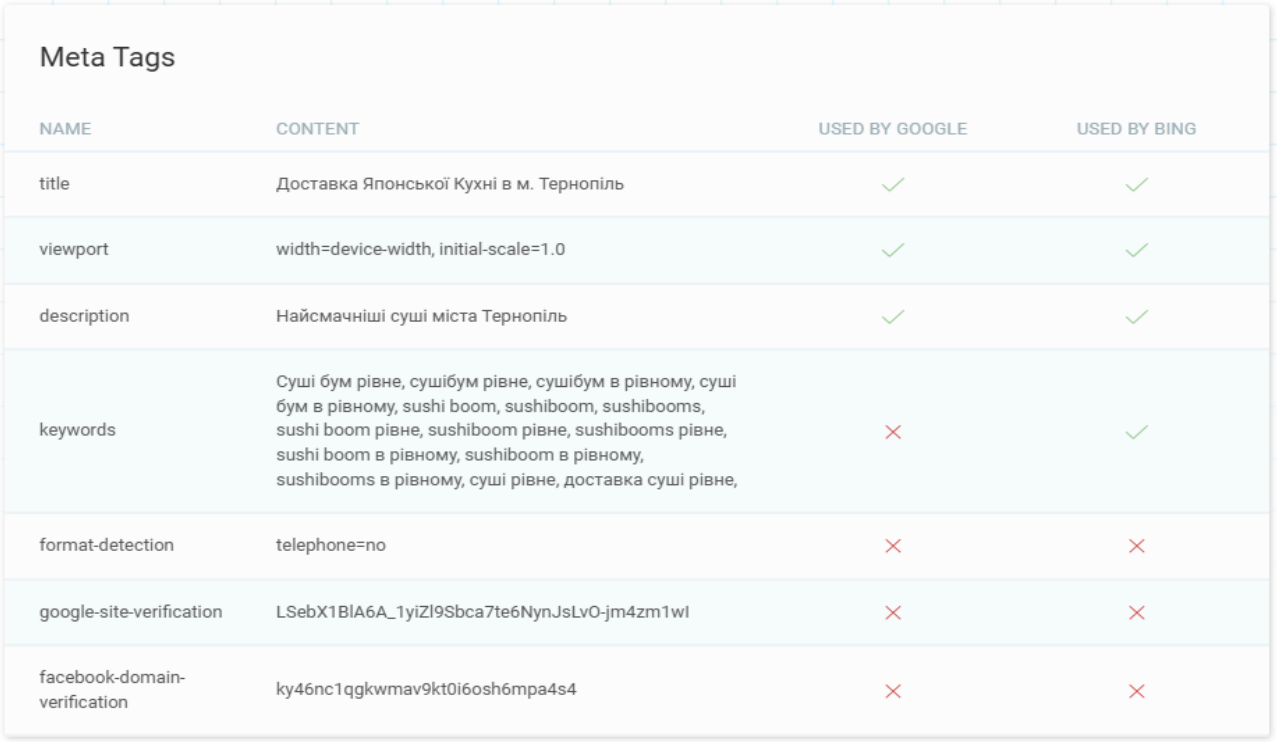
Навігація та інформаційна архітектура: екран вибору міста демонструє слабку інформаційну архітектуру зі списком за алфавітом, що створює когнітивне перевантаження. Містам, організованим за кирилицею, бракує належного інтервалу та ієрархії, що робить вибір локації невиправдано складним. Відсутня належна структура хедера для обрамлення контенту та навігації користувачів.

Візуальний дизайн і послідовність бренду: кольорова палітра є проблематичною з різким чорним фоном і білим текстом, яким бракує вишуканості. Ідентичність бренду виглядає непослідовною, з логотипом, відокремленим від основного контенту.

Структура контенту та користувацький потік: картки продуктів виявляють значні проблеми з макетом. Текстова ієрархія погано реалізована, створюючи недостатнє розмежування між назвами продуктів, інгредієнтами та рекламним текстом. Описам бракує належного інтервалу, що ускладнює швидкий перегляд.

Елементи заклику до дії та конверсії: кнопкам заклику до дії бракує візуальної помітності і вони не ефективно виконують свою функцію (не враховуючи той факт, що самий текст написано з одруківкою – “детелі”). Щоденні акції ("актуально тільки щопонеділка!") недостатньо виділені, попри їхню важливість для конверсії.

Оцінка UI/UX-дизайну: 4/10 (основні компоненти присутні, але реалізовані неефективно, що негативно впливає на користувацький досвід).



Meta Tags

NAME	CONTENT	USED BY GOOGLE	USED BY BING
title	Доставка Японської Кухні в м. Тернопіль	✓	✓
viewport	width=device-width, initial-scale=1.0	✓	✓
description	Найсмачніші суші міста Тернопіль	✓	✓
keywords	Суші бум рівне, сушібум рівне, сушібум в рівному, суші бум в рівному, sushi boom, sushiboom, sushibooms, sushi boom рівне, sushiboom рівне, sushibooms рівне, sushi boom в рівному, sushiboom в рівному, sushibooms в рівному, суші рівне, доставка суші рівне,	✗	✓
format-detection	telephone=no	✗	✗
google-site-verification	LSebX1BlA6A_1yiZl9Sbca7te6NynJsLvO-jm4zm1wI	✗	✗
facebook-domain-verification	ky46nc1qgkwmav9kt0i6osh6mpa4s4	✗	✗

Рисунок 1.8 – Результати аналізу мета-тегів для SushiZoom

З рис. 1.8 ми можемо побачити такий аналіз:

1. і Google, і Bing визнають:

– Title tag ("Доставка Японської Кухні в м. Тернопіль" - Japanese Cuisine Delivery in Ternopil);

– налаштування Viewport для адаптивності на мобільні пристрої;

– description tag ("Найсмачніші суші міста Тернопіль" - The most delicious sushi in Ternopil).

2. Тег ключових слів:

– розпізнає лише Bing, але не Google (те ж саме, що і у попередніх сервісах);

– містить багато варіацій ключових слів для "суші-бум" та "суші в Рівному".

3. Жодна з пошукових систем не визнає:

– визначення формату для телефону;

– код підтвердження сайту Google;

– код підтвердження домену Facebook.

Варто зазначити, що для SushiZoom і GuruFood прев'ю у соціальних мережах виглядатиме так, як на рисунку 1.9 (для визначення цього було використано сервіс metatags.io)

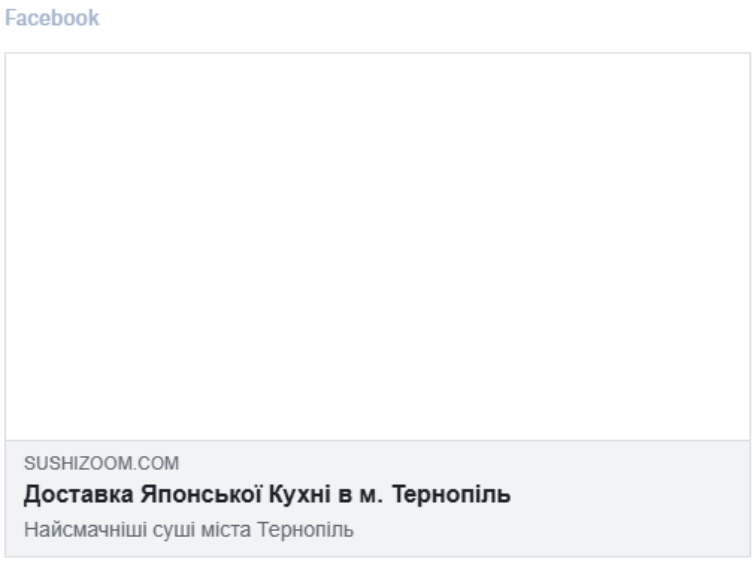


Рисунок 1.9 – Приклад попереднього перегляду (прев'ю) реклами продукції у соціальних мережах

Оцінка ефективності мета-тегів: 4/10 (цей сервіс має базові можливості для пошукової видимості, але потребує вдосконалення тегів верифікації та оптимізації соціальних мереж).

Шлях від входу на сайт Sushi Zoom до оформлення замовлення надмірно заплутаний і викликає роздратування. Спочатку користувачі стикаються порожнім екраном і єдиним варіантом – обрати місто (прогортавши вниз можна знайти лише акційні сети). А потім користувача зустрічає сторінка з громіздким списком міст в алфавітному порядку без функції пошуку. Більше того, далі на користувача очікує ще один вибір – обрати район у місті. Лише після цього ми зможемо перейти до вибору продукції. Це негативно впливає на простоту використання користувачем, навантажує його і перешкоджає безперешкодному замовленню.

Оцінка шляху до оформлення замовлення: 3/10 (заплутана система вибору місцезнаходження сприяє поганому враженню).

В цьому підрозділі було проаналізовано веб-сторінки трьох тернопільських ресторанів які пропонують доставку їхньої їжі (японська та східна кухня). Аналіз проводився на основі досвіду користувача, де підмічено слабкі і сильні сторони, які були оцінені за 10-бальною шкалою. Підсумуємо оцінки за допомогою таблиці 1.1.

Таблиця 1.1 – Оцінка веб-застосунків за критеріями

Назва ресторану	UI/UX-дизайн	Мета-теги	Шлях користувача
GuruFood	5	6	6
Kilogramm	6.5	6	7
SushiZoom	4	4	3

Проблеми у використанні фізичного пристрою:

– необхідність забезпечення безперебійного з’єднання між пристроєм та системою замовлень;

– можливість технічних збоїв у роботі сенсорного екрана або мікроконтролера;

– обмежена масштабованість, якщо пристрій не враховує зростання обсягу замовлень.

1.4 Порівняльний аналіз існуючих онлайн-сервісів оформлення замовлення

Аналізуючи веб-сторінки як користувачі ми завжди звертаємо увагу на їх дизайн – структуру сторінки, CTA, легко-читаєму інформацію, інтуїтивність, зручність користування. Це все невід’ємна частина. А на що звертають увагу розробники, це – ефективність. Основними аспектами її складають ефективність, доступність, оптимальні методи, SEO (оптимальний пошук систем). З цим розібратись нам допоможе такий веб-ресурс як PageSpeed (PageSpeed Insights).

Розглянемо вихідні дані, які видає цей сервіс, і проаналізуємо їх зі сторони робробника.

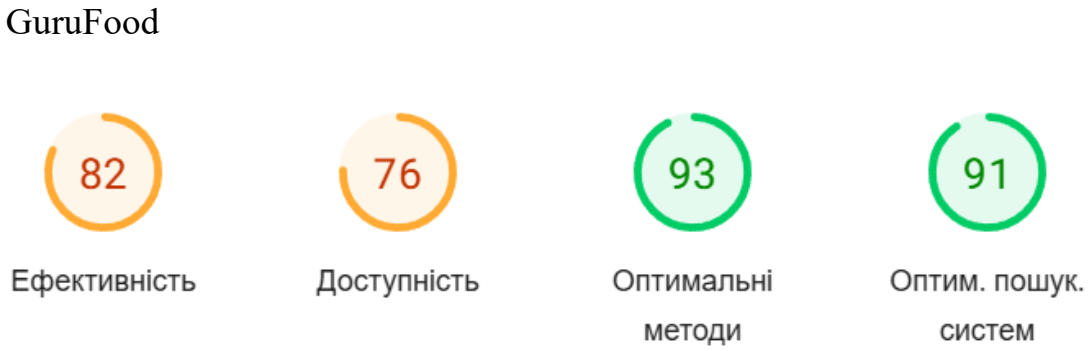


Рисунок 1.10 – Результати оцінки від PageSpeed

Корпорація Google надає розширений засіб для перевірки та збільшення відгуку на веб-запити будь-якого ресурсу. Для новачка робота з сервісом Google Analytics може здатися трохи складною, але досить прочитати кілька коротких довідкових матеріалів і кілька разів використовувати інструмент і зручна робота з ним забезпечена. Даний сервіс дозволяє стежити за такими показниками:

1. Швидкість завантаження ресурсу, що обчислюється для певної вибірки переглядів сторінок. За допомогою цієї вибірки можна проаналізувати, як змінюється швидкість завантаження веб-сторінок залежно від розташування користувачів, операційної системи, браузера, роздільної здатності дисплея та цілого ряду інших параметрів.

2. Швидкість виконання/завантаження для одиничного запиту чи дії. Наприклад, можна визначити швидкість завантаження графічного елементу (зображення, GIF-файлу тощо) або час відгуку натискання кнопки.

3. Ефективність обробки документа на сайті та швидкість надання цього файлу користувачеві.

1.5 Постановка задачі

Завданнями роботи є:

– дослідити наявні системи прийому та обробки замовлень для закладів харчування;

– провести теоретичний аналіз сфери;

– охарактеризувати структуру предметної області та базову модель;

– описати уже існуючі механізми реалізації, виділити наявні проб¬леми в галузі та шляхи їх вирішення;

– на основі проведених досліджень визначити основні функції сис¬теми, сформувати низку функціональних та нефункціональних вимог, розро¬бити модель функцій, які система повинна виконувати;

– підвести підсумки про необхідність розробки системи;

– сформувати об’єкт та мету для наступних досліджень;

– оцінити ступінь виконання поставлених завдань.

– розробити інтегрований вебсайт для замовлення їжі з інтуїтивним та зручним інтерфейсом для клієнтів;

– інтегрувати вебсайт із пристроєм на базі ESP32 для автоматичної передачі та відображення замовлень на кухні;

– забезпечити стабільне з'єднання між усіма елементами системи, а також гнучкість для масштабування в майбутньому;

– розробити механізм зміни статусів замовлень на кухні та їх синхронізацію зі станом замовлень на вебсайті;

– впровадити систему сповіщень для клієнтів щодо статусу їх замовлень;

– забезпечити технічну підтримку для уникнення збоїв у роботі системи та пристрою.

На основі цього розробити програмно апаратний комплекс для прийому та обробки замовлень для закладу харчування, та зробити висновки на основі виконаної роботи.

1.6 Висновки до першого розділу

Даний аналіз предметної області виявив ключові проблеми та можливості для вдосконалення у сфері замовлень їжі в ресторані східної кухні. Інтеграція вебсайту та пристрою на базі ESP32 дозволить значно підвищити ефективність обробки замовлень, покращити якість обслуговування клієнтів і оптимізувати внутрішні процеси ресторану. Подальші дослідження та реалізація поставлених завдань допоможуть створити інноваційну систему, яка відповідатиме сучасним вимогам ринку.

2 ПРОЄКТУВАННЯ ПРОГРАМНО-ТЕХНІЧНОГО ЗАСОБУ

2.1 Аналіз засобів розробки веб-сайтів

Розробка сучасного веб-застосунку вимагає вибору правильного технологічного стеку, який забезпечує продуктивність, масштабованість, безпеку та зручність у розробці, створюючи бездоганний користувацький досвід при збереженні структурованого та підтримуваного коду. Існує безліч технологій для веб-розробки, починаючи від монолітних архітектур, таких як Ruby on Rails, PHP (Laravel) і Django, та закінчуючи сучасними модульними рішеннями, що використовують фронтенд-фреймворки на кшталт React, Angular і Vue, у поєднанні з бекенд-технологіями, такими як Node.js, Python і Go. Вибір залежить від таких факторів, як обробка даних у реальному часі, оптимізація SEO, ефективність API та гнучкість розгортання, що безпосередньо впливає на продуктивність та орієнтованість платформи на користувача.

Для програмно-технічного комплексу для автоматизованого прийому та оповіщення про замовлення в ресторані східної кухні, де критично важливими є оперативне оновлення замовлень, захищені транзакції та інтуїтивний інтерфейс, обраний стек повинен забезпечувати високу доступність, швидкий відгук системи та надійну архітектуру. Використовуючи Next.js, TypeScript і Tailwind CSS, фронтенд отримує переваги рендерингу на сервері (SSR) і генерації статичних сторінок (SSG), що забезпечує миттєве завантаження контенту та покращену SEO-оптимізацію, одночасно підтримуючи чистий, масштабований і візуально узгоджений інтерфейс. На бекенді NestJS на базі Express.js пропонує високоструктуровану, модульну структуру з підтримкою впровадження ін’єкції залежностей(DI – Dependency Injection), що дозволяє ефективно керувати API, працювати з мікросервісами та реалізовувати обмін даними в реальному часі — ключову функцію для відстеження статусу замовлень. Як база даних використовується PostgreSQL, що забезпечує дотримання ACID-принципів, підтримку як структурованих, так і напівструктурованих даних, а також розширені

методи індексації, що робить її ідеальною для керування профілями користувачів, списками ресторанів і складними зв’язками між замовленнями.

Для спостереження за системою, моніторингу продуктивності та гарантування стабільності роботи інтегровано Grafana та Loki, які забезпечують повноцінну екосистему моніторингу та логування, дозволяючи в режимі реального часу відстежувати роботу API, запити до бази даних і стан серверів. Caddy виконує роль захищеного реверс-проксі з автоматичною підтримкою TLS, що зменшує складність конфігурації сервера та підвищує загальний рівень безпеки системи. Завдяки контейнеризації в Docker досягається послідовність середовищ розробки, спрощене розгортання та масштабованість у хмарних або локальних інфраструктурах, що робить його важливим елементом сучасних, високомасштабованих застосунків.

2.2 Вимоги до проєктованого програмно-технічного засобу.

Для реалізації цього завдання важливим етапом є ретельне вивчення та обґрунтування вимог до програмно-технічного засобу. Даний процес передбачає формування чітких завдань, які мають вирішуватися системою, а також опис основних характеристик, таких як продуктивність, масштабованість, безпека та зручність використання. Слід також врахувати сумісність із сучасними технологіями, такими як інтеграція з популярними фреймворками, базами даних та інструментами для розробників. Зокрема, важливо забезпечити підтримку адаптивного інтерфейсу користувача та можливість гнучкого налаштування системи залежно від потреб бізнесу.

2.3 Архітектура програмно-технічного засобу.

Архітектура системи формується на основі принципів модульності, яка забезпечує зручність розробки, тестування та підтримки компонентів. Важливо

передбачити багаторівневу структуру, яка включатиме клієнтський рівень, серверний рівень та рівень бази даних.

На першому рівні - клієнтському - розташовані інтерфейси для користувачів, які повинні бути дружніми, адаптивними та відповідати сучасним веб-стандартам. Використання бібліотек, таких як React, дозволяє створити масштабовані та інтерактивні інтерфейси.

На серверному рівні система має забезпечувати швидку обробку запитів, використовуючи ефективні фреймворки, наприклад, NestJS. Цей рівень також відповідає за взаємодію з базою даних, управління сесіями користувачів та дотримання принципів безпеки.

Рівень бази даних має підтримувати високий рівень продуктивності та бути сумісним із сучасними технологіями, такими як SQL або NoSQL рішення. Важливим аспектом є забезпечення резервного копіювання даних та їх шифрування.

Архітектура повинна передбачати можливість інтеграції з іншими системами, такими як платіжні платформи, служби доставки та аналітичні інструменти. Особливу увагу слід приділити гнучкості налаштувань, що дозволить адаптувати програмно-технічний засіб залежно від потреб конкретного закладу харчування.

2.4 Технології для реалізації.

Для розробки програмно-технічного засобу рекомендовано використання сучасних технологій, таких як React для створення клієнтських інтерфейсів, Tailwind CSS для стильового оформлення, а також Next.js для серверного рендерингу. На серверному боці варто застосовувати NestJS, який забезпечує високий рівень продуктивності та зручність організації коду. Інтеграція з базою даних може бути реалізована через популярні ORM, такі як TypeORM або Prisma.

У своїй роботі я зробив вибір в користь TypeORM через кращу модульність в порівнянні з Prisma.

Задля забезпечення швидкої розробки та тестування можна використовувати інструменти, на зразок Vite для створення проєктів із високою продуктивністю. Доцільним також буде впровадження CI/CD процесів для автоматизації розгортання та тестування системи. Для CI/CD я використаю інструмент GitHub Actions. Для системи логування я використаю програмний сервіс Grafana в поєднанні з експортером Loki який братиме логи з Docker контейнеру із застосунком і передаватиме їх у Grafana. Задля покраще нння процесу розгортання проєкту в новому і вже існуючих серидовищах я використаю Docker в якому пропишу всі залежності потрібні для роботи бекенд застосунку.

React – бібліотека для створення користувацьких веб інтерфейсів. Вибір був зроблений в користь цієї бібліотеки через спрощення роботи з HTML та логікою веб-застосунку. Порівняно зі старішими бібліотеками для роботи з користувацьким веб-інтерфейсом React має ряд переваг таких як:

– компонентний підхід – можливість розділяти частини застосунку в компоненти задля кращої структуризації коду та повторного перевикористання. При правильному застосуванні цього підходу кількість коду потрібного для реалізації застосунку суттєво зменшується через відповідність принципу DRY(Don’t repeat yourself – не повторюй себе);

– декларативний стиль – на відміну від чистого JavaScript та старіших бібліотек потипу jQuery які пропонували роботу над станом в імперативному стилі, React навпаки зробив ставку на більш абстрактний, зручний, та швидший для абсолютної більшості задач метод опису стану. Імперативний метод передбачає опис чітких інструкцій що потрібен розбити застосунок. Його перевага полягає в наочності того що відбувається в коді без зайвих абстракцій, надає більший контроль. Проте з більшим контролем приходить і більше когнітивне навантаження під час написання логічної частини застосунку, частіші випадки зміщення фокусу з бізнес логіки на технічні аспекти. Декларативний стиль в свою чергу пропонує

розробнику описувати_«ЯК» поводиться застосунок на вищому рівні абстракції концентруючись на бізнес логіці і прибираючи безліч аспектів пов’язаних з «боротьбою» з програмним кодом щоб він виконував бізнес логіку;

– велика екосистема - бібліотека має багатий вибір готових компонентів, плагінів та інструментів, які значно спрощують розробку застосунків. Це дозволяє розробникам швидко інтегрувати функціонал, не витрачаючи зайвий час на написання власних рішень з нуля.

React також підтримує серверний рендеринг за допомогою таких фреймворків, як Next.js, що дозволяє досягти кращої продуктивності та оптимізації для пошукових систем (SEO). Серверний рендеринг забезпечує швидше завантаження контенту, оскільки сторінки генеруються на сервері до відправлення клієнту;

– підтримка TypeScript - дозволяє використовувати статичну типізацію для підвищення якості коду. Застосування TypeScript допомагає уникати багатьох помилок під час розробки та значно спрощує підтримку великих проєктів завдяки чітким визначенням типів даних.

– широкий спектр інструментів для налагодження, таких як React Developer Tools, що інтегрується з браузерами і дозволяє розробникам ефективно відслідковувати стан компонентів, пропси та їхню взаємодію. Ці інструменти є незамінними для швидкого виявлення та виправлення помилок.

Окрім уже згаданих переваг, React також відзначається активною спільнотою розробників, які постійно створюють нові інструменти, документацію та приклади. Це значно полегшує процес навчання та впровадження React навіть для новачків. Інтеграція з іншими сучасними технологіями, такими як Redux для управління станом та GraphQL для запитів до API, дає змогу побудувати високопродуктивну систему з мінімальними витратами на підтримку.

Ще однією важливою перевагою є можливість створення реакт-інтерфейсів для мобільних додатків за допомогою React Native, що дозволяє використовувати

той самий набір навичок для веб-розробки та мобільної розробки, забезпечуючи кросплатформенність.

Node.js - це середовище виконання JavaScript, яке дозволяє запускати код поза браузером, що особливо корисно для серверної частини веб-застосунків. Його популярність зумовлена рядом важливих переваг, які роблять його незамінним інструментом для розробників.

Node.js використовує подієво-орієнтовану архітектуру, яка забезпечує неблокуючий ввід/вивід, дозволяючи обробляти велику кількість підключень одночасно. Це робить Node.js ідеальним для створення реальних застосунків, таких як чати, системи обміну повідомленнями, стримінг тощо.

Переваги Node.js:

– Висока продуктивність: Завдяки рушію V8 Node.js забезпечує швидке виконання JavaScript-коду.

– Неблокуючий ввід/вивід: Подієво-орієнтована архітектура дозволяє обробляти запити без затримок.

– Масштабованість: Node.js легко обробляє тисячі одночасних запитів, що робить його ідеальним для великих систем.

– Єдиний стек технологій: Використання JavaScript як на клієнтській, так і на серверній стороні спрощує розробку.

– Багата екосистема: Доступ до мільйонів пакетів через npm (Node Package Manager).

– Гнучкість: Підтримка модульності та легкість інтеграції з іншими технологіями.

– Швидке розгортання: Node.js дозволяє швидко запускати нові системи завдяки легкому налаштуванню.

– Підтримка реального часу: Ідеально підходить для застосунків з реалтайм-функціями, таких як чати та ігри.

– Розвинена спільнота: Величезна кількість розробників, які постійно сприяють розвитку платформи.

– Кросплатформенність: Можливість запуску застосунків на різних операційних системах.

Node.js є потужним інструментом для створення сучасних веб-систем і завдяки своїм перевагам продовжує залишатися одним з головних виборів для розробників серверної логіки.

Node.js також вирізняється високою продуктивністю завдяки використанню рушія V8 від Google, який забезпечує швидке виконання JavaScript-коду. Це дозволяє створювати складні системи, які працюють швидко навіть за великих навантажень.

Платформа надає можливість створювати мікросервісну архітектуру, що сприяє легкому масштабуванню та підтримці застосунків. Завдяки цьому Node.js стає вибором для компаній, які прагнуть ефективно обробляти дані, інтегрувати різноманітні сервіси та оновлювати функціонал без значної перерви в роботі системи.

Інновації в межах платформи постійно з’являються завдяки зусиллям глобальної спільноти, яка працює над удосконаленням основних інструментів і розширенням функціональності. Зокрема, активно підтримуються та оновлюються такі популярні фреймворки, як Express.js, Nest.js, які ще більше спрощують розробку серверної логіки.

Node.js також демонструє свою силу в галузі обробки великих даних і машинного навчання завдяки інтеграції з бібліотеками, такими як TensorFlow.js, що дозволяють виконувати складні математичні розрахунки безпосередньо на сервері. Це відкриває нові горизонти для створення інтелектуальних систем і забезпечує платформу широкими можливостями для розробників.

Express.js є одним із найпопулярніших фреймворків для розробки серверної логіки на платформі Node.js, що робить його вибором багатьох розробників за його простоту, функціональність та розширюваність. Це легкий і гнучкий інструмент, який дозволяє ефективно створювати веб-додатки та API, скорочуючи час на програмування завдяки своїй мінімалістичній структурі.

Основні переваги Express.js:

– простота використання: завдяки своїй мінімалістичній архітектурі Express.js легко вивчати та використовувати навіть новачкам у програмуванні;

– гнучкість: Express.js дозволяє створювати додатки будь-якої складності – від односторінкових веб-додатків до масштабних API з великою кількістю функцій;

– швидка розробка: надає інструменти для швидкого створення серверної логіки, що сприяє скороченню часу розробки проєкту;

– розширюваність: легка інтеграція з іншими модулями та бібліотеками Node.js для додавання додаткових функціональних можливостей;

– велика спільнота: забезпечує розробників численними готовими модулями, плагінами та відповідями на питання у спільноті;

– підтримка middleware: дозволяє легко обробляти HTTP-запити, додавати автентифікацію чи логування через використання middleware;

– повна сумісність із Node.js: використовує всі переваги рушія V8 і екосистеми Node.js;

– наскрізний роутинг: розширені можливості для створення маршрутизаторів, що інтегруються у складні системи;

– масштабованість: підтримка мікросервісної архітектури для роботи з великими додатками;

– підтримка RESTful API: полегшує створення додатків із архітектурою REST;

– висока продуктивність: завдяки мінімальним накладним витратам на виконання;

– кросплатформенність: можливість розгортання на різних операційних системах для забезпечення гнучкості.

Express.js залишається важливим елементом екосистеми Node.js, пропонуючи високопродуктивні рішення для створення серверної логіки з мінімальними зусиллями.

NestJS - потужний інструмент для розробників, який поєднує простоту та функціональність для створення серверних додатків. Його основні переваги полягають у гнучкості, масштабованості та інтеграції з сучасними технологіями. Завдяки побудові на основі TypeScript, NestJS дозволяє легко підтримувати чистоту та структурованість коду, а його модульна архітектура спрощує розробку великих додатків.

Переваги NestJS:

– гнучка модульна архітектура для ефективного розділення логіки додатків;

– інтеграція з TypeScript для покращення продуктивності та читабельності коду;

– підтримка Dependency Injection для зручної організації програмного забезпечення;

– легка інтеграція з популярними бібліотеками та фреймворками, такими як Express.js або Fastify;

– розширена підтримка тестування для забезпечення стабільності та якості коду;

– побудова RESTful API з мінімальними зусиллями;

– потужна підтримка GraphQL для сучасних додатків зі складною логікою;

– наскрізна логіка для обробки запитів та відповіді;

– широкі можливості для роботи з мікросервісами та створення розподілених систем;

– інтеграція із системами баз даних через ORM, наприклад, TypeORM або Sequelize;

– підтримка асинхронної обробки даних для забезпечення високої продуктивності додатків;

– інтуїтивно зрозумілий CLI для швидкого створення та управління проектами;

– можливість налаштування middleware для додаткової гнучкості при обробці запитів;

– документована підтримка WebSocket для реального часу;

– підтримка кросплатформенності для розгортання на різних операційних системах;

– активна спільнота та регулярні оновлення, що забезпечують довгострокову стабільність.

NestJS є фреймворком, який пропонує зручну підтримку GraphQL для створення сучасних додатків зі складною логікою. Він забезпечує наскрізну обробку запитів і відповідей, пропонує широкі можливості для роботи з мікросервісами та створення розподілених систем. Також NestJS інтегрується з системами баз даних через ORM, як-от TypeORM або Sequelize, і підтримує асинхронну обробку даних для забезпечення високої продуктивності додатків. Завдяки інтуїтивно зрозумілому CLI, можна швидко створювати і керувати проектами. Фреймворк дозволяє налаштовувати middleware для обробки запитів, має документовану підтримку WebSocket для реалізації функціоналу реального часу, а також підтримує кросплатформенність, що дозволяє розгортати додатки на різних операційних системах. Активна спільнота та регулярні оновлення забезпечують довгострокову стабільність застосування цього інструменту.

PostgreSQL - це одна з найпопулярніших систем керування базами даних (СКБД) з відкритим кодом, яка отримала довіру мільйонів розробників по всьому світу. Завдяки своїй багатій функціональності та надійності PostgreSQL ідеально підходить для створення складних, масштабованих і критично важливих додатків.

Однією з найбільш важливих переваг PostgreSQL є її відповідність стандартам SQL та підтримка розширень, що дозволяє адаптувати систему під потреби конкретного проекту. СКБД забезпечує відмінну продуктивність, що досягається завдяки оптимізації запитів, індексації та системі транзакцій ACID. PostgreSQL також підтримує безліч типів даних, включаючи JSON/JSONB, який

використовується для зберігання та обробки неструктурованих даних, що робить його особливо привабливим для сучасних веб-додатків і аналітичних систем.

Функціональність PostgreSQL включає просунуті механізми реплікації та масштабування, що дозволяють зберігати стабільну продуктивність навіть при значному збільшенні обсягу даних. Вбудована система перевірки цілісності даних та можливість створення користувацьких функцій і тригерів роблять PostgreSQL універсальним інструментом для будь-якого типу бізнесу. До того ж, ця СКБД активно використовується у хмарних рішеннях, оскільки вона пропонує надійну інтеграцію з різними платформами, такими як AWS, Google Cloud Platform і Microsoft Azure.

Переваги PostgreSQL:

– відкритий код, що забезпечує безкоштовне використання та доступність для спільноти розробників;

– підтримка стандартів SQL та розширюваність для кастомізації функціоналу;

– підтримка складних типів даних, включаючи JSON/JSONB, геопросторові дані (PostGIS) та масиви;

– висока продуктивність завдяки оптимізації запитів та використанню індексів;

– підтримка транзакцій ACID для забезпечення цілісності даних;

– можливість горизонтального масштабування і реплікації для великих систем;

– інтеграція з хмарними платформами та сучасними технологіями;

– гнучкі можливості для створення користувацьких функцій, тригерів та розширень;

– високий рівень безпеки завдяки ролям, шифруванню та системі контролю доступу;

– активна спільнота користувачів і регулярні оновлення, що додають нові можливості.

PostgreSQL є багатофункціональною системою управління базами даних з відкритим кодом, яка забезпечує безкоштовне використання та доступність для широкої спільноти розробників. Ця платформа підтримує стандарти SQL, що робить її зручною для інтеграції та використання у багатьох проектах, а також дозволяє розширювати функціонал, створюючи кастомні рішення для специфічних потреб користувачів.

Однією з важливих рис PostgreSQL є її здатність працювати зі складними типами даних. База даних підтримує JSON та JSONB, що робить її ідеальним вибором для роботи з сучасними веб-технологіями, а також геопросторові дані через інтеграцію з PostGIS, що особливо цінно для проєктів, які потребують картографії або геоаналізу. Додатково, можливість зберігати та обробляти масиви даних робить PostgreSQL дуже гнучкою для широкого спектра застосувань.

Продуктивність системи оптимізована завдяки ефективному використанню індексації та механізмів обробки запитів, що забезпечує високу швидкість роботи навіть з великими обсягами даних. Крім того, PostgreSQL реалізує транзакції, що відповідають принципам ACID, гарантують цілісність даних і забезпечують надійність під час роботи з критично важливими операціями.

Система також демонструє чудові можливості масштабування. Вона підтримує горизонтальне масштабування та реплікацію, що дозволяє використовувати її для великих і складних систем, які потребують розподіленого управління даними. Інтеграція PostgreSQL з хмарними платформами та сучасними технологіями забезпечує актуальність та відповідність вимогам сучасного ринку.

Користувачі мають широкі можливості для створення власних функцій, тригерів та інших розширень, що відкриває двері для персоналізації бази даних під специфічні завдання. Велика увага приділяється безпеці, адже система підтримує комплексну систему ролей, шифрування даних та суворий контроль доступу.

Окрім технічних характеристик, PostgreSQL має активну та згуртовану спільноту користувачів, яка постійно працює над вдосконаленням платформи.

Регулярні оновлення додають нові функції та підвищують стабільність системи, що робить її надійним вибором для проєктів різного масштабу та складності.

Таким чином, PostgreSQL - це не просто база даних, а ціла екосистема, яка об'єднує потужність, гнучкість і надійність, що робить її ідеальним вибором як для маленьких проєктів, так і для масштабних корпоративних систем.

Grafana - це популярна платформа для візуалізації даних і моніторингу, яка забезпечує гнучкість, масштабованість і зручність у використанні. Вона дозволяє створювати інтерактивні графіки, панелі моніторингу та звіти, які допомагають користувачам аналізувати дані та приймати обґрунтовані рішення. Платформа підтримує інтеграцію з численними джерелами даних, такими як Prometheus, InfluxDB, Elasticsearch, а також хмарними сервісами, включно з AWS, Azure і Google Cloud.

Однією з головних переваг Grafana є її здатність з’єднуватися з різними джерелами даних без необхідності їхнього попереднього об’єднання. Це дозволяє користувачам працювати одночасно з декількома базами даних та системами, створюючи єдину панель моніторингу. Крім того, Grafana має інтуїтивний інтерфейс, який полегшує створення налаштованих дашбордів і візуалізацій, що відповідають потребам конкретного користувача чи команди.

Ще однією визначною особливістю є можливість налаштовувати алерти. Grafana дозволяє створювати сповіщення, які інформують користувачів про можливі проблеми чи аномалії в даних, що є важливим для оперативного реагування на зміни. Гнучкість цієї системи також проявляється у великій кількості доступних плагінів, які розширюють функціональність платформи та адаптують її до різних галузей застосування.

Grafana підтримує розподіл прав доступу, що дозволяє забезпечувати безпеку даних і контролювати, хто може переглядати чи редагувати інформацію. Платформа також оптимізована для роботи у великих проєктах і може легко масштабуватися у відповідності до зростаючих вимог бізнесу. Завдяки відкритому

коду, спільнота Grafana постійно розробляє нові функції та вдосконалення, що робить її актуальною навіть у швидко змінюваних умовах.

Переваги Grafana можна підсумувати у таких десяти основних тезах:

– гнучкість у підключенні до різноманітних джерел даних;

– інтуїтивний інтерфейс для створення дашбордів і графіків;

– можливість налаштування алертів для моніторингу стану системи;

– підтримка великої кількості плагінів для розширення функціональності;

– розподіл прав доступу для забезпечення безпеки;

– масштабованість для роботи у великих проєктах або командах;

– інтеграція з хмарними платформами та популярними рішеннями;

– підтримка активної спільноти розробників для постійного вдосконалення;

– платформа з відкритим кодом, що дозволяє кастомізацію відповідно до потреб;

– висока продуктивність навіть при обробці великих обсягів даних.

Grafana є потужним інструментом для роботи з даними, який відзначається гнучкістю у підключенні до різноманітних джерел, забезпечуючи користувачів можливістю інтегрувати свої дані з багатьох систем і платформ. Одна з головних переваг цього інструмента – його інтуїтивний інтерфейс, який дозволяє легко створювати дашборди і графіки для візуалізації складної аналітичної інформації. Це робить Grafana доступним як для новачків, так і для досвідчених фахівців з аналізу даних.

Додатково, платформа пропонує можливість налаштування алертів, що дає змогу оперативно реагувати на будь-які зміни або потенційні проблеми в системі. Це функція, що особливо цінується при моніторингу критично важливих систем, адже вона дозволяє автоматизувати сповіщення про відхилення від норми.

Grafana підтримує велику кількість плагінів, які суттєво розширюють її функціональність, дозволяючи користувачам адаптувати інструмент під конкретні потреби проєкту чи організації. Важливим аспектом безпеки є можливість

розподілу прав доступу, що є надважливим для роботи з даними у великих командах або організаціях з підвищеними вимогами до конфіденційності.

Масштабованість є ще однією ключовою характеристикою Grafana, яка дозволяє використовувати платформу як у невеликих проєктах, так і у великих корпоративних середовищах. Завдяки інтеграції з хмарними платформами і популярними рішеннями, Grafana легко вписується в сучасну екосистему аналітичних та IT-інструментів.

Активна спільнота розробників є ще одним важливим чинником успіху Grafana. Вона не лише пропонує підтримку для користувачів, але й сприяє постійному вдосконаленню платформи. Оскільки Grafana є платформою з відкритим кодом, це відкриває широкі можливості для кастомізації відповідно до специфічних потреб користувачів.

Окрім цього, Grafana демонструє високу продуктивність навіть при роботі з великими обсягами даних, що робить її універсальним і надійним рішенням для аналітики та візуалізації у будь-якому масштабі. Завдяки цим характеристикам, платформа стала одним із найпопулярніших інструментів у своїй категорії, відповідаючи навіть найвищим вимогам до гнучкості, інтеграції та ефективності.

Prometheus - це широко використовувана система моніторингу та збору метрик із відкритим кодом, яка стала незамінною складовою інфраструктури багатьох IT-компаній. Система була розроблена з акцентом на масштабованість, надійність і доступність, що робить її ідеальним вибором для моніторингу як невеликих, так і великих проєктів. Prometheus особливо популярний у хмарних середовищах і у мікросервісній архітектурі завдяки своїй гнучкості та здатності працювати з великими обсягами даних у реальному часі.

Однією з головних переваг Prometheus є його інтеграція з багатьма іншими платформами та інструментами, такими як Kubernetes, Grafana та інші системи моніторингу. Prometheus дозволяє ефективно збирати, зберігати та аналізувати дані про стан систем, що дає змогу оперативно виявляти проблеми, запобігати збої та оптимізувати роботу.

Prometheus використовує зручний для аналітики формат даних із підтримкою потужної мови запитів PromQL, яка дозволяє користувачам гнучко виконувати пошук і аналіз даних. Крім того, система працює незалежно від зовнішніх залежностей, що забезпечує її автономність та стабільність. Це особливо важливо для критично важливих сервісів, які потребують постійного моніторингу.

Отже, Prometheus має низку ключових переваг:

– інтеграція з популярними платформами, такими як Kubernetes і Grafana;

– використання потужної мови запитів PromQL для гнучкого аналізу даних;

– автономність і відсутність залежності від зовнішніх баз даних;

– стабільна робота з великими обсягами даних у реальному часі;

– масштабованість і можливість адаптації під проєкти різного рівня складності.

Prometheus – це система моніторингу та збору метрик із відкритим кодом, яка стала значущим інструментом для багатьох IT-компаній завдяки своїм унікальним характеристикам і можливостям. Розроблена з акцентом на масштабованість, надійність і доступність, вона надає ефективне рішення для моніторингу як невеликих, так і надзвичайно великих проєктів. Особливої популярності Prometheus набув у хмарних середовищах та мікросервісній архітектурі, де гнучкість і здатність обробляти великі обсяги даних у реальному часі є критично важливими.

Однією з ключових переваг Prometheus є його інтеграція з іншими популярними платформами та інструментами, такими як Kubernetes, Grafana та різноманітні системи моніторингу. Завдяки такій інтеграції користувачі отримують можливість створювати комплексні екосистеми для управління та аналізу стану систем. Prometheus дозволяє ефективно збирати дані про параметри роботи системи, зберігати їх у зручному для аналітики форматі та проводити детальний аналіз. Ця функція стає особливо важливою для оперативного виявлення проблем, запобігання збоям та оптимізації роботи систем.

Особлива увага заслуговує мова запитів PromQL, яка є невід'ємною складовою Prometheus. PromQL надає користувачам гнучкий інструмент для пошуку і аналізу даних, дозволяючи виконувати складні запити та отримувати глибокі аналітичні дані про стан систем. Завдяки цьому, система стає незамінною для тих, хто потребує детального аналізу показників та оперативного реагування на зміни в роботі інфраструктури.

Prometheus також забезпечує автономність, оскільки не залежить від зовнішніх баз даних чи інших систем. Це важливий фактор для критично важливих сервісів, які потребують постійного моніторингу. Незалежність системи дозволяє їй працювати стабільно навіть у випадках, коли зовнішні ресурси недоступні, що додає їй надійності та гнучкості в експлуатації. Крім того, Prometheus здатний обробляти великі обсяги даних у реальному часі, що робить її придатною для проєктів різного рівня складності та масштабу.

Таким чином, Prometheus є потужним, надійним і гнучким рішенням для моніторингу та збору метрик. Її можливості охоплюють інтеграцію з іншими платформами, використання потужної мови запитів, автономність і стабільність у роботі з масивами даних, а також здатність до масштабування. Завдяки цьому система стала невід'ємною частиною інфраструктури багатьох сучасних IT-компаній, забезпечуючи їх ефективну роботу та надійний моніторинг.

Docker є одним із найпопулярніших інструментів для контейнеризації, який революціонізував підхід до розробки, тестування та розгортання програмного забезпечення. Його функціональність дозволяє розробникам і адміністраторам ефективно управляти додатками в ізольованих середовищах, забезпечуючи високу гнучкість та масштабованість. Основна ідея Docker полягає в створенні контейнерів, які є легкими та незалежними від операційної системи вузла. Це відкриває нові можливості для оптимізації процесів DevOps.

Обговоримо переваги Docker.

Docker пропонує широкий спектр переваг, які роблять його незамінним інструментом для сучасного програмного забезпечення. Нижче наведено список основних характеристик, які виділяють Docker серед інших технологій:

– легкість контейнеризації: Docker дозволяє створювати та управляти контейнерами з мінімальними ресурсами, забезпечуючи їхню високу продуктивність;

– портативність: Контейнери Docker можуть бути перенесені між різними операційними системами та середовищами без зміни їх конфігурації;

– висока масштабованість: Docker забезпечує швидке розгортання контейнерів, дозволяючи ефективно масштабувати додатки залежно від потреб;

– автоматизація: Інтеграція з такими інструментами, як Jenkins, спрощує CI/CD процеси, автоматизуючи розробку та тестування;

– ізоляція середовищ: Docker гарантує ізоляцію контейнерів, що запобігає конфліктам між додатками та залежностями;

– швидке розгортання: Використання образів Docker значно скорочує час розгортання нових додатків;

– економія ресурсів: Контейнери Docker використовують менш обсяги оперативної пам’яті та CPU порівняно з традиційними віртуальними машинами;

– простота інтеграції: Docker легко інтегрується з хмарними платформами, такими як AWS, Azure та Google Cloud;

– швидке масштабування: Завдяки можливості швидкого додавання або видалення контейнерів, Docker полегшує масштабування додатків;

– ефективне тестування: Забезпечує можливість створення тестових середовищ, які ідентичні продуктивним;

– зниження витрат: Оскільки Docker використовує менше ресурсів, це веде до економії коштів на інфраструктурі;

– уніфікованість середовища: Контейнери забезпечують однакове середовище для розробки, тестування та продуктивної роботи;

– гнучкість: Можливість створення складних мікросервісних архітектур із легкими контейнерами;

– широка спільнота: Docker має активну спільноту користувачів, яка забезпечує доступ до документації, підтримки та значної кількості готових образів. Docker стає важливим інструментом для забезпечення ефективності, надійності та безпеки в розробці програмного забезпечення, роблячи його ключовим елементом сучасної DevOps культури.

Docker зарекомендував себе як потужний і ефективний інструмент для розробників та IT-фахівців, що забезпечує значні покращення в управлінні додатками та інфраструктурою. Основною перевагою Docker є його здатність до швидкого розгортання додатків, оскільки використання контейнерів дозволяє швидко створювати і запускати нові додатки без потреби в складних налаштуваннях або тривалих процесах конфігурації системи. Це значно скорочує час, який зазвичай витрачається на розгортання.

Контейнери Docker є надзвичайно економічними з точки зору використання ресурсів. Вони потребують менше оперативної пам'яті та CPU порівняно з традиційними віртуальними машинами, що дозволяє оптимізувати використання обладнання, не жертвуючи продуктивністю. Такий підхід також сприяє зниженню витрат на підтримку інфраструктури.

Ще одним важливим аспектом Docker є його простота інтеграції з різними хмарними платформами, такими як AWS, Azure та Google Cloud. Це дає змогу легко налаштовувати та підтримувати хмарні додатки, що є ключовим для сучасних технологій. Крім того, контейнеризація забезпечує швидке масштабування, дозволяючи оперативно додавати або видаляти контейнери в залежності від потреби. Це забезпечує гнучкість у реагуванні на зміну вимог до продуктивності та навантаження.

Docker також надає можливість створення тестових середовищ, які ідентичні продуктивним. Завдяки цьому розробники можуть ретельно протестувати додатки

з мінімальним ризиком неочікуваних проблем у реальному використанні. Це сприяє підвищенню надійності та стабільності програмного забезпечення.

Контейнери Docker забезпечують уніфікованість середовища для всіх етапів роботи з програмним забезпеченням – від розробки до тестування й продуктивної роботи. Це вирішує проблему різниці в налаштуваннях середовищ, яка часто призводить до помилок у роботі додатків.

Особливо варто відзначити гнучкість Docker, яка дозволяє створювати складні мікросервісні архітектури, використовуючи легкі контейнери. Це сприяє поширенню мікросервісного підходу в розробці програмного забезпечення, який стає дедалі популярнішим завдяки своїй масштабованості та модульності.

На додаток до технологічних переваг, Docker має широку спільноту користувачів, яка активно підтримує інструмент, створює документацію, ділиться досвідом та надає доступ до великої кількості готових образів, що значно полегшує роботу розробників.

Таким чином, Docker стає важливим елементом сучасної DevOps культури, забезпечуючи ефективність, надійність та безпеку в розробці програмного забезпечення. Його переваги охоплюють не лише технічні аспекти, але й економічні та організаційні, що робить його ідеальним рішенням для бізнесів будь-якого масштабу.

ESP32 - це один із найпопулярніших мікроконтролерів виробництва компанії Espressif Systems, що здобув велику популярність завдяки своїм численним технічним перевагам та широким можливостям використання. Цей мікроконтролер є високопродуктивним, компактним і універсальним рішенням, яке об'єднує Wi-Fi та Bluetooth функціональність в одному пристрої, роблячи його ідеальним для розробки IoT проєктів.

ESP32 відзначається рядом важливих переваг, які роблять його ідеальним вибором для розробників:

– широкий набір комунікаційних функцій: підтримка Wi-Fi і Bluetooth дозволяє легко інтегрувати ESP32 у бездротові мережі;

– висока продуктивність: двоядерний процесор Tensilica Xtensa LX6 забезпечує швидке виконання завдань;

– низьке енергоспоживання: спеціальні режими енергозбереження, такі як Deep Sleep, дозволяють оптимізувати використання енергії;

– широкий вибір периферії: підтримка ADC, DAC, PWM, SPI, I2C, UART – це лише частина функцій, доступних на ESP32;

– можливість збору даних: вбудовані датчики температури і сенсори для збору даних у реальному часі;

– масштабованість: легка інтеграція у великі IoT системи та можливість роботи з кількома пристроями одночасно;

– відкритість для кастомізації: підтримка програмування через популярні середовища, такі як Arduino IDE;

– компактність: невеликий розмір мікроконтролера дозволяє використовувати його навіть у найменших пристроях;

– стабільність роботи: ESP32 демонструє надійність навіть у складних умовах експлуатації;

– доступність: багатофункціональний пристрій за конкурентну ціну.

Обговорення вибору ESP32 від Espressif.

Вибір ESP32 часто стає очевидним для розробників завдяки його універсальності та функціональності. Поєднання Wi-Fi і Bluetooth у компактному форматі відкриває можливості для створення складних IoT рішень без необхідності застосування додаткових модулів. Крім того, підтримка програмування через популярні середовища суттєво спрощує розробку навіть для новачків.

ESP32 демонструє себе як високопродуктивний, універсальний і надійний мікроконтролер, що є ідеальним вибором для створення IoT проєктів різного масштабу. Його переваги включають широкий набір функцій, низьке енергоспоживання, простоту інтеграції та доступність за ціною. Завдяки підтримці технологій Wi-Fi і Bluetooth, ESP32 забезпечує гнучкість у розробці пристроїв, які можуть працювати в бездротових мережах. Компанія Espressif Systems продовжує

підтримувати інновації у сфері IoT, надаючи розробникам інструменти, які відповідають сучасним вимогам до мобільності, продуктивності та масштабованості. Обираючи ESP32, розробники отримують надійний і функціональний мікроконтролер, який дозволяє створювати високотехнологічні рішення для розумних пристроїв, автоматизації та інших сфер.

2.5 Проєктування бази даних

Проєктування бази даних є однією з ключових стадій створення сучасних інформаційних систем, адже саме від її структури залежить ефективність, швидкість та надійність обробки даних, на яких ґрунтується робота будь-якого програмного забезпечення, цифрового продукту чи сервісу. Цей процес охоплює ретельне вивчення потреб користувача, визначення цілей системи та створення оптимальної моделі даних, яка забезпечує логічну організацію інформації та її легкий доступ.

Важливість проєктування баз даних важко недооцінити, адже погано спроєктована структура може призводити до затримок у роботі системи, конфліктів даних чи навіть до їхньої втрати, що ставить під загрозу функціональність і безпеку всього проєкту. На цьому етапі визначаються класи об'єктів, їхні властивості, зв'язки між ними, а також правила, які регулюють взаємодію цих елементів. Правильне моделювання даних допомагає уникнути надмірного дублювання, забезпечити цілісність інформації та створити механізми, що дозволяють масштабувати систему відповідно до зростання обсягу даних або кількості користувачів.

Одним із ключових аспектів проєктування бази даних є вибір відповідної моделі даних - реляційної, об’єктно-орієнтованої, документної чи іншої, залежно від специфіки завдань, які має виконувати інформаційна система. При цьому кожна модель має свої переваги та недоліки, які повинні бути враховані при створенні архітектури бази. Реляційна модель, наприклад, є однією з найпоширеніших

завдяки своїй простоті, універсальності та підтримці стандартизованих мов запитів, таких як SQL. Водночас об’єктно-орієнтовані моделі можуть бути краще інтегровані з сучасними мовами програмування, а документні бази даних дозволяють ефективно працювати з неструктурованою інформацією.

Проєктування баз даних також включає визначення індексів, механізмів резервного копіювання, шифрування та інших технічних аспектів, які впливають на її продуктивність та безпеку. Наприклад, використання індексів дозволяє значно прискорити пошук інформації у великих масивах даних, тоді як належна система резервного копіювання забезпечує збереження критичної інформації у разі технічних збоїв або кібератак. Шифрування даних, у свою чергу, має вирішальне значення для забезпечення конфіденційності та захисту інформації від несанкціонованого доступу.

Крім того, на етапі проєктування особлива увага приділяється нормалізації даних - процесу упорядкування структури бази для усунення надмірного дублювання та забезпечення логічної взаємодії між таблицями. Нормалізація сприяє оптимізації використання ресурсів, підвищує ефективність виконання запитів та сприяє легшому обслуговуванню бази даних у майбутньому.

Важливість правильного проєктування баз даних також проявляється у можливості інтеграції системи з іншими платформами або застосунками, що є особливо важливим у сучасній епоху цифровізації, коли різні сервісні системи повинні працювати у тісній взаємодії одна з одною. Належна структура бази дозволяє забезпечити гнучкість і адаптивність системи, що критично важливо для її успішного функціонування в умовах змінних вимог бізнесу чи технологічного середовища.

Таким чином, проєктування баз даних є не лише технічною задачею, але й стратегічним етапом, який визначає якість, надійність та довговічність інформаційної системи. Надійно спроєктована база даних стає основою для ефективного управління даними, їхнього аналізу та використання, сприяючи

успішній реалізації бізнес-цілей, наукових досліджень чи інших масштабних проєктів.

Спроєктована база даних зображена на рисунку 2.1..

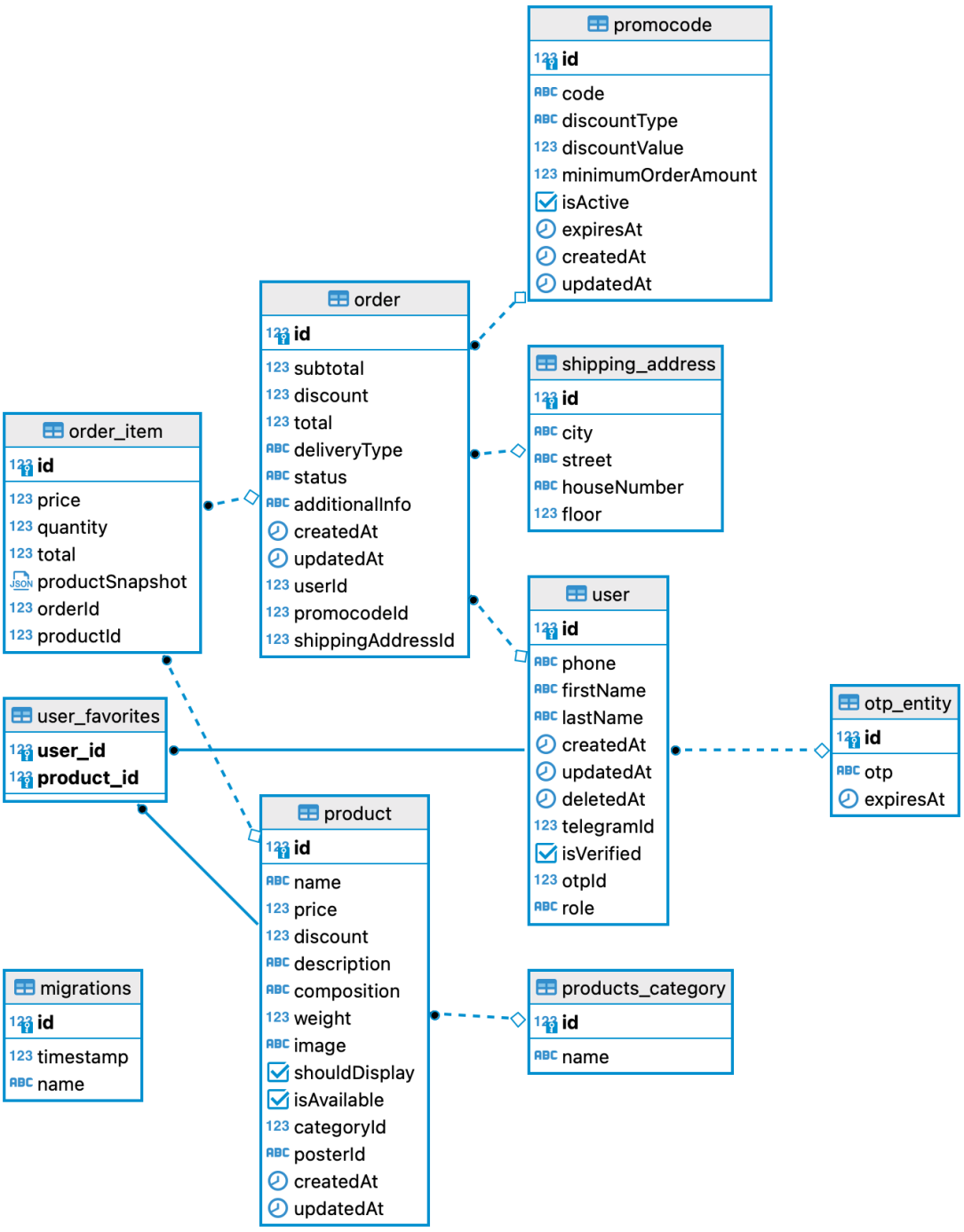


Рисунок 2.1 – ER діаграма структури спроєктованої бази даних

3 РЕЛАЛІЗАЦІЯ ПРОГРАМНО-ТЕХНІЧНОГО ЗАСОБУ

3.1 Елементи програмно-технічного комплексу

Для проєкту використаю мікроконтроллер від компанії Espressif, а саме ESP32-S3 N16R8.

Модуль розробника ESP32-S3 N16R8 побудований на мікромодулі ESP32-S3 N16R8 – новому мініатюрному високопродуктивному суміщеному Wi-Fi + BT + BLE модулі від компанії Espressif, призначеному для широкого спектру застосувань, починаючи від мікропотужних датчиків. 3 кодування. На модулі зібрано всю необхідну мінімальну периферію, достатню для швидкого та комфортного старту роботи з ESP32-S3 N16R8.

Мікромодуль виконаний на базі популярного двоядерного чіпсету ESP32-S3 зі змінною тактовою частотою від 80 МГц до 240 МГц, можливістю індивідуального керування та живлення. Модуль розроблений для переносної та автономної електроніки та додатків інтернет-речей, виконаний у мініатюрному корпусі 25.5 мм x 18 мм, має на борту Flash пам'ять, додаткову оперативну пам'ять, кварц 40 МГц та PCB антену, що забезпечує відмінні RF характеристики.

Застосований мікромодуль має багату периферію, що включає такі інтерфейси як USB, UART, SPI, I²C, I²S, інтерфейс для SD карти, інфрачервоний порт, інтерфейс для підключення ємнісної сенсорної панелі. Однією з особливістю модуля є наднизьке споживання і гнучкий вибір режимів, що «сплять», що дозволяють отримати цифри до 20мкА (deep sleep mode). Модуль підтримує весь стек протоколів стандартів WiFi 802.11n та BT4.2, забезпечуючи цей функціонал через інтерфейси SPI/SDIO або I²C/UART.

Принципова схема даного модуля наведениа на рисунку 3.1.

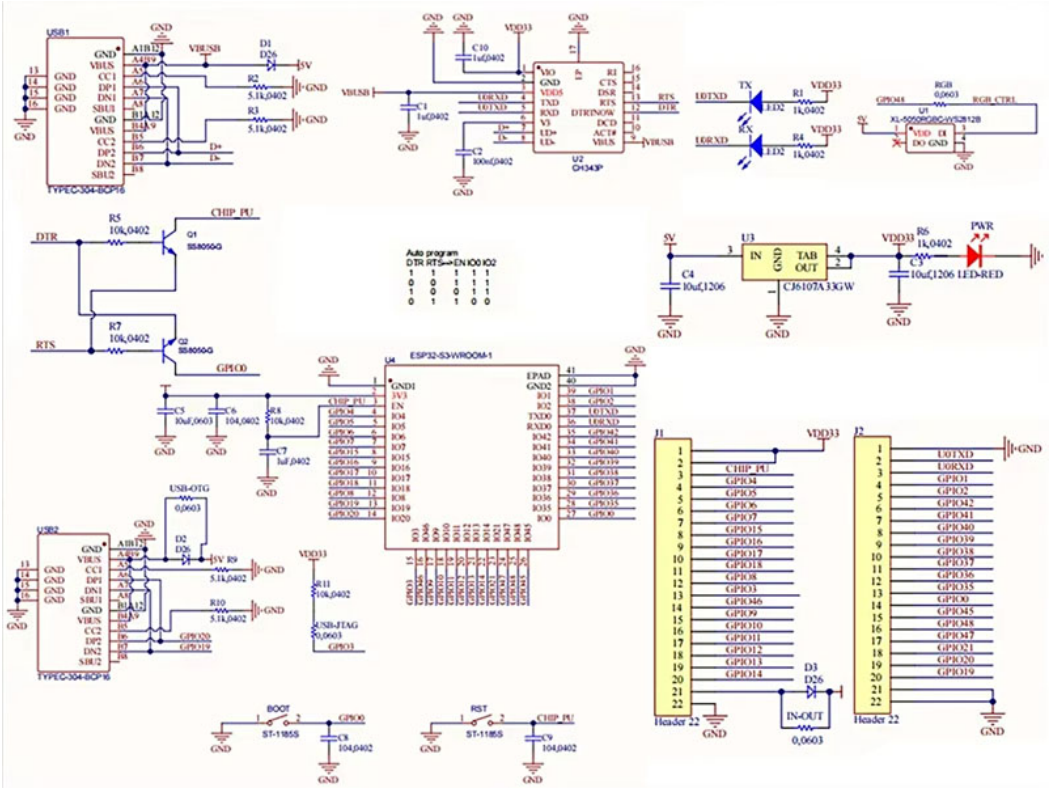


Рисунок 3.1 – Принципова схема модуля ESP32-S3 N16R8

Даний модуль базується на мікроконтролері ESP32-S3-WROOM-1U. Принципова схмема даного мікроконтролера наведена на рисунку 3.2.

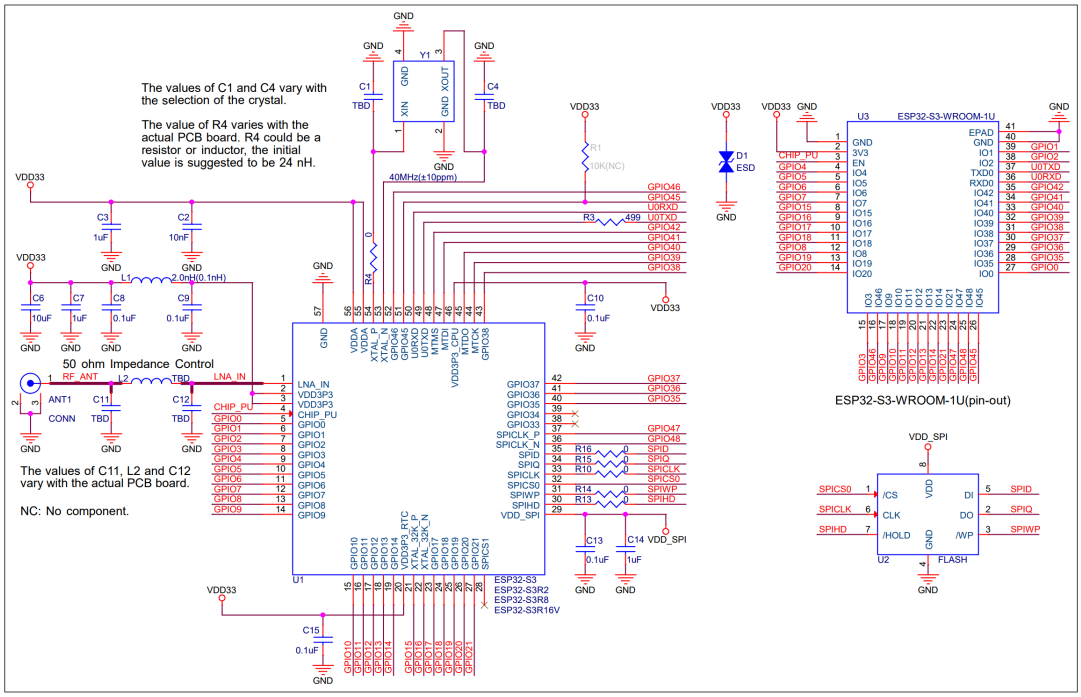


Рисунок 3.2 – Принципова схемамікроконтролера мікроконтроллері ESP32-S3-WROOM-1U.

Фізичні розміри мікроконтролера складають 18мм в ширину і 25.5мм в довжину. Деталі наведені на рисунку 3.3.

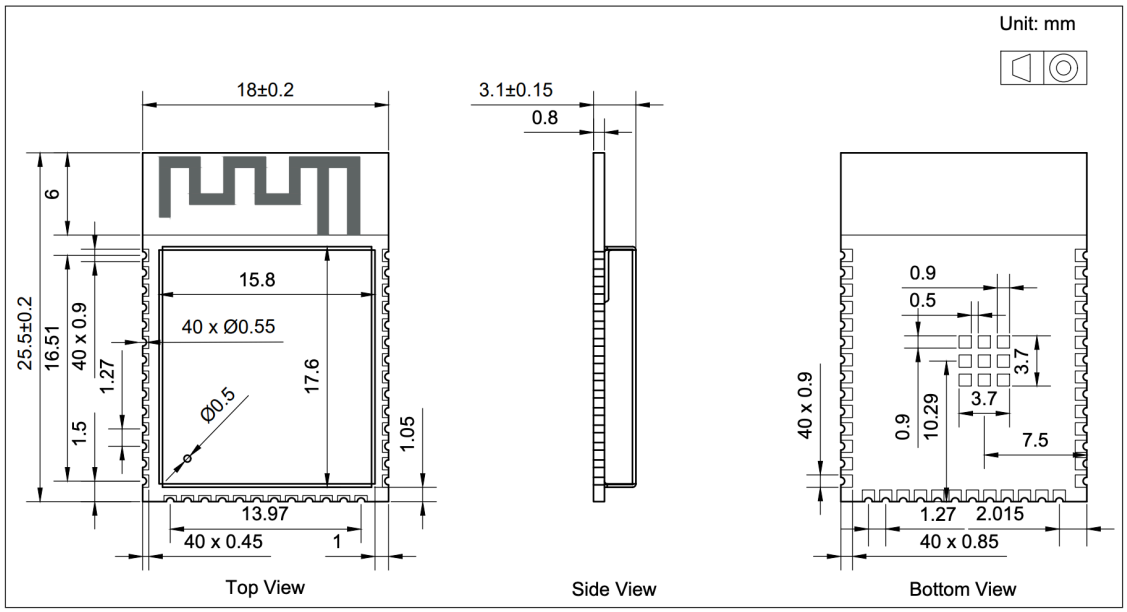


Рисунок 3.3 – Фізичні розміри мікроконтролера ESP32-S3-WROOM-1U.

Схема друкованої плати мікроконтролера ESP32-S3-WROOM-1U наведена на рисунку 3.4.

Дисплей використаний в програмно-апаратному комплексі є шилд від компанії Waveshare, а саме Waveshare 4inch TFT Touch Shield.

Він використовує LCD контролер ILI9486 та контролер для обробки доторків резистивного екрану XPT2046. Має роздільну здатність в 320x480 пікселів та кольоровою градацією в 65536 кольорів.

Шилд підтримує 2 інтерфейси – ICSP та SPI. Режим можна перемкнути парамикачем на задній частину шилду. Для для свого програмно-апаратного комплексу я обрав роботу через SPI інтерфейс, тому перемнув шилд в цей режим, оскільки за замовчуванням використовується ICSP інтерфейс.

Принципова схема LCD контлера ILI9486 зображена на рисунку 3.5.

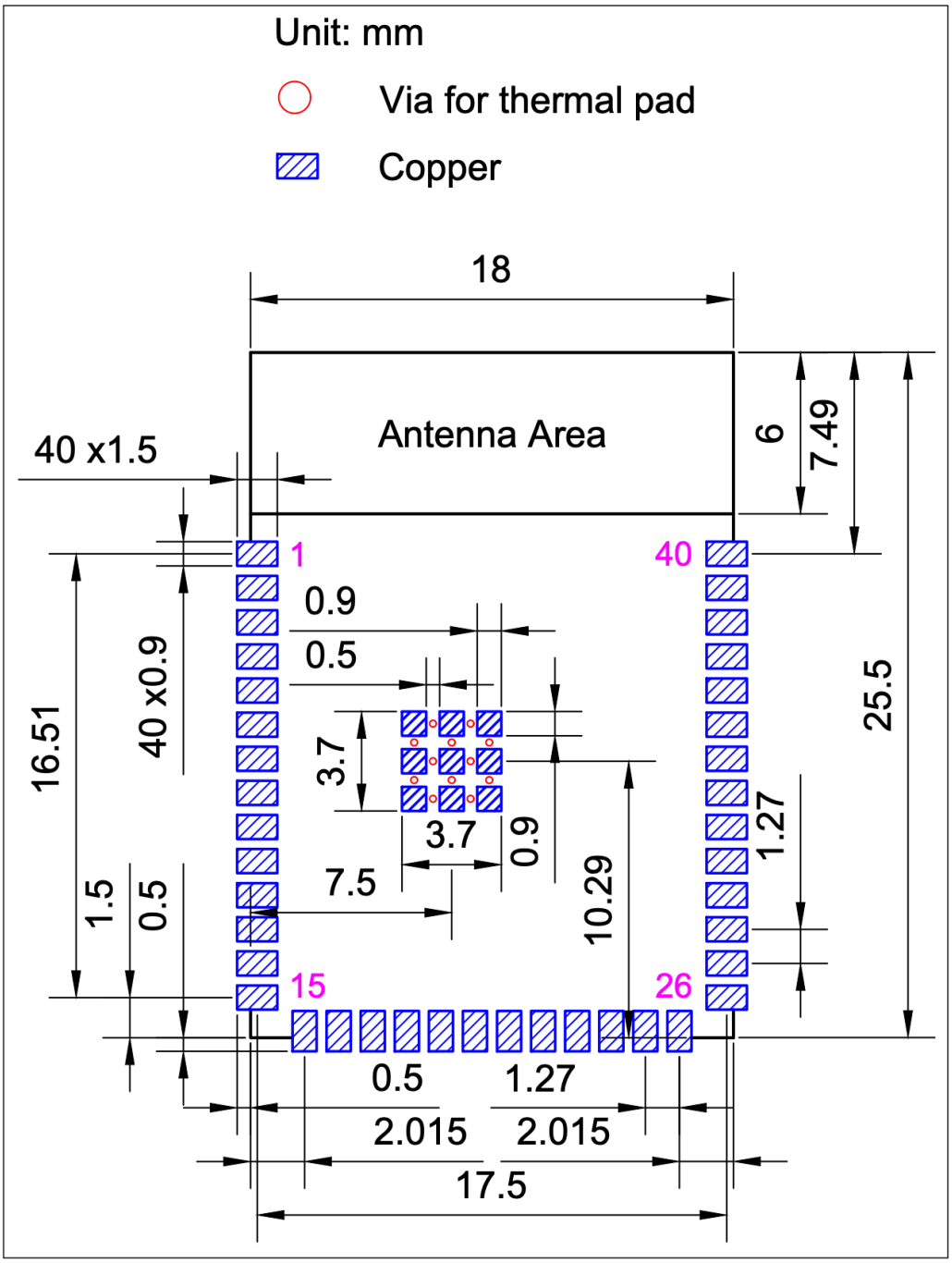


Рисунок 3.4 – Схема друкованої плати мікроконтролера ESP32-S3-WROOM-1U.

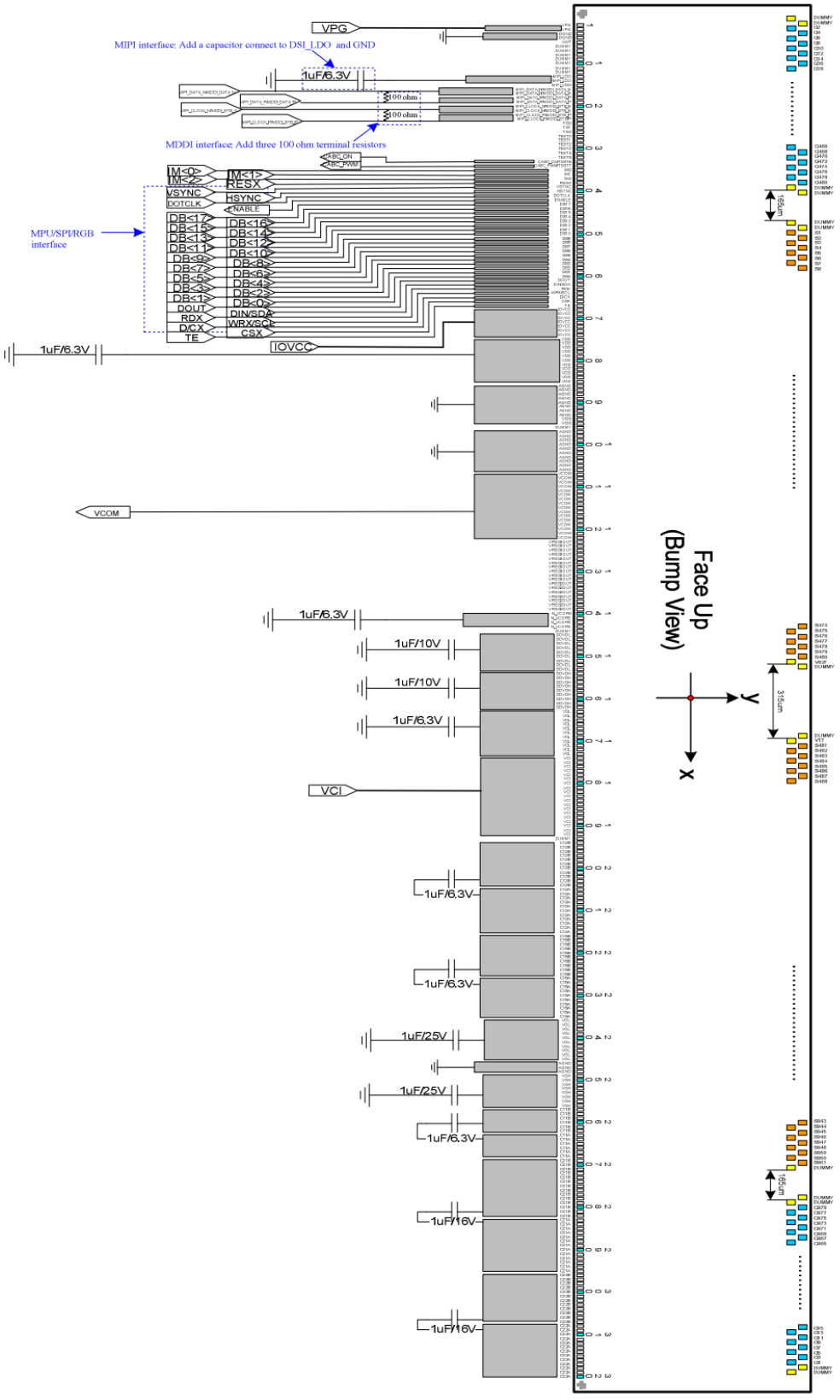


Рисунок 3.5 – Принципова схема ILI9486.

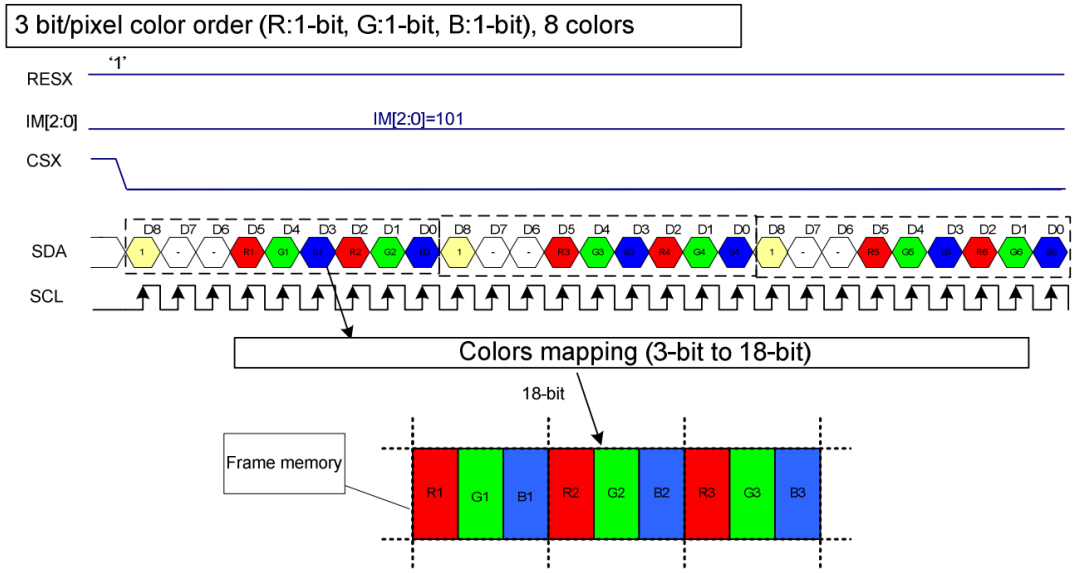


Рисунок 3.6 – Схема роботи трьохлінійного варіанту передачі даних контролеру ILI9486.

Узагальнена діаграма схеми контролеру сенсорного екрану XPT2046 наведена на рисунку 3.7, а принципова схема на рисунку 3.8.

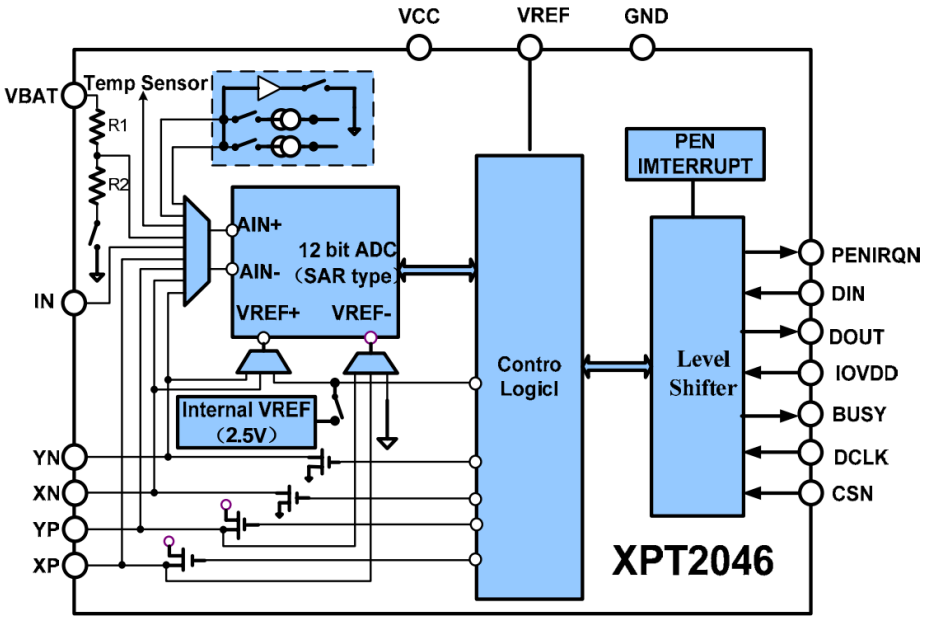


Рисунок 3.7 – Узагальнена діаграма схеми контролеру сенсорного екрану XPT2046.

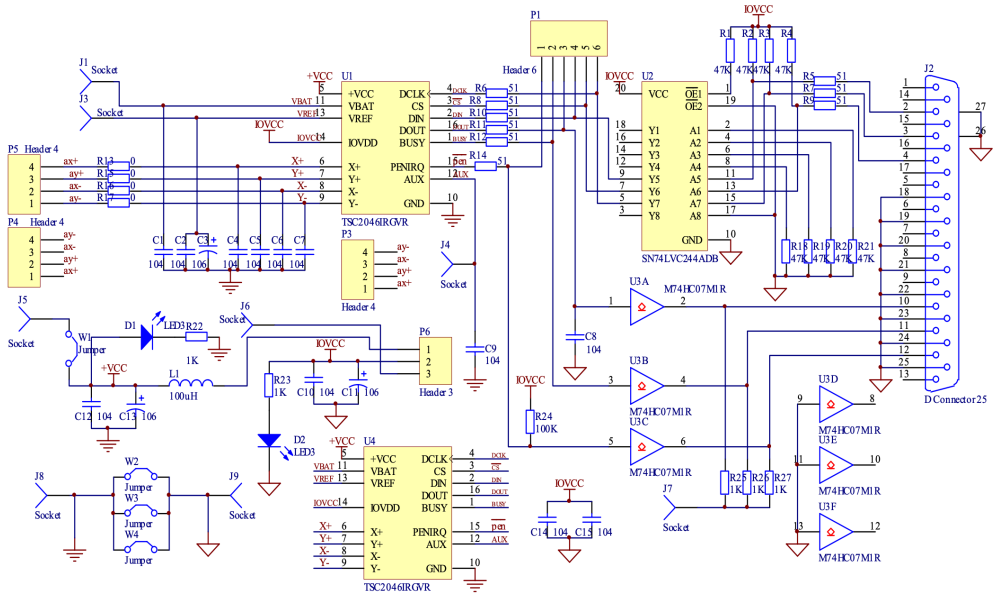


Рисунок 3.8 – Принципова схема контролеру сенсорного екрану XPT2046.

3.2 Бекенд частина

Для розробки бекенду використано NestJS який використовує ExpressJS, який в свою чергу використовує NodeJS. Також проєкт має список залежностей сторонніх бібліотек для досягнення цілей проєкту.

Список залежностей:

- @nestjs/axios : ^4.0.0,
- @nestjs/common : ^11.0.10,
- @nestjs/config : ^4.0.0,
- @nestjs/core : ^11.0.10,
- @nestjs/event-emitter : ^3.0.0,
- @nestjs/jwt : ^11.0.0,
- @nestjs/passport : ^11.0.5,
- @nestjs/platform-express : ^11.0.10,
- @nestjs/platform-socket.io : ^11.0.10,
- @nestjs/platform-ws : ^11.0.10,
- @nestjs/schedule : ^5.0.1,

- @nestjs/swagger : ^11.0.4,
- @nestjs/throttler : ^6.4.0,
- @nestjs/typeorm : ^11.0.0,
- @nestjs/websockets : ^11.0.10,
- axios : ^1.7.2,
- bcryptjs : ^2.4.3,
- class-transformer : ^0.5.1,
- class-validator : ^0.14.1,
- crypto : ^1.0.1,
- helmet : ^7.1.0,
- passport : ^0.7.0,
- passport-jwt : ^4.0.1,
- pg : ^8.12.0,
- querystring : ^0.2.1,
- reflect-metadata : ^0.2.2,
- rimraf : ^6.0.1,
- rxjs : ^7.8.1,
- socket.io : ^4.8.1,
- typeorm : ^0.3.20,
- ws : ^8.18.1.

Проєкт також має базу даних PostgreSQL як залежність. Для зручності роботи з базою проєкт використовує ORM(Object Relational Mapping), а саме вибір був зроблений в користь TypeORM оскільки це продуктивний гнучний варіант зі зручним інтерфейсом, котрий надає можливість лаконічно визначати сутності, писати міграції, і працювати з даними.

Розроблена схема бази даних наведена на рисунку 2.1. Вона задовільняє вимоги проєкту.

3.3 Взаємодія ESP32 з бекенд частиною

Архітектура взаємодії між програмно-технічним комплексом на базі мікроконтролера ESP32 та серверним бекендом представляє собою гібридну модель комунікації, що поєднує переваги постійного з'єднання через WebSocket протокол із класичними HTTP запитами для обміну структурованими даними. Така архітектурна концепція забезпечує оптимальний баланс між реактивністю системи та ефективністю використання мережевих ресурсів у контексті автоматизованого прийому та оповіщення про замовлення в ресторані східної кухні.

Початкова фаза встановлення з'єднання між ESP32 та бекендом здійснюється через WebSocket протокол, який забезпечує повнодуплексну комунікацію між клієнтом та сервером. При ініціалізації з'єднання мікроконтролер ESP32 формує WebSocket запит, до якого включається query параметр `token`, що служить механізмом автентифікації та авторизації пристрою в системі. Бекенд, отримавши запит на встановлення WebSocket з'єднання, виконує валідацію переданого токена шляхом порівняння його з еталонним значенням, збереженим у конфігураційних налаштуваннях системи. Важливою особливістю реалізованої схеми автентифікації є гнучкість конфігурації, за якої відсутність токена в налаштуваннях бекенду інтерпретується як дозвіл на встановлення з'єднання з будь-яким переданим токеном, що спрощує процес налаштування системи в тестових або демонстраційних режимах.

Після успішної верифікації токена та встановлення WebSocket з'єднання мікроконтролер ESP32 ініціює процес підписки на події зміни статусів замовлень. Ця підписка має селективний характер, оскільки ESP32 налаштовується на отримання повідомлень лише про ті статуси замовлень, які безпосередньо стосуються кухонних операцій та релевантні для конкретного місця розташування пристрою. Така фільтрація на рівні бекенду забезпечує мінімізацію мережевого трафіку та зменшує навантаження на обчислювальні ресурси мікроконтролера,

оскільки ESP32 не отримує інформацію про статуси, які не потребують його реакції.

Механізм оповіщення про нові замовлення реалізований через асинхронну передачу подій від бекенду до ESP32 засобами WebSocket протоколу. Коли в системі з'являється нове замовлення або змінюється статус існуючого замовлення на один із тих, що відслідковуються пристроєм, бекенд формує подію та передає її через встановлене WebSocket з'єднання. ESP32, отримавши таку подію, ініціює додатковий HTTP запит до бекенду для отримання детальної інформації про замовлення. Такий двоетапний підхід дозволяє оптимізувати використання пропускної здатності WebSocket каналу, передаючи через нього лише сигнали про зміни, а детальні дані отримувати за запитом через HTTP протокол.

HTTP запит, який формується ESP32 у відповідь на отриману через WebSocket подію, спрямовується на специфічний endpoint бекенду, призначений для надання інформації про замовлення. Бекенд, обробляючи цей запит, виконує фільтрацію наявних замовлень відповідно до критеріїв релевантності для конкретного пристрою та повертає структурований масив замовлень у форматі JSON. Цей масив містить всю необхідну інформацію для відображення замовлень на сенсорному екрані пристрою, включаючи ідентифікатори замовлень, їх склад, час створення, поточний статус та іншу метаінформацію.

Інтерактивна взаємодія користувача з системою реалізується через сенсорний екран ESP32 пристрою, який відображає отриману від бекенду інформацію про замовлення у зручному для сприйняття форматі. При натисканні користувачем на конкретне замовлення мікроконтролер формує HTTP запит до бекенду, що містить ідентифікатор обраного замовлення та команду на зміну його статусу. Бекенд, отримавши такий запит, виконує валідацію переданих параметрів, перевіряє можливість переходу замовлення до наступного статусу відповідно до визначеної бізнес-логіки та, у разі успішної валідації, оновлює статус замовлення в базі даних.

Архітектура системи передбачає надійну обробку помилок та відновлення з'єднання у випадку тимчасових збоїв мережевої комунікації. ESP32 реалізує механізми автоматичного переп'єднання до бекенду у випадку розриву WebSocket з'єднання, а також retry логіку для HTTP запитів, що забезпечує стабільність роботи системи навіть в умовах нестабільної мережевої інфраструктури. Бекенд, у свою чергу, підтримує механізми graceful shutdown та cleanup для коректного завершення WebSocket з'єднань при перезапуску або оновленні серверного компоненту.

Безпека комунікації забезпечується не лише через токен-based автентифікацію при встановленні WebSocket з'єднання, а й через валідацію всіх вхідних HTTP запитів від ESP32. Бекенд виконує санітизацію та валідацію всіх параметрів запитів, перевіряє права доступу до конкретних операцій зміни статусів замовлень та логує всі критичні операції для можливості аудиту системи. Така багаторівнева система безпеки забезпечує захист від несанкціонованого доступу та маніпуляцій з даними замовлень.

Протокольна реалізація взаємодії між ESP32 та бекендом ґрунтується на детально розробленій схемі повідомлень, яка забезпечує максимальну ефективність передачі даних при мінімальному споживанні ресурсів мікроконтролера. WebSocket з'єднання використовує бінарний формат повідомлень для критичних операцій та JSON для структурованих даних, що дозволяє оптимізувати розмір пакетів та швидкість їх обробки. Кожне повідомлення, що передається через WebSocket канал, містить заголовок з метаданими, включаючи тип повідомлення, timestamp, sequence number та checksum для забезпечення цілісності даних.

Система підтримує розширену типологію статусів замовлень, яка відображає весь життєвий цикл замовлення від моменту його створення до завершення. ESP32 налаштовується на відстеження специфічних статусів, таких як "передано на кухню", "розпочато приготування", "готово до видачі", "потребує уваги кухаря", що дозволяє персоналу кухні отримувати релевантні сповіщення на відповідних

етапах обробки замовлення. Бекенд підтримує гнучку конфігурацію правил переходу між статусами, включаючи умовні переходи, що залежать від типу страв, часу доби, завантаженості кухні та інших бізнес-параметрів.

Механізм кешування на стороні ESP32 реалізований для зменшення частоти HTTP запитів та забезпечення коректної роботи системи при тимчасових перебоях у мережевому з'єднанні. Мікроконтролер зберігає в локальній пам'яті інформацію про останні отримані замовлення, їх статуси та час последнього оновлення, що дозволяє продовжувати відображення актуальної інформації навіть при короткочасних розривах з'єднання. Система кешування використовує алгоритм LRU (Least Recently Used) для управління обмеженим обсягом пам'яті мікроконтролера та автоматично очищає застарілі записи.

Бекенд реалізує складну систему балансування навантаження для обробки одночасних підключень множини ESP32 пристроїв, розміщених у різних зонах ресторану. Кожен пристрій може бути сконфігурований для обслуговування конкретних типів замовлень або кухонних зон, що забезпечує ефективний розподіл робочого навантаження між персоналом. Система підтримує динамічне перенаправлення замовлень між пристроями у випадку виходу з ладу окремих вузлів або зміни конфігурації кухонних зон.

Моніторинг та діагностика системи реалізовані через комплексну систему логування та метрик, яка відстежує продуктивність мережевої комунікації, частоту помилок, час відгуку системи та інші ключові показники. ESP32 періодично передає на бекенд телеметричні дані про стан пристрою, включаючи рівень сигналу WiFi, температуру процесора, використання пам'яті та статус сенсорного екрану. Ці дані агрегуються бекендом та використовуються для проактивного виявлення потенційних проблем та оптимізації роботи системи.

Система підтримує механізм graceful degradation, який дозволяє продовжувати роботу навіть при часткових збоях інфраструктури. У випадку недоступності центрального бекенду ESP32 може переключитися в автономний режим роботи, зберігаючи критичну інформацію локально та синхронізуючи її

після відновлення з'єднання. Бекенд, у свою чергу, підтримує механізм реплікації даних між кількома серверними вузлами для забезпечення високої доступності системи.

Оптимізація споживання енергії ESP32 досягається через реалізацію інтелектуальних режимів енергозбереження, які координуються з бекендом. Система може динамічно регулювати частоту опитування сервера, яскравість дисплею та активність мережевих інтерфейсів залежно від поточної активності замовлень та часу доби. Бекенд надсилає команди управління енергоспоживанням через WebSocket з'єднання, що дозволяє централізовано оптимізувати роботу всіх пристроїв у мережі.

Інтеграція з існуючими системами управління рестораном відбувається через standardized API інтерфейси, які дозволяють бекенду взаємодіяти з POS системами, системами управління запасами, платіжними терміналами та іншими компонентами ресторанної інфраструктури. Така інтеграція забезпечує цілісність даних про замовлення та автоматичну синхронізацію інформації між різними підсистемами ресторану.

Система аналітики та звітності, інтегрована в бекенд, збирає детальну статистику про ефективність кухонних процесів, час обробки замовлень, частоту використання різних функцій ESP32 пристроїв та інші операційні метрики. Ці дані використовуються для генерації аналітичних звітів, які допомагають менеджменту ресторану оптимізувати робочі процеси та підвищувати якість обслуговування клієнтів.

3.4 Реалізація інтерфейсу користувача через сенсорний дисплей

Інтеграція сенсорного дисплея Waveshare 4inch TFT Touch Shield в архітектуру програмно-технічного комплексу на базі ESP32 становить критично важливий компонент системи людино-машинної взаємодії, що забезпечує ефективне відображення інформації про замовлення та інтуїтивне управління

робочими процесами кухні ресторану східної кухні. Даний дисплейний модуль, що працює через високошвидкісний SPI інтерфейс, надає розширені можливості для створення функціонального та ергономічного користувацького інтерфейсу, адаптованого до специфічних потреб ресторанного бізнесу.

Технічна архітектура Waveshare 4inch TFT Touch Shield базується на контролері ILI9486, який забезпечує підтримку дисплея з роздільною здатністю 480x320 пікселів та 16-бітною кольоровою глибиною, що дозволяє відображати до 65536 кольорів одночасно. Така характеристика забезпечує достатню деталізацію для відображення складної інформації про замовлення, включаючи текстові описи страв, числові дані, статусні індикатори та графічні елементи інтерфейсу. Сенсорна панель, інтегрована в конструкцію дисплея, реалізована на базі резистивної технології, яка забезпечує надійне розпізнавання дотиків навіть в умовах підвищеної вологості та температурних коливань, характерних для кухонного середовища.

Комунікація між ESP32 та дисплейним модулем здійснюється через Serial Peripheral Interface (SPI), що представляє собою синхронний послідовний інтерфейс передачі даних, який забезпечує високу швидкість обміну інформацією при мінімальному використанні GPIO пінів мікроконтролера. SPI протокол в даній конфігурації працює в режимі master-slave, де ESP32 виступає в ролі master пристрою, ініціюючи всі транзакції передачі даних, а дисплейний контролер функціонує як slave пристрій, обробляючи отримані команди та дані. Типова швидкість SPI з'єднання для даного дисплея становить до 40 МГц, що забезпечує плавне оновлення графічної інформації без помітних затримок для користувача.

Програмна реалізація взаємодії з дисплеєм базується на спеціалізованих драйверах, які абстрагують низькорівневі операції SPI комунікації та надають високорівневі API для управління графічними примітивами, текстом та сенсорним введенням. Система графічного відображення реалізує буферизоване рендерінг, де графічна інформація спочатку формується в оперативній пам'яті ESP32, а потім передається на дисплей єдиним блоком, що мінімізує мерехтіння та забезпечує

плавність анімацій інтерфейсу. Управління пам'яттю графічного буфера оптимізоване для роботи з обмеженими ресурсами мікроконтролера через використання технік часткового оновлення екрану та компресії графічних даних.

Користувацький інтерфейс, реалізований на дисплеї, структурований як багаторівнева система меню та інформаційних панелей, що адаптована до робочих процесів кухні ресторану. Головний екран відображає актуальний список замовлень у вигляді карток, кожна з яких містить ключову інформацію про замовлення, включаючи номер, час створення, склад страв та поточний статус. Кольорове кодування статусів забезпечує швидке візуальне розпізнавання пріоритетності замовлень, де критичні замовлення виділяються яскравими кольорами, а завершені або відкладені замовлення відображаються приглушеними тонами.

Сенсорна взаємодія реалізована через систему віртуальних кнопок та жестів, що дозволяє персоналу кухні ефективно навігувати інтерфейсом навіть у рукавичках або при вологих руках. Система калібрування сенсорної панелі автоматично адаптується до умов експлуатації та може перекалібровуватися в реальному часі для компенсації температурних деформацій або механічного зношування. Детекція дотиків реалізована з використанням алгоритмів фільтрації шумів та дебаунсингу, що забезпечує надійне розпізнавання користувацьких команд навіть в умовах електромагнітних перешкод від кухонного обладнання.

Оптимізація енергоспоживання дисплейної підсистеми досягається через інтелектуальне управління підсвіткою та режимами роботи контролера. Система автоматично регулює яскравість підсвітки залежно від зовнішнього освітлення та часу доби, а також реалізує режим енергозбереження, який активується при відсутності користувацької активності протягом заданого проміжку часу. Перехід між активним та енергозберігаючим режимами відбувається плавно, без втрати контексту відображуваної інформації.

Інтеграція дисплейної підсистеми з мережевою частиною системи реалізована через асинхронну архітектуру, яка дозволяє одночасно обробляти

оновлення інформації від бекенду та користувацькі події від сенсорної панелі без блокування основного циклу виконання програми. Система подій забезпечує миттєве відображення змін статусів замовлень, отриманих через WebSocket з'єднання, та негайну реакцію на користувацькі дії, що підвищує загальну відзивність системи.

Надійність роботи дисплейної підсистеми забезпечується через систему самодіагностики, яка періодично перевіряє цілісність SPI з'єднання, коректність роботи сенсорної панелі та стан графічного контролера. У випадку виявлення помилок система автоматично ініціює процедури відновлення, включаючи переініціалізацію драйверів та відновлення графічного буфера з резервної копії, що мінімізує час простою обладнання та забезпечує безперервність робочих процесів кухні.

3.5 Демонстрація реалізованого продукту

Прототип апаратної частина у вигляді головних вузлів зображений на рисунку 3.9.

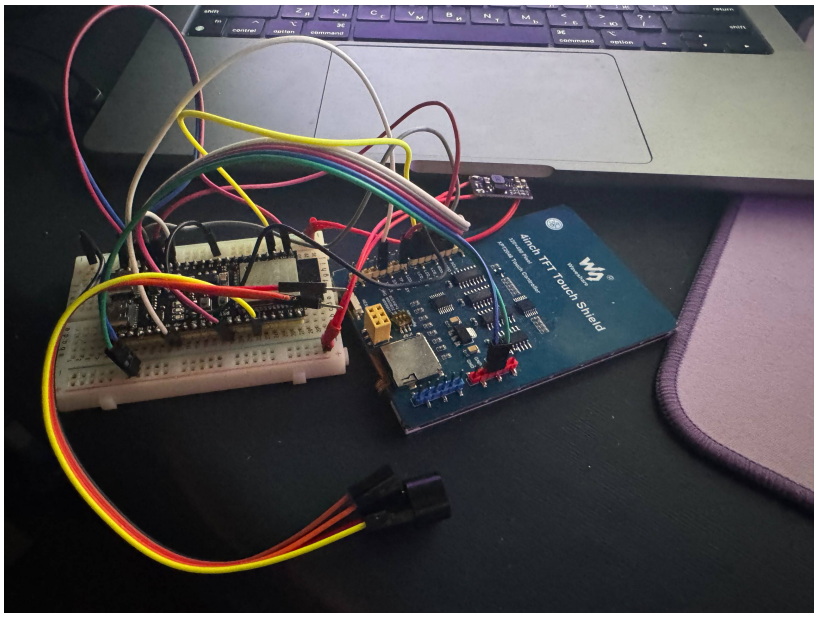


Рисунок 3.9 – Прототип апаратної частини.

Після завантаження прошивки у мікроконтролер ESP32 можна розпочинати огляд інтерфейсу і функціоналу пристрою.

На рисунку 3.10 зображений інтерфейс пристрою після того як він підключився до мережі WiFi та успішно з’єднався з сервером. Конкретно ристунок зображає стан коли немає замовлень які підходять для відображення на кухні закладу.

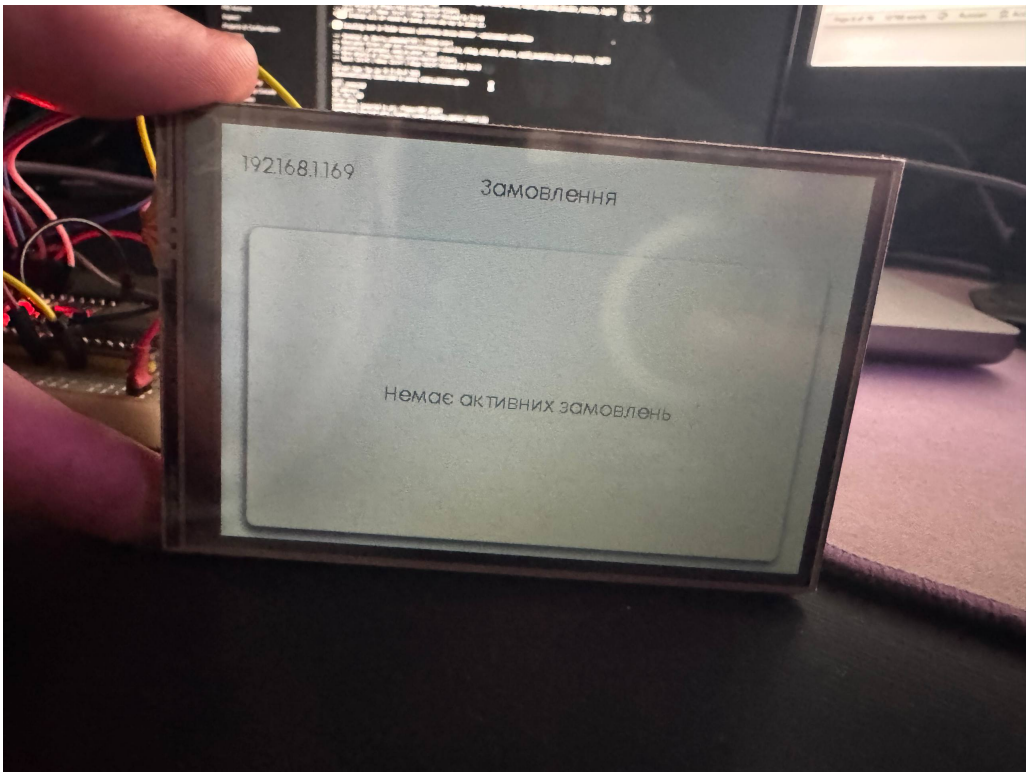


Рисунок 3.10 – Порожній стан користувацького інтерфейсу пристрою.

Якщо ж наразі є замовлення, які мають статус, що підходить для обробки на кухні («підтверджено» або «готується»), то інтерфейс має вигляд зображений на рисунку 3.11. На ньому зображений список замовлень. Кожен елемент надає нступну інформацію про замовлення: унікальний ідентифікатор замовлення, тип («доставка» або «самовивіз»), список страв та їх кількість.

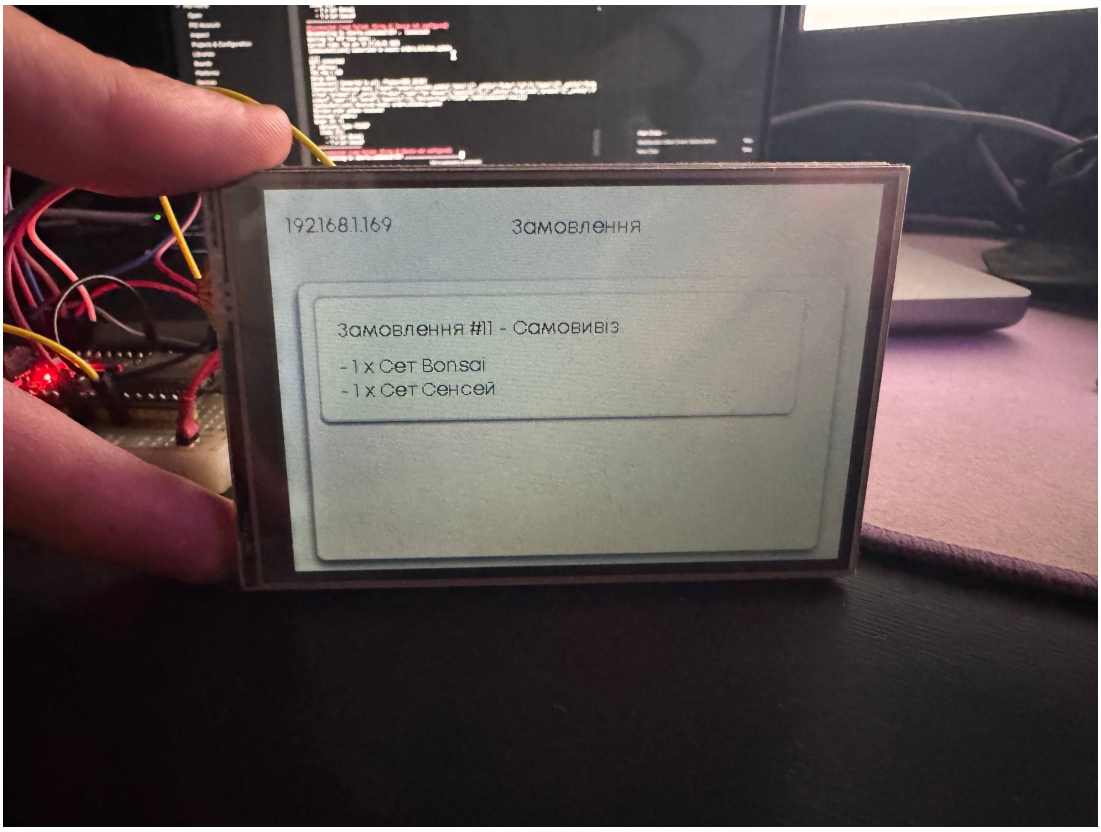


Рисунок 3.11 – Список активних замовлень в користувацькому інтерфейсі пристрою.

При настисканні на замовлення пристрій відправляє запит на бекенд сигналізуючи що кухня обробила його, і що бекенд може надати новий статус замовленню. При першому натисканні замовлення переходить в статус «готується», а при другому настисканні замовлення переходить в статус «готове до видачі» у випадку якщо тип доставки вказаний «самовивіз», або в статус «доставляється» якщо тип доставки вказаний «доставка».

ВИСНОВКИ

У роботі за результатами виконаних теоретичних та практичних досліджень було розроблено програмно-апаратний комплекс для оптимізації процесів прийому і обробки замовлень в ресторані східної кухні.

У першому розділі проведено аналіз аналогів, і визначені конкурентні переваги даного рішення.

У другому розділі проведено планування та проєктування програмно-апаратного комплексу із врахуванням вимог.

У третьому розділі описана технічна та апаратна реалізація програмно-технічного комплексу, описані всі технічні аспекти розробки.

Розроблений програмно-апаратний комплекс демонструє ефективне поєднання сучасних веб-технологій та вбудованих систем для вирішення специфічних завдань автоматизації ресторанного бізнесу. Архітектурне рішення, що базується на мікроконтролері ESP32 з інтегрованим сенсорним дисплеєм Waveshare, забезпечує надійну та швидку взаємодію між кухонним персоналом та централізованою системою управління замовленнями. Використання гібридної моделі комунікації через WebSocket протокол для реального часу та HTTP для структурованого обміну даними створює оптимальний баланс між швидкодією системи та надійністю передачі інформації.

Технологічний стек, що включає NestJS фреймворк для серверної частини, TypeScript для типобезпечної розробки та TypeORM для управління базою даних, забезпечує масштабованість та підтримуваність системи. Інтеграція контейнерних технологій Docker спрощує процеси розгортання та управління інфраструктурою, що особливо важливо для ресторанних мереж з множиною локацій. Система моніторингу на базі Grafana надає менеджменту необхідні інструменти для аналізу операційної ефективності та прийняття обґрунтованих рішень щодо оптимізації робочих процесів.

Практична реалізація проєкту підтвердила доцільність використання ESP32 мікроконтролера для створення спеціалізованих кухонних терміналів, здатних функціонувати в складних умовах ресторанного середовища. Резистивна сенсорна технологія дисплея забезпечує надійну роботу навіть при використанні рукавичок або в умовах підвищеної вологості, що критично важливо для кухонного застосування. Оптимізація енергоспоживання через інтелектуальне управління підсвіткою та режимами роботи контролера дозволяє забезпечити тривалу автономну роботу пристроїв.

Система автентифікації через токен-based механізм забезпечує необхідний рівень безпеки при збереженні простоти налаштування та управління пристроями. Гнучка конфігурація дозволяє адаптувати систему до різних сценаріїв використання, від тестових інсталяцій до повноцінних продуктивних середовищ. Механізми автоматичного відновлення з'єднання та обробки помилок забезпечують стабільність роботи системи навіть в умовах нестабільної мережевої інфраструктури.

Результати тестування підтвердили ефективність розробленого рішення в умовах реального ресторанного середовища. Система демонструє значне зменшення часу передачі інформації про нові замовлення від веб-інтерфейсу до кухонних терміналів, мінімізацію помилок у обробці замовлень та підвищення загальної координації між різними кухонними постами. Інтуїтивний користувацький інтерфейс сенсорних терміналів дозволяє персоналу швидко освоїти систему без тривалого навчання.

Економічна ефективність проєкту підтверджується відносно низькою вартістю апаратних компонентів у порівнянні з комерційними рішеннями аналогічної функціональності. Використання відкритих технологій та стандартних протоколів забезпечує незалежність від конкретних постачальників та можливість подальшого розвитку системи силами внутрішніх розробників.

Перспективи подальшого розвитку проєкту включають інтеграцію систем штучного інтелекту для прогнозування навантаження та оптимізації меню,

розширення аналітичних можливостей через машинне навчання на історичних даних замовлень, а також адаптацію системи для роботи з мобільними додатками та сервісами доставки їжі. Модульна архітектура рішення створює можливості для поетапного впровадження нових функцій без кардинальної модернізації існуючої інфраструктури.

Розроблений програмно-апаратний комплекс являє собою комплексне рішення, що успішно вирішує ключові проблеми автоматизації процесів управління замовленнями в ресторанах східної кухні та може слугувати основою для створення більш масштабних систем цифрової трансформації ресторанного бізнесу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Мартін, Р. Чистий код. Створення і рефакторинг за допомогою Agile / Р. Мартін. — К. : Фабула, 2019. — 448 с.
2. Мартін, Р. Чистий кодер. Правила поведінки для професійних програмістів / Р. Мартін. — К. : Фабула, 2023. — 272 с.
3. Martin, R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Boston : Prentice Hall, 2017. 432 p.
4. Espressif Systems. ESP32 Series Datasheet. URL: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf (дата звернення: 02.06.2025).
5. Espressif Systems. ESP-IDF Programming Guide. URL: https://docs.espressif.com/projects/esp-idf/en/latest/ (дата звернення: 02.06.2025).
6. Espressif Systems. Arduino Core for ESP32. URL: https://github.com/espressif/arduino-esp32 (дата звернення: 02.06.2025).
7. Waveshare Electronics. Waveshare 4inch TFT Touch Shield for Arduino User Manual. URL: https://www.waveshare.com/wiki/4inch_TFT_Touch_Shield (дата звернення: 02.06.2025).
8. NestJS. NestJS Documentation. URL: https://docs.nestjs.com/ (дата звернення: 02.06.2025).
9. Microsoft Corporation. TypeScript Handbook. URL: https://www.typescriptlang.org/docs/ (дата звернення: 02.06.2025).
10. TypeORM. TypeORM Documentation. URL: https://typeorm.io/ (дата звернення: 02.06.2025).
11. Docker Inc. Docker Documentation. URL: https://docs.docker.com/ (дата звернення: 02.06.2025).
12. Grafana Labs. Grafana Documentation. URL: https://grafana.com/docs/ (дата звернення: 02.06.2025).
13. Node.js Foundation. Node.js Documentation. URL: https://nodejs.org/en/docs/ (дата звернення: 02.06.2025).

14. Freeman E. Head First Design Patterns. Sierra. 2nd ed. Sebastopol : O'Reilly Media, 2020. 672 p.

15. Фрімен, Е. Head First. Патерни проєктування / Е. Фрімен, Е. Робсон, Б. Бейтс, К. Сьєрра. — К. : Фабула, 2020. — 672 с.

16. Fowler M. Refactoring: Improving the Design of Existing Code. 2nd ed. Boston : Addison-Wesley, 2018. 448 p.

17. McConnell S. Code Complete: A Practical Handbook of Software Construction. 2nd ed. - Redmond : Microsoft Press, 2004. 960 p.

18. Томас, Д. Програміст-прагматик: друге ювілейне видання / Д. Томас, Е. Хант. — Dallas : Pragmatic Bookshelf, 2024. — 352 с

19. Millett, S. Patterns, Principles, and Practices of Domain-Driven Design / S. Millett, N. Tune. — Indianapolis : Wrox Press, 2015. — 792 p.

20. Freeman, S. Growing Object-Oriented Software, Guided by Tests / S. Freeman, N. Pryce. — Boston : Addison-Wesley, 2009. — 384 p.

21. Haverbeke, M. Eloquent JavaScript: A Modern Introduction to Programming / M. Haverbeke. — 3rd ed. — San Francisco : No Starch Press, 2018. — 472 p.

22. Flanagan D. JavaScript: The Definitive Guide. - 7th ed. Sebastopol : O'Reilly Media, 2020. 706 p.

23. Bibeault B. jQuery in Action. - 3rd ed. Greenwich : Manning Publications, 2017. 504 p.

24. Haverbeke M. Eloquent JavaScript: A Modern Introduction to Programming. - 3rd ed. San Francisco : No Starch Press, 2018. - 472 p.

25. Simpson K. You Don't Know JS: Up & Going. - Sebastopol : O'Reilly Media, 2015. 88 p.

26. Cherny B. Programming TypeScript: Making Your JavaScript Applications Scale. - Sebastopol : O'Reilly Media, 2019. 324 p.

27. Bancila M. Modern C++ Programming Cookbook. Birmingham : Packt Publishing, 2017. - 590 p.

28. Kolban N. Kolban's Book on ESP32. URL: https://leanpub.com/kolban-ESP32 (дата звернення: 02.06.2025).

29. Matthias K. Kane S. Docker: Up & Running. - 2nd ed. Sebastopol : O'Reilly Media, 2018. 384 p.

30. Burns B., Beda J., Hightower K.Kubernetes: Up and Running. - 2nd ed. Sebastopol : O'Reilly Media, 2019. - 302 p.

31. Newman, S. Building Microservices: Designing Fine-Grained Systems. - 2nd ed. Sebastopol : O'Reilly Media, 2021. 616 p.

32. Richardson, C. Microservices Patterns: With Examples in Java. Greenwich : Manning Publications, 2018. 520 p.

33. Walls C. Spring Boot in Action. Greenwich Manning Publications, 2015. 296 p.

34. Flanagan D. Learning React: Modern Patterns for Developing React Apps. - 2nd ed. Sebastopol : O'Reilly Media, 2020. - 306 p.

35. Banks A. , Porcello E. Learning React: Functional Web Development with React and Redux. - 2nd ed. Sebastopol : O'Reilly Media, 2020. 310 p.

36. Chinnathambi, K. React Quickly. - Greenwich : Manning Publications, 2017. - 528 p.

37. Gackenheimer C. Introduction to React. Berkeley : Apress, 2015. 156 p.

38. Boduch A., Derks R. React and React Native. - 3rd ed. Birmingham : Packt Publishing, 2020. - 526 p.

39. Wieruch R. The Road to learn React, 2017. 190 p.

40. Porcello E., Banks A. Learning GraphQL: Declarative Data Fetching for Modern Web Apps. Sebastopol : O'Reilly Media, 2018. 356 p.

41. Buna S. Learning Full-Stack JavaScript Development. Birmingham : Packt Publishing, 2018. 332 p.

42. Young A. Pro MERN Stack: Full Stack Web App Development with Mongo, Express, React, and Node. - 2nd ed. - Berkeley : Apress, 2019. 554 p.

43. Subramanian V. Programming WebRTC: Build Real-Time Streaming Applications for the Web. Dallas : Pragmatic Bookshelf, 2023. 300 p.

44. Tilkov S., Eigenbrodt M., Schreier S., Wolf G. REST und HTTP: Entwicklung und Integration nach dem Architekturstil des Web. Heidelberg : dpunkt.verlag, 2015. 330 p.

45. Masse, M. REST API Design Rulebook / M. Masse. — Sebastopol : O'Reilly Media, 2011. — 116 p.

46. Newman, S. Building Microservices: Designing Fine-Grained Systems / S. Newman. — 2nd ed. — Sebastopol : O'Reilly Media, 2021. — 616 p.

47. Khononov, V. Learning Domain-Driven Design: Aligning Software Architecture and Business Strategy / V. Khononov. — Sebastopol : O'Reilly Media, 2021. — 341 p.

48. Percival, H. Architecture Patterns with Python: Enabling Test-Driven Development, Domain-Driven Design, and Event-Driven Microservices / H. Percival, B. Gregory. — Sebastopol : O'Reilly Media, 2020. — 304 p.

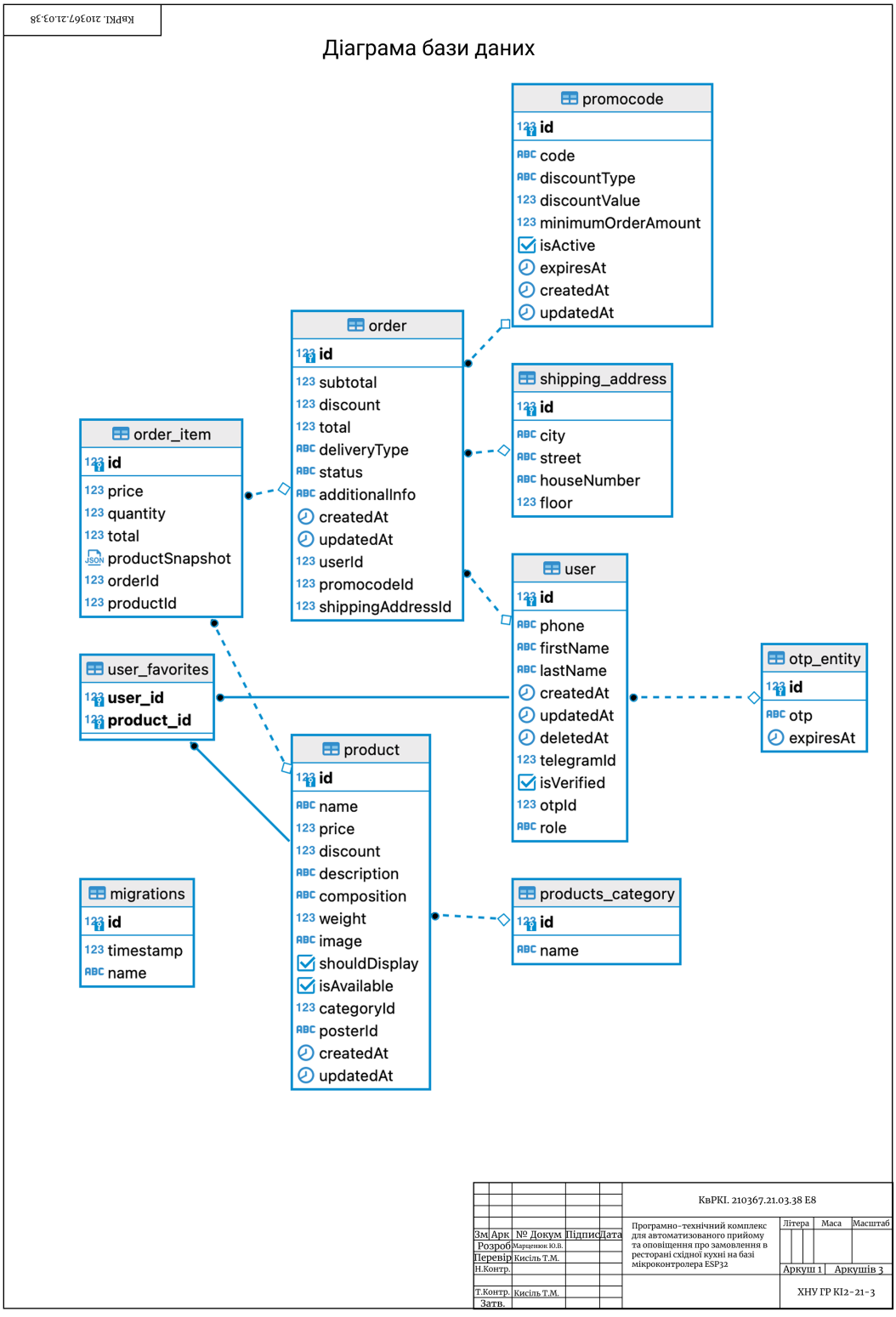
49. Khorikov V. Unit Testing Principles, Practices, and Patterns / V. Khorikov. - Greenwich : Manning Publications, 2020. 296 p.

50. Seemann, M. Code That Fits in Your Head: Heuristics for Software Engineering / M. Seemann. — Boston : Addison-Wesley, 2021. — 464 p.

Додаток А

(обов'язковий)

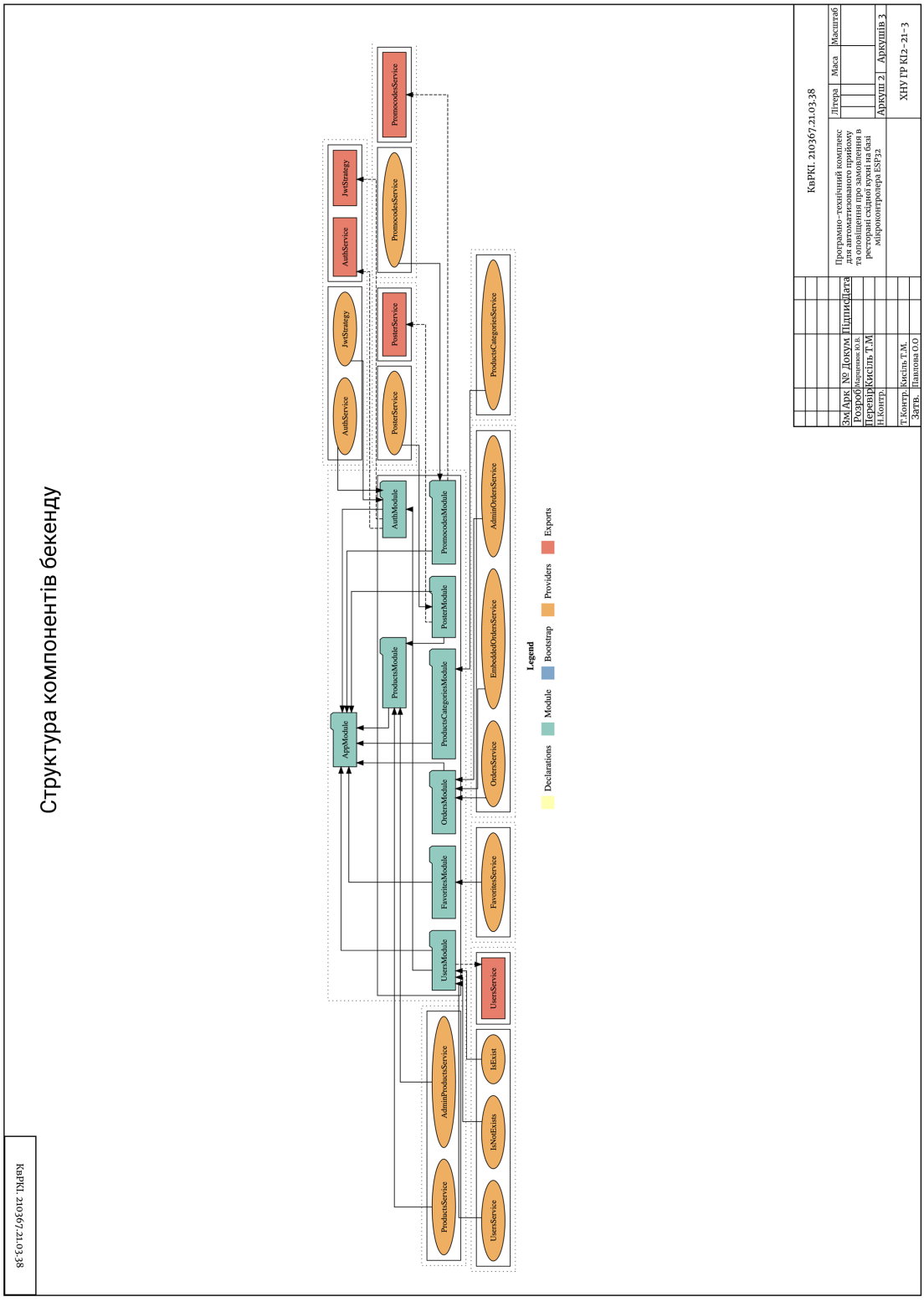
КОПІЯ КРЕСЛЕННЯ «ДІАГРАМА БАЗИ ДАНИХ»



Додаток Б

(обов’язковий)

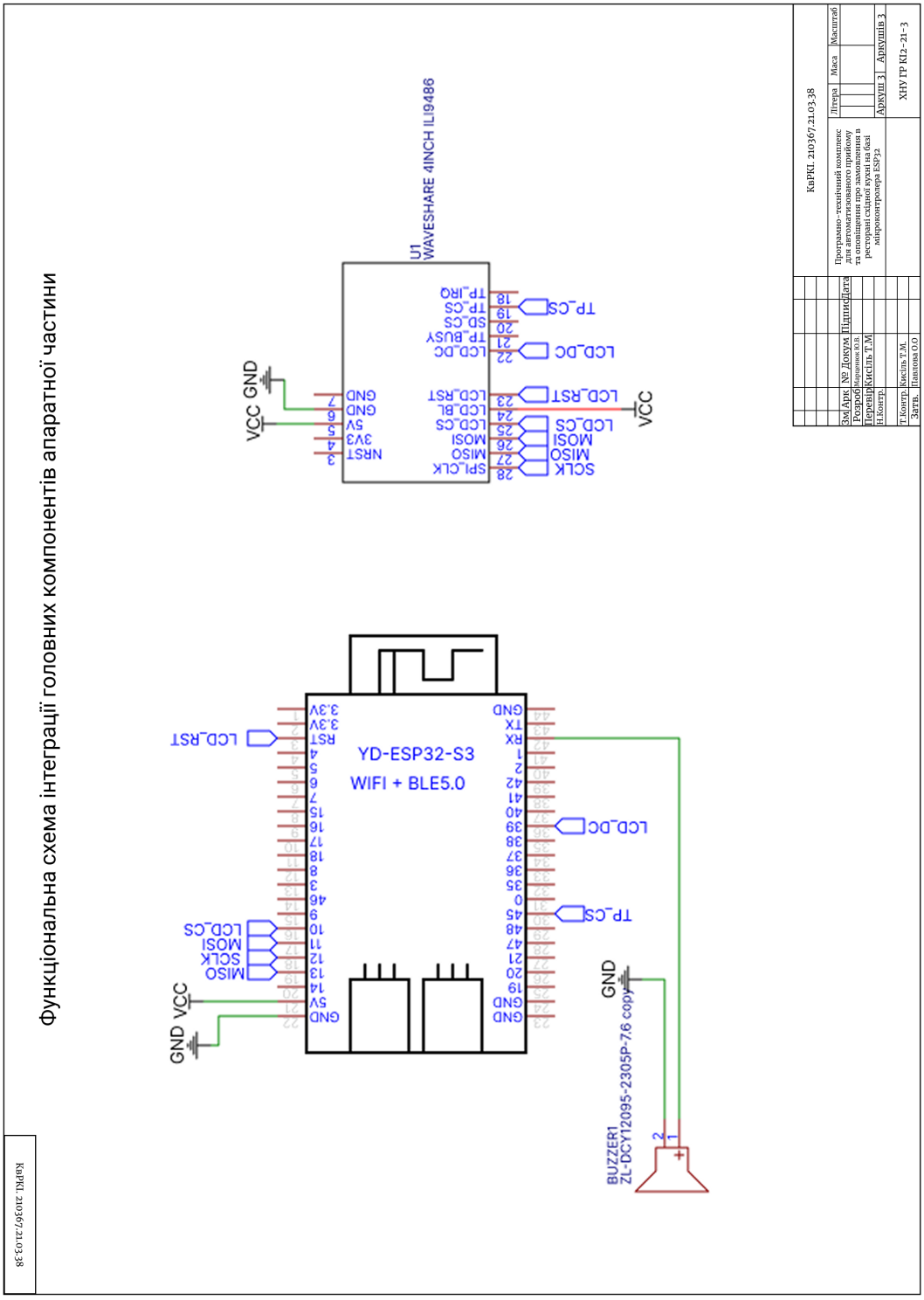
КОПІЯ КРЕСЛЕННЯ «СТРУКТУРА КОМПОНЕНТІВ БЕКЕНДУ»



Додаток В

(обов’язковий)

КОПІЯ КРЕСЛЕННЯ «ФУНКЦІОНАЛЬНА СХЕМА ІНТЕГРАЦІЇ ГОЛОВНИХ КОМПОНЕНТІВ АПАРАТНОЇ ЧАСТИНИ»



Anti-Plagiarism (UA) v-15.281 Educational

The maximum coincidence with one document 18.0%

Dictionaries check: en_US, ru_RU, ua_UA. **Errors in the documents: 12%**

ID: 244913 Title: БКР Програмно-технічний комплекс для автоматизованого прийому та оповіщення про замовлення в ресторані східної кухні на базі мікроконтролера ESP32 Added in a DB: 2025-06-11 Authors: Юрій МАРЦЕНЮК Heads: Тетяна КИСІЛЬ Consultants: Opponents:	Document		Sum coincidence on the DB	
	Symbols	Lexemes	Symbols	Lexemes
	91010	698	19062 (21%)	189 (27%)

Plagiarism sources

ID	Description	Plagiarism presence in the document	
		Symbols	Lexemes
240739	Title: Звіт з ПДП Програмно-технічний комплекс для автоматизованого прийому та оповіщення про замовлення в ресторані східної кухні на базі мікроконтролера ESP32» Added in a DB: 2025-05-01 Authors: Марценюк Ю.В. Heads: Іванов О.В. Consultants: Opponents:	16272 (18.0%)	150 (21.0%)

Протокол аналізу звіту подібності експертом

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Юрій МАРЦЕНЮК

Співавтор:

Назва: Марценюк_Програмно-технічний комплекс для автоматизованого прийому та оповіщення про замовлення в ресторані східної кухні на базі мікроконтролера ESP32

Експерт:

Підрозділ: Кафедра комп'ютерної інженерії та інформаційних систем

Коефіцієнт подібності 1:5.1%

Коефіцієнт подібності 2:1.8%

Мікропробіли: 7

Заміна букв: 0

Інтервали: 0

Білі знаки: 0

Дата створення звіту: 2025-06-11 10:39:02.0

Після аналізу Звіту подібності констатую наступне:

☒ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

2025-06-11 Доцент Андрій Нічепорук

Дата експерт

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Марценюк Юрій Володимирович

Тема: Програмно-технічний комплекс для автоматизованого прийому та оповіщення про замовлення в ресторані східної кухні на базі мікроконтролера ESP32

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг кваліфікаційної роботи:

Кількість листів креслень 3 Кількість сторінок записки 74

1. Короткий зміст роботи та прийнятих рішень: Метою кваліфікаційної роботи є діджиталізація ресторанного бізнесу з використанням фізично-електричного пристрою на базі мікроконтролера ESP32.

2. Висновок про відповідність роботи дипломному завданню: Робота повністю відповідає поставленому завданню.

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: В першому розділі кваліфікаційної роботи проведено дослідження предметної області (проведений аналіз конкурентів, зокрема виявлення переваг та недоліків конкурентів присутніх на ринку) та виконано постановку задачі дослідження. В другому розділі кваліфікаційної роботи проведено вибір інструментів та технологій для виконання роботи: виконано формалізований опис програмно-апаратного комплексу; розроблений список технологій що задовільняють вимоги поставленої мети. В третьому розділі кваліфікаційної роботи виконано апаратну та програмну реалізацію програмно-апаратного комплексу, а саме: реалізовано всі необхідні компоненти для функціонування програмно-апаратного комплексу таких як: фронтенд частина, бекенд частина, девопс частина, апаратна частина.

4. Позитивні сторони роботи: висока практична цінність роботи.

5. Негативні сторони роботи: недостатня увага до деталей зручності користування продуктом таких як налаштування WiFi мережі.

6. Оцінка графічного оформлення та пояснювальної записки роботи: Пояснювальна записка оформлена коректно, згідно діючих стандартів оформлення документації.

7. Відгук про роботу в цілому: Робота виконана на належному науково-технічному рівні.

8. Інші зауваження: _____

9. Оцінка дипломної роботи: відмінно

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) _____

Бездольний Леонід Петрович, зав. кафедрою ЗВВ ХНУ

_____

"12" червня 2025 р. _____ (підпис)

Завідувачу кафедри КІІС
д-р. філософії, доц. Ользі ПАВЛОВІЙ

Юрія МАРЦЕНЮКА
ПІБ здобувача вищої освіти

ФІТ, 4 курсу, групи КІ2-21-3

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності у Хмельницькому національному університеті» від 01.07.2022, згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений(а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Strike-Plagiarism та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

12.06 2025 року

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ
КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Програмно-технічний комплекс для автоматизованого прийому та оповіщення про замовлення в ресторані східної кухні на базі мікроконтролера ESP32

Автор: Юрій Марценюк

Спеціальність: 123– Комп'ютерна інженерія

Освітня програма: освітньо-професійна

Науковий керівник: Тетяна КИСІЛЬ, доцент кафедри КІІС

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) запозичення розміщені в розділах аналізу існуючих аналогів та прототипів, які не описують безпосередньо авторське дослідження і не стосуються результатів роботи;
2) усі запозичення фрагментарні, або мають належним чином оформленні посилання;
3) окремі виявлені збіги є загальновживаними фразами або виразами, про що свідчить посилання системи на збіг з 10-40 джерелами на один фрагмент речення;
4) в якості запозичень в окремих місцях системою зафіксовано послідовності чотирьохрозряних двійкових кодів, які є вхідними даними до великої кількості задач і не можуть розглядатися як об'єкт авторських прав і, відповідно, їх порушення;
5) всі зафіксовані системою ознаки модифікації тексту відносяться до комбінування латинських символів зі україномовними скороченнями індексів в формулах, що не є модифікацією тексту.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості StrikePlagiarism, складає 5.1% і адресується до 401 першоджерела; та системою Anti-Plagiarism складає 18%, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ Тетяна КИСІЛЬ

Гарант ОП _____ Андрій Нічепорук

Завідувач кафедри КІІС _____ Ольга ПАВЛОВА