

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій
(повне найменування факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавр

(ступінь вищої освіти)

на тему РОЗРОБЛЕННЯ ТА ДОСЛІДЖЕННЯ ПРОГРАМНОГО
ЗАБЕЗПЕЧЕННЯ АВТОМАТИЗАЦІЇ ЗАКЛАДІВ ХАРЧУВАННЯ
DEVELOPMENT AND RESEARCH OF SOFTWARE FOR
AUTOMATION OF CATERING ESTABLISHMENTS

Виконав(ла): студент(ка) 4 курсу, групи КНТ-132
Спеціальності 121 Інженерія програмного
(код і найменування спеціальності)
забезпечення

Освітня програма (спеціалізація)
Інженерія програмного забезпечення

МАРЯХІН Р.О.

(ПРІЗВИЩЕ та ініціали)

Керівник ПОДКОВАЛІХІНА О.О.

(ПРІЗВИЩЕ та ініціали)

Рецензент ЗЕЛІК О.В.

(ПРІЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Факультет КНТ
 Кафедра програмних засобів
 Ступінь вищої освіти бакалавр
 Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
 Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
 “ ” 2026 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

МАРЯХІНА Романа Олексійовича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розроблення та дослідження програмного забезпечення автоматизації закладів харчування. Development and Research of Software for Automation of Catering Establishments

керівник проєкту (роботи) к.ф.-м.н., доцент,

ПОДКОВАЛІХІНА Олена Олександрівна

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

затверджені наказом закладу вищої освіти від “ 07 ” квітня 2026 року № 139

2. Строк подання студентом проєкту (роботи) 10 червня 2026 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Проєктування та математичне забезпечення системи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання ви- дав	прийняв виконане за- вдання
1-4 Основна частина	ПОДКОВАЛІХІНА О.О., доцент		
Нормоконтроль	БСЛОВА А.В., асистент		

7. Дата видачі завдання « 20 » квітня 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій	2 тиждень	Розділ 2
	розробки.		
4	Розробка архітектури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділ 3, 4
6	Тестування та експериментальне дослідження	5 тиждень	Розділ 4
	програмного забезпечення.		
7	Оформлення пояснювальної записки та документів	6 тиждень	Додатки
	до неї.		
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

(підпис) Роман МАРЯХІН
(Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Олена ПОДКОВАЛІХІНА
(Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 116 с., 7 табл., 17 рис., 3 дод., 39 джерел.

QR-МЕНЮ, АВТОМАТИЗАЦІЯ РЕСТОРАНІВ, МАТРИЦЯ КАСА-ВАНИ-СМІТА, МЕНЮ ІНЖИНІРИНГ, АВС-АНАЛІЗ, DJANGO, PANDAS, РЕНТАБЕЛЬНІСТЬ.

Об'єкт дослідження – процес розроблення програмного забезпечення для автоматизації сервісних операцій та програмної реалізації методів обробки даних у ресторанному бізнесі.

Предмет дослідження – методи програмної реалізації QR-інтерфейсів та алгоритми автоматизованого розрахунку параметрів асортименту.

Мета роботи – підвищення швидкості обробки замовлень через впровадження QR-технологій та автоматизація отримання результатів оцінки рентабельності меню.

Матеріали, методи та технічні засоби: об'єктно-орієнтоване програмування, реляційні бази даних, методи статистичної обробки даних, алгоритмічні методи виявлення асоціативних зв'язків у масивах даних, мови Python та JavaScript, фреймворк Django, бібліотеки Pandas і Chart.js, СКБД PostgreSQL, HTML5/CSS3, персональний комп'ютер з процесором Intel Core i7-10750H під управлінням Microsoft Windows 11.

Результати. Розроблено архітектуру та програмну реалізацію веб-застосунку, що забезпечує клієнтам можливість замовлення через QR-інтерфейси. Для адміністраторів реалізовано функціонал моніторингу продажів та автоматизованого обчислення показників рентабельності меню з програмною генерацією рекомендацій.

Висновки. Розроблено систему автоматизації закладів харчування, яка інтегрує клієнтські сервісні модулі з інструментами алгоритмічної обробки даних для підтримки процесів управління асортиментом.

Галузь використання – цифровізація оперативної діяльності та меню-менеджменту в закладах сегмента HoReCa.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 116 pages, 7 tables, 17 figures, 3 appendixes, 39 sources.

QR-MENU, RESTAURANT AUTOMATION, KASAVANA-SMITH MATRIX, MENU ENGINEERING MODEL, ABC ANALYSIS, DJANGO, PANDAS, PROFITABILITY.

Object of study – the process of developing software for automating service operations and the software implementation of data processing methods in the restaurant business.

Subject of study – methods of software implementation of QR interfaces and algorithms for the automated calculation of assortment parameters.

Purpose of the work – increasing the speed of order processing through the implementation of QR technologies and automating the retrieval of menu profitability evaluation results.

Materials, methods, and technical tools: object-oriented programming, relational databases, statistical data processing methods, algorithmic methods for detecting associative links in data arrays, Python and JavaScript languages, Django framework, Pandas and Chart.js libraries, PostgreSQL DBMS, HTML5/CSS3, a personal computer with an Intel Core i7-10750H processor running Microsoft Windows 11.

Results. The architecture and software implementation of a web application that provides clients with the ability to order via QR interfaces have been developed. For administrators, the functionality of sales monitoring and automated calculation of menu profitability indicators with the programmatic generation of recommendations has been implemented.

Conclusions. A food service automation system has been developed that integrates client service modules with algorithmic data processing tools to support assortment management processes.

Field of application – digitalization of operational activities and menu management in establishments of the HoReCa segment.

ЗМІСТ

	С.
Вступ.....	11
1 Аналіз предметної області.....	13
1.1 Аналіз поточного стану у сфері громадського харчування.....	13
1.2 Огляд та порівняльний аналіз існуючих систем автоматизації	17
1.3 Висновки за розділом 1	21
2 Проєктування та математичне забезпечення системи.....	22
2.1 Постановка задачі та формування вимог до ПЗ.....	22
2.2 Математичне обґрунтування методів аналізу рентабельності	23
2.2.1 ABC-аналіз та принцип Парето	23
2.2.2 Модель Касавани-Сміта	26
2.3 Математичне забезпечення системи рекомендацій МВА	29
2.4 Вибір та обґрунтування технологічного стека	30
2.4.1 Вибір мови програмування та серверного фреймворку.....	30
2.4.2 Вибір системи керування базами даних	33
2.4.3. Інструменти оцінки та роботи з даними	35
2.4.4. Технології клієнтської частини	36
2.5 Висновки за розділом 2	37
3 Опис програми.....	39
3.1 Проєктування архітектури та структури бази даних.....	39
3.2 Реалізація серверної логіки	41
3.2.1 Структура програмного модуля	41
3.2.2 Програмна реалізація ORM.....	44
3.2.3 Обробка запитів та контролери	46
3.2.4 Забезпечення безпеки та валідація даних.....	47
3.3. Програмна реалізація розрахункових модулів.....	47
3.3.1 Програмна реалізація ABC-аналізу.....	48
3.3.2 Програмна реалізація методу Касавани-Сміта	49

3.3.3 Програмна реалізація модуля асоціативних правил МВА	50
3.4 Проектування та реалізація користувацьких інтерфейсів	52
3.4.1 Концепція архітектури клієнтської частини	52
3.4.2 Реалізація інтерфейсу користувача	53
3.4.3 Реалізація інтерфейсу дашборду в панелі адміністратора.....	54
3.5 Висновки за розділом 3	55
4 Експлуатація, тестування та експериментальне дослідження програми	56
4.1 Призначення й умови застосування програми.....	56
4.2 Методика та результати тестування програмного забезпечення	57
4.3 Експериментальне дослідження роботи розрахункових алгоритмів.....	58
4.4 Інструкція по експлуатації програми	60
4.4.1 Звернення до програми.....	60
4.4.2. Вхідні й вихідні дані	60
4.4.3 Повідомлення	61
4.4.4 Виконання програмного забезпечення	61
4.5 Висновки за розділом 4	62
Висновки	63
Перелік джерел посилання	66
ДОДАТОК А Технічне завдання	71
ДОДАТОК Б Фрагмент тексту програми	76
ДОДАТОК В Слайди презентації.....	108

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

ABC-аналіз	– метод класифікації ресурсів за ступенем їхнього впливу на загальний прибуток;
Cross-selling	– стратегія продажів, яка полягає у пропозиції клієнту схожих або додаткових товарів або послуг під час основної покупки з метою збільшення середнього чека та підвищення рентабельності.
HoReCa	– Hotel, Restaurant, Cafe/Catering;
MBA	– Market Basket Analysis;
Mobile-First	– підхід у веб-розробці, при якому інтерфейс спочатку створюється для смартфонів, а лише потім масштабується десктопних пристроїв;
ORM	– Object-Relational Mapping;
QR code	– Quick Response code;
БД	– база даних;
ПЗ	– програмне забезпечення;
СКБД	– система керування базами даних.

ВСТУП

Актуальність теми. У сучасній індустрії гостинності HoReCa активно впроваджуються цифрові технології для оптимізації бізнес-процесів і покращення клієнтського сервісу. Звичайні паперові меню та ручне приймання замовлень поступово втрачають ефективність через високі витрати часу та ризик впливу людського фактору [1]. Саме тому системи на основі QR-кодів, що забезпечують швидкий, зручний і безконтактний доступ до товарів закладу зі смартфонів, фактично перетворилися на стандарт у цьому секторі [2].

Однак більшість існуючих рішень на ринку обмежуються лише функцією електронного меню, ігноруючи широкі можливості використання даних, що накопичуються під час взаємодії користувача з системою. Збір та аналіз цих даних є критично важливими для прийняття обґрунтованих стратегічних рішень. Для того, щоб максимізувати прибуток закладу необхідно не лише автоматизувати процес замовлення, але й впровадити інтелектуальні модулі підтримки прийняття рішень. Зокрема, використання матриці Касавани-Сміта та класичного ABC-аналізу дозволяє детально класифікувати позиції за їх рентабельністю та популярністю, надаючи можливість для оптимізації асортименту страв [3]-[5].

Застосування алгоритмів асоціативних правил МВА під час формування кошика дає змогу ефективно здійснювати перехресні продажі, пропонуючи клієнтам додаткові страви та підвищуючи середній чек [6]-[8]. Тому створення ПЗ, що поєднує зручність QR-меню для клієнтів разом з інструментами оцінки асортименту для адміністрації, є актуальним та економічно обґрунтованим завданням.

Об'єкт дослідження – процес розроблення програмного забезпечення для автоматизації сервісних операцій та програмної реалізації методів обробки даних у ресторанному бізнесі.

Предмет дослідження – методи програмної реалізації QR-інтерфейсів та алгоритми автоматизованого розрахунку параметрів асортименту.

Мета роботи полягає у підвищенні швидкості обробки замовлень через впровадження QR-технологій та автоматизації отримання результатів оцінки рентабельності меню.

Для досягнення поставленої мети було вирішено такі задачі:

- проведено аналіз предметної області та досліджено переваги й недоліки існуючих систем автоматизації ресторанів [1], [2];
- спроектовано архітектуру та реляційну БД веб-застосунку для керування замовленнями;
- розроблено клієнтську частину з підтримкою локального кошика та зручним підбором страв;
- досліджено та реалізовано алгоритм асоціативних правил МВА для створення персоналізованих рекомендацій клієнтам [4];
- створено модуль моніторингу для керівництва на основі матриці Касавани-Сміта та ABC-аналізу [3], [5], [9];
- проведено тестування розробленого ПЗ, що продемонструвало його економічну ефективність.

Методи дослідження. У роботі застосовано методи об'єктно-орієнтованого програмування та проектування для створення архітектури системи; реляційну алгебру для побудови бази даних; статистичні методи для оцінки ефективності меню; алгоритми асоціативних правил для реалізації системи рекомендацій.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз поточного стану у сфері громадського харчування

Останнє десятиліття повністю змінило світову індустрію гостинності HoReCa. Цифрова трансформація, яку ще кілька років тому багато власників вважали просто приємною додатковою фішкою для великих ресторанних мереж, сьогодні стала обов'язковою умовою для виживання закладів будь-якого розміру [10]-[12].

Індустрія гостинності є стратегічно важливим сектором економіки, що включає підприємства, ключовою функцією яких є надання послуг з тимчасового розміщування та організації харчування. При цьому забезпечення прибутковості визначається як необхідна умова для фінансової стійкості суб'єктів цієї галузі [13].

Цей процес значно прискорився через глобальні проблеми. Спочатку пандемія COVID-19 змусила власників бізнесів шукати безконтактні способи роботи з відвідувачами. Для України ситуація стала ще складнішою через повномасштабну війну. В таких важких умовах ресторани стикаються з нестабільною економікою, нестачею кваліфікованих працівників, проблемами з логістикою та постійним збільшенням цін на продукти. Тому впровадження нових маркетингових і організаційних рішень, особливо систем автоматизації, стало критично важливим для відновлення і подальшого розвитку галузі [13]-[15].

Звичайна стара модель обслуговування, коли офіціант видає паперове меню і вручну записує замовлення, поступово ставала все менш ефективною. Такий підхід потребує грошей і багато часу. Друк та постійне оновлення паперових меню обходиться дорого, особливо коли ціни на продукти змінюються майже щотижня. Крім того, через людський фактор часто трапляються помилки в замовленнях. Це знижує враження гостей і зменшує прибутки закладу.

Сучасні системи автоматизації в ресторанному бізнесі рухаються в бік гнучких і зручних безконтактних рішень. Одним із найкращих і найвигідніших інструментів зараз вважаються системи на основі QR-кодів. При використанні

цієї технології не потрібно завантажувати сторонні додатки для того щоб замовити страви (рис. 1.1) [1], [2].



Рисунок 1.1 – Схема взаємодії клієнта та системи QR-замовлень [1]

Спершу відвідувач має відсканувати QR-код, розміщений на столі, після чого він отримує доступ до меню ресторану. Далі клієнт формує та надсилає замовлення, а дані про нього автоматично зберігаються в БД. Однією з головних переваг переходу на цифрове меню є скорочення часу, який витрачається на кожну операцію замовлення. Дослідження показують, що автоматизація через QR-коди дозволяє оптимізувати роботу закладу на кожному етапі – від моменту посадки гостя до закриття рахунку.

Для підтвердження ефективності такого підходу в таблиці 1.1 наведено порівняння часових витрат при звичайному способі обслуговування та при використанні системи DineWise RMS на основі QR-кодів [2].

Таблиця 1.1 – Порівняння часових витрат на обслуговування [2]

Етап	Звичайний спосіб обслуговування	Система з QR-кодом	Економія часу
Бронювання столика	3 хв	30 сек	2.5 хв
Час у черзі	15 хв (фізична черга)	10 хв (віртуальна черга)	5 хв
Оформлення замовлення	5 хв	1 хв	4 хв
Обробка платежу	4 хв	1 хв	3 хв
Загальний час взаємодії	27 хв	12.5 хв	15.5 хв

З таблиці 1.1 видно, що цифрові рішення значно скорочують час виконання ключових процесів у закладах громадського харчування. Це свідчить про те, що перехід від ручних до цифрових систем дозволяє економити час на кожному етапі обслуговування.

Впровадження інтерактивних QR-меню допомагає закладам вирішити кілька серйозних проблем:

- зниження навантаження на персонал: офіціантам більше не треба постійно носити меню і стояти біля столу, поки гість вирішує що саме він бажає. Особливо це рятує в години пікового навантаження. Клієнт сам обирає і оформлює замовлення, а персонал може зосередитися на подачі страв і підвищенні якості комунікації з відвідувачами;
- економія коштів: заклад повністю відмовляється від витрат на друк паперових меню;
- гнучкий маркетинг: ціни, сезонні пропозиції чи акції можна змінювати в будь-який момент. Прямо в реальному часі можна додавати яскраві позначки до страв: «Новинка», «Вибір шефа», «Гостре» тощо. Це відразу привертає увагу відвідувачів.

Просто замінити паперове меню на звичайний документ або PDF-файл, що лежить на хмарному сховищі, вже не є актуальним. Бо такі прості рішення мають великий недолік – вони не збирають жодних даних про гостей [1], [16].

Сьогодні дані про замовлення клієнтів – це один з найцінніших ресурсів закладу. Якщо власник не бачить, які страви люди переглядають найчастіше, а які взагалі ігнорують, він керує рестораном фактично наосліп. Тому зараз потрібні розумні системи, які не тільки показують меню, а й працюють як потужний інструмент для меню-менеджменту [3], [17].

Дослідники в ресторанному менеджменті давно говорять про необхідність регулярно проводити інжиніринг меню [18]. Одна з найзручніших моделей для цього – класична модель Касавани-Сміта. Вона дозволяє оцінювати кожну страву не тільки за кількістю продажів, а й за її реальною прибутковістю. Досвід як українських, так і європейських ресторанів показує, що такий аналіз допомагає очистити меню від не вигідних або збиткових позицій і зробити акцент на стравах, які дійсно приносять прибуток [3], [16], [19].

Також, сучасні системи автоматизації дозволяють використовувати штучний інтелект і аналіз даних прямо під час продажу. Наприклад, метод МВА допомагає знаходити приховані зв'язки між стравами. Якщо клієнти в 80% випадків замовляють певний соус або напій до конкретної піци, система може автоматично пропонувати цей товар. Такі рекомендації в форматі cross-selling на практиці підвищують середній чек на 10–20% [3], [6]–[8].

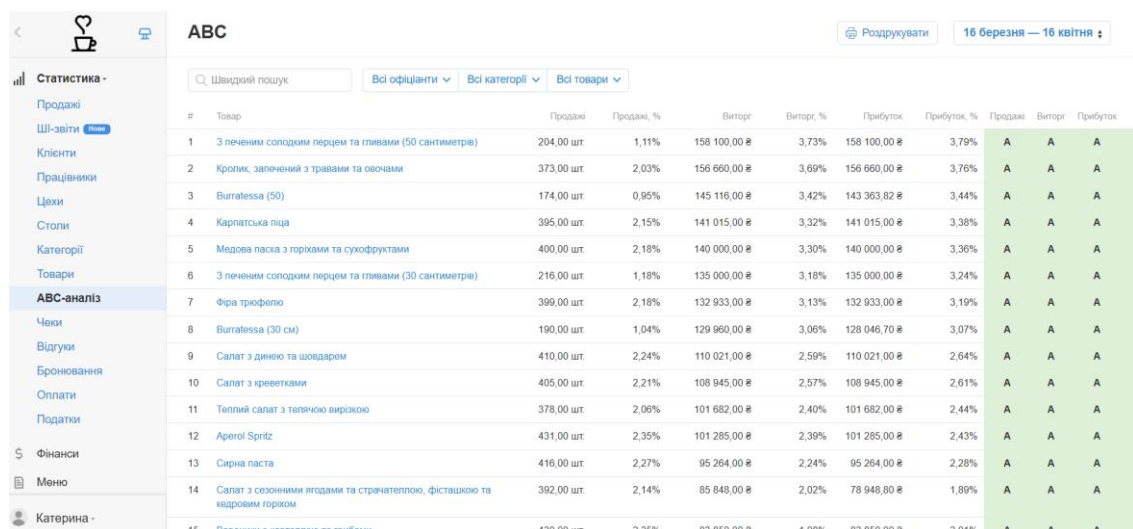
Тому можна сказати, що найперспективнішим напрямком наразі є створення комплексної системи автоматизації. Вона повинна зручно поєднувати простий і сучасний мобільний інтерфейс для гостей з функціями обробки даних та засобами стратегічного планування. Така система дозволить власнику не просто приймати замовлення, а й керувати прибутковістю меню на основі реальних даних продажів.

1.2 Огляд та порівняльний аналіз існуючих систем автоматизації

На сьогодні сфера систем для закладів громадського харчування пропонує багато різних рішень – від комплексних систем до вузьких сервісів. Дослідивши їх, можна зазначити, що більшість платформ або зосереджені лише на фінансово-складському обліку, або тільки на візуальному поданні меню через QR-код.

Щоб обґрунтувати потребу у створенні нової системи, слід розглянути функціональні можливості серед найбільш популярних рішень, що використовуються в Україні: Poster, ChoiceQR та Me-Menu [20]-[22].

Poster є однією з найбільш відомих і поширених хмарних систем автоматизації кафе, ресторанів та магазинів в Україні та за її межами. Головний акцент у цьому продукті зроблено на стабільній роботі каси, управлінні залишками на складі та веденні фінансового обліку (рис. 1.2) [20].



#	Товар	Продажі	Продажі, %	Виторг	Виторг, %	Прибуток	Прибуток, %	Продажі	Виторг	Прибуток
1	З печеним солодким перцем та гливками (50 сантиметрів)	204,00 шт.	1,11%	158 100,00 ₴	3,73%	158 100,00 ₴	3,79%	A	A	A
2	Кролик, заправлений з травами та овочами	373,00 шт.	2,03%	156 660,00 ₴	3,69%	156 660,00 ₴	3,76%	A	A	A
3	Butafressa (50)	174,00 шт.	0,95%	145 116,00 ₴	3,42%	143 363,82 ₴	3,44%	A	A	A
4	Карпатська піца	395,00 шт.	2,15%	141 015,00 ₴	3,32%	141 015,00 ₴	3,38%	A	A	A
5	Медова паска з горіхами та сухофруктами	400,00 шт.	2,18%	140 000,00 ₴	3,30%	140 000,00 ₴	3,36%	A	A	A
6	З печеним солодким перцем та гливками (30 сантиметрів)	216,00 шт.	1,18%	135 000,00 ₴	3,18%	135 000,00 ₴	3,24%	A	A	A
7	Фіра трифено	399,00 шт.	2,18%	132 933,00 ₴	3,13%	132 933,00 ₴	3,19%	A	A	A
8	Butafressa (30 см)	190,00 шт.	1,04%	129 960,00 ₴	3,06%	128 046,70 ₴	3,07%	A	A	A
9	Салат з динкою та шовдаром	410,00 шт.	2,24%	110 021,00 ₴	2,59%	110 021,00 ₴	2,64%	A	A	A
10	Салат з креветками	405,00 шт.	2,21%	108 945,00 ₴	2,57%	108 945,00 ₴	2,61%	A	A	A
11	Теплий салат з телячою вирізкою	378,00 шт.	2,06%	101 682,00 ₴	2,40%	101 682,00 ₴	2,44%	A	A	A
12	Aperol Spritz	431,00 шт.	2,35%	101 285,00 ₴	2,39%	101 285,00 ₴	2,43%	A	A	A
13	Сирна паста	416,00 шт.	2,27%	95 264,00 ₴	2,24%	95 264,00 ₴	2,28%	A	A	A
14	Салат з сезонними ягодами та страчатеплою, фісташкою та кадровим горохом	392,00 шт.	2,14%	85 848,00 ₴	2,02%	78 948,80 ₴	1,89%	A	A	A
15	Вареники з картоплею та грибами	430,00 шт.	2,35%	83 850,00 ₴	1,98%	83 850,00 ₴	2,01%	A	A	A

Рисунок 1.2 – Інтерфейс системи Poster [20]

QR-меню в екосистемі Poster присутнє, але воно реалізоване швидше як додаткова, аніж як ключова функція. Інтерфейс замовлення є базовим і не пропонує складних інтелектуальних рекомендацій гостю. Poster надає зручні

звіти про продажі, рух коштів і собівартість страв. Проте для глибокого аналізу, наприклад побудови матриці Касавани-Сміта, дані доводиться експортувати в Excel і обробляти вручну. Система показує статистику без автоматичних рекомендацій щодо страв [20].

На відміну від Poster, ChoiceQR зосереджений на клієнтському досвіді та маркетингу. Платформа пропонує онлайн-меню, безконтактну оплату за столом (Apple Pay, Google Pay) та систему виклику офіціанта (рис. 1.3) [21].

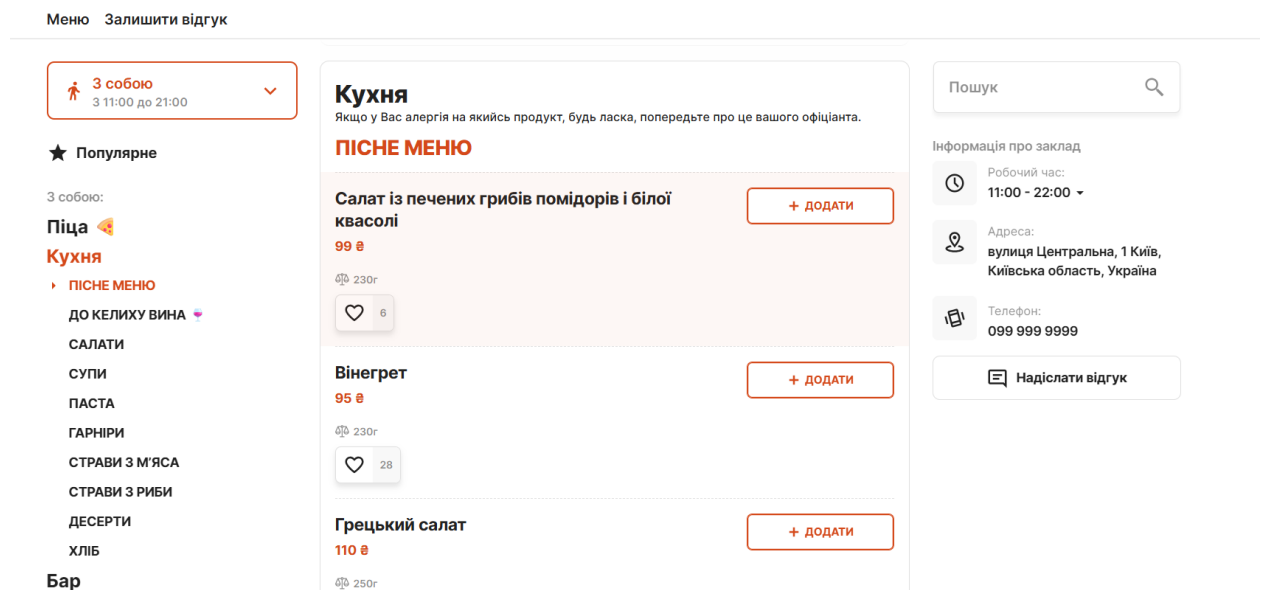


Рисунок 1.3 – Інтерфейс клієнтського меню системи ChoiceQR [21]

ChoiceQR добре працює для взаємодії з відвідувачами, але його аналітика обмежена. Власник бачить лише базові звіти – кількість замовлень, години відвідування та виручку через QR-оплати. Система не має модуля для оцінки прибутковості страв у співвідношенні з їхньою популярністю (меню-інжиніринг). Тож платформа допомагає продавати, але не показує, наскільки вигідні окремі позиції [21].

Me-Menu це сервіс для швидкого створення цифрового меню. Він підходить для невеликих кав'ярень, пекарень чи мобільних точок харчування, що мають незмінний асортимент (рис. 1.4) [22].

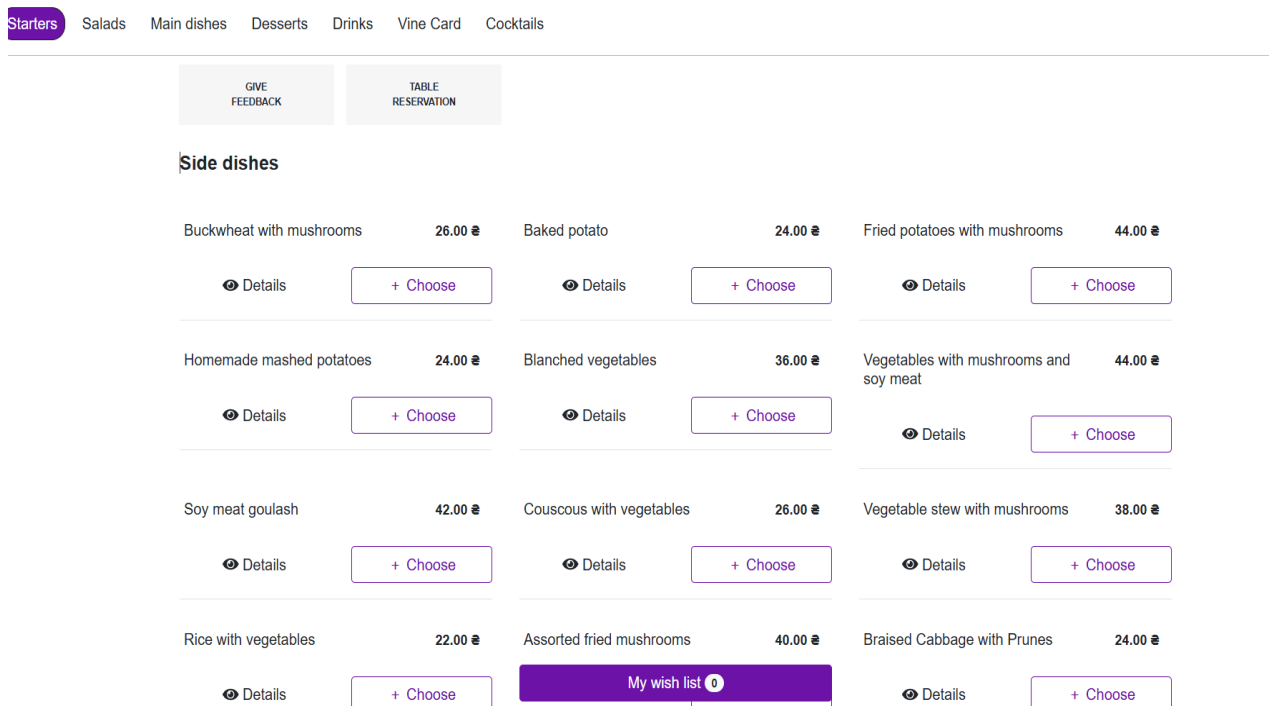


Рисунок 1.4 – Приклад згенерованого меню в системі Me-Menu [22]

Основна перевага сервісу Me-Menu це швидкий запуск. Для великих ресторанів або закладів із частими змінами меню можливостей системи бракує. Відсутність інтеграції зі складським обліком не дозволяє контролювати собівартість і маржинальність у реальному часі, тому глибокий аналіз неможливий [22].

Для узагальнення порівняємо функціональні можливості розглянутих рішень. Результати порівняння наведено у таблиці 1.2.

Таблиця 1.2 – Порівняльна характеристика систем автоматизації

Функціональна можливість	Poster [20]	ChoiceQR [21]	Me-Menu [22]	Scan & Savor
Електронне QR-меню та замовлення	Так	Так	Так	Так
Матриця Касавани-Сміта	Ні	Ні	Ні	Так

Кінець таблиці 1.2

Функціональна можливість	Poster [20]	ChoiceQR [21]	Me-Menu [22]	Scan & Savor
Складський та фінансовий облік	Так	Ні	Ні	Ні
АВС-аналіз	Так	Ні	Ні	Так
Система розумних рекомендацій	Ні	Так	Так	Так
Розрахунок динаміки про- дажів у реальному часі	Так	Так	Ні	Так

Аналіз таблиці 1.2 показує, що популярні комерційні системи орієнтовані переважно на ведення обліку та презентацію меню для клієнтів. Вони надають керівництву лише базову статистику продажів, не звертаючи уваги на детальний аналіз рентабельності кожної позиції.

На відміну від інших комерційних систем, головною перевагою системи «Scan & Savor», що розробляється в дипломній роботі, є наявність повноцінного модуля оцінки ефективності меню. Завдяки інтеграції матриці Касавани-Сміта та алгоритмів МВА вона не просто зберігає дані про продажі, а й автоматично формує рекомендації для власника на основі цих даних. Власнику можуть бути надані рекомендації щодо страв, які варто активно просувати, які слід змінити для зниження собівартості, а які взагалі прибрати з меню через низьку маржинальність.

Таким чином, розробка «Scan & Savor» є актуальною тому, що поєднує зручний інструмент для безконтактних замовлень разом з центром керування прибутком, створюючи нові можливості на ринку програмного забезпечення для HoReCa.

1.3 Висновки за розділом 1

Проведено детальний аналіз поточного стану предметної області та визначено основні тенденції розвитку систем автоматизації в індустрії гостинності HoReCa. З'ясовано, що в умовах економічних проблем та необхідності зменшення витрат перехід до цифрових безконтактних рішень, зокрема QR-меню, є критично важливим для ефективної роботи закладів громадського харчування. Використання таких технологій дозволяє зменшити навантаження на робочий персонал, відмовитися від витрат на друк паперових меню та забезпечити гнучке використання інструментів дослідження продажів у режимі реального часу.

Проведений порівняльний аналіз існуючих на українському ринку програмних рішень. Більшість популярних систем добре виконують базові операційні завдання – відображення електронного каталогу страв, ведення складського обліку та здійснення оплати. Проте вони не мають вбудованих інтелектуальних інструментів для глибокого аналізу рентабельності меню. Власник отримує лише загальну статистику продажів, без автоматизованих рекомендацій щодо оптимізації асортименту.

На основі виявлених недоліків обґрунтовано необхідність створення нового комплексного ПЗ – системи «Scan & Savor». Інтеграція модуля керування прибутком, побудованого на матриці Касавани-Сміта, дозволить ефективно класифікувати страви за їхньою маржинальністю та популярністю. Додаткове застосування алгоритмів МВА безпосередньо у кошику відвідувача відкриває можливості для інтелектуального cross-selling, що збільшує середній чек закладу.

Отже, результати аналізу предметної області підтверджують високу актуальність та практичну цінність розроблюваного застосунку. Сформульовані висновки стануть основою для визначення функціональних вимог, вибору технологічного стека та подальшого проєктування архітектури системи.

2 ПРОЄКТУВАННЯ ТА МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

2.1 Постановка задачі та формування вимог до програмного забезпечення

На основі аналізу ринку, проведеного в першому розділі, метою дипломного проєкту є розробка веб-системи «Scan & Savor». Система автоматизує процес обслуговування клієнтів за допомогою QR-меню та надає адміністрації закладу інструменти для оцінки прибутковості та популярності страв.

Об'єктом розробки є програмна система, що складається з клієнтської частини для взаємодії з відвідувачами та серверної частини з вбудованим центром керування прибутком.

При розробці ПЗ необхідно реалізувати такі основні функціональні вимоги:

а) для клієнтського інтерфейсу:

- 1) швидкий доступ до меню через сканування QR-коду без обов'язкової реєстрації;
- 2) зручне відображення категорій страв та детальної інформації про кожну позицію: назва, інгредієнти, ціна, зображення страви;
- 3) повноцінне керування кошиком замовлення на стороні клієнта: додавання страв, зміна кількості, видалення;
- 4) показ інтелектуальних рекомендацій додаткових страв прямо в кошику на основі аналізу спільних покупок;

б) для адміністративного інтерфейсу з модулем оцінки меню:

- 1) автоматичний розрахунок рентабельності та популярності кожної страви;
- 2) візуалізація результатів інжинірингу меню за допомогою матриці Касавани-Сміта;
- 3) проведення багатовимірною ABC-аналізу для визначення головних джерел прибутку закладу харчування;

4) моніторинг динаміки щоденної виручки та кількості замовлень за допомогою графіків;

5) зручне керування базою даних страв і категорій, а також генерація нових QR-ідентифікаторів для столиків.

Окрім функціональних вимог, система повинна відповідати ключовим технічним критеріям: швидко завантажуватися на мобільних пристроях, витримувати пікові навантаження та стабільно працювати з великими масивами даних.

Завдання передбачає не лише написання коду, а й опрацювання математичних методів, що стануть основою центру керування прибутком. Тому наступним етапом буде детальний розгляд методологій, що використовуватимуться для меню-менеджменту.

2.2 Математичне обґрунтування методів аналізу рентабельності

Ефективне керування сучасним рестораном неможливе без регулярної оцінки меню. Для системи «Scan & Savor» обрано комбінований підхід, що поєднує два методи оцінки: багатовимірний ABC-аналіз та матрицю інжинірингу меню Касавани-Сміта. Це дозволяє оцінити кожен страву як за її внеском у загальний дохід, так і за рівнем прибутковості.

2.2.1 ABC-аналіз та принцип Парето

ABC-аналіз базується на фундаментальному принципі Парето (правило 80/20), який у контексті ресторанного бізнесу означає, що приблизно 20 % страв у меню дають до 80 % загального доходу або чистого прибутку закладу. У розробленій системі «Scan & Savor» реалізовано оцінку за двома ключовими критеріями: дохід від продажів та маржинальний прибуток.

Процес розрахунку та класифікації складається з таких математичних етапів.

Етап 1. Визначення загального доходу для кожної i -ї страви (2.1):

$$D_i = K_i \times P_i, \quad (2.1)$$

де K_i – кількість проданих порцій i -ї страви;

P_i – ціна продажу i -ї страви.

Етап 2. Сортуння списку страв за спаданням доходу D_i .

Етап 3. Обчислення накопичувального (кумулятивного) доходу (2.2):

$$Cum D_i = \sum_{i=1}^j D_i \quad (j = 1 \dots n), \quad (2.2)$$

Етап 4. Обчислення кумулятивного відсотка від загального доходу (2.3):

$$Cum \%_i = \frac{Cum D_i}{D_{total}} \times 100, \quad (2.3)$$

де $D_{total} = \sum_{i=1}^n D_i$ – загальний дохід від усіх проданих страв за обраний період.

На основі розрахованого накопичувального відсотка здійснюється сегментація асортименту за трьома порогами:

- категорія А (до 80 % доходу) – найбільш цінні позиції, що формують ядро доходу закладу;
- категорія В (наступні 15 %) – проміжні позиції зі стабільними показниками;
- категорія С (решта 5 %) – найменш значущі позиції, які є першочерговими кандидатами на заміну, зміну рецепту або виключення з меню.

Результати класифікації подано у вигляді кругової діаграми, що демонструє співвідношення між групами (рис. 2.1).

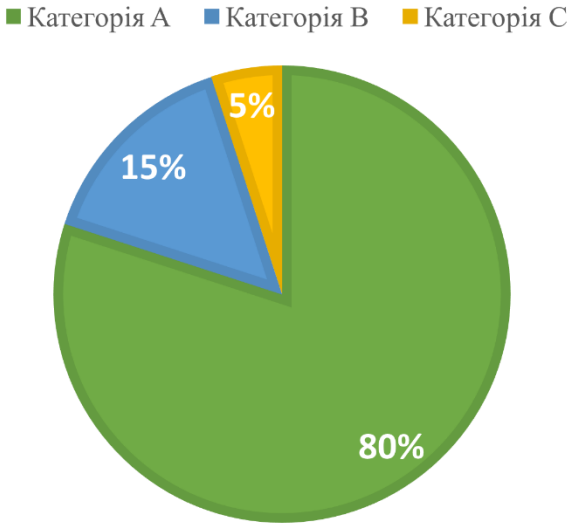


Рисунок 2.1 – Кругова діаграма ABC-аналізу асортименту страв

Для демонстрації роботи алгоритму було змодельовано вибірку з 10 страв. Результати обчислень індивідуального та накопичувального доходу наведено в таблиці 2.1.

Таблиця 2.1 – Вхідні дані та результати розрахунку доходу страв

Страва	Кількість	Ціна (грн)	Дохід (грн)	Накопичувальний дохід	Накопичувальний %
Стейк Рібай	65	320	20800	20800	17,2 %
Паста Карбонара	140	135	18900	39700	32,9 %
Борщ український	180	95	17100	56800	47,1 %
Піца Маргарита	110	145	15950	72750	60,3 %
Салат Цезар	95	110	10450	83200	68,9 %
Кава Еспресо	220	45	9900	93100	77,1 %
Салат Грецький	70	105	7350	100450	83,2 %
Чізкейк Нью-Йорк	75	95	7125	107575	89,1 %
Тірамісу	80	85	6800	114375	94,8 %
Суп крем з грибів	55	115	6325	120700	100,0 %

Для виявлення найбільш значущих позицій сформуємо діаграму Парето (рис. 2.2).

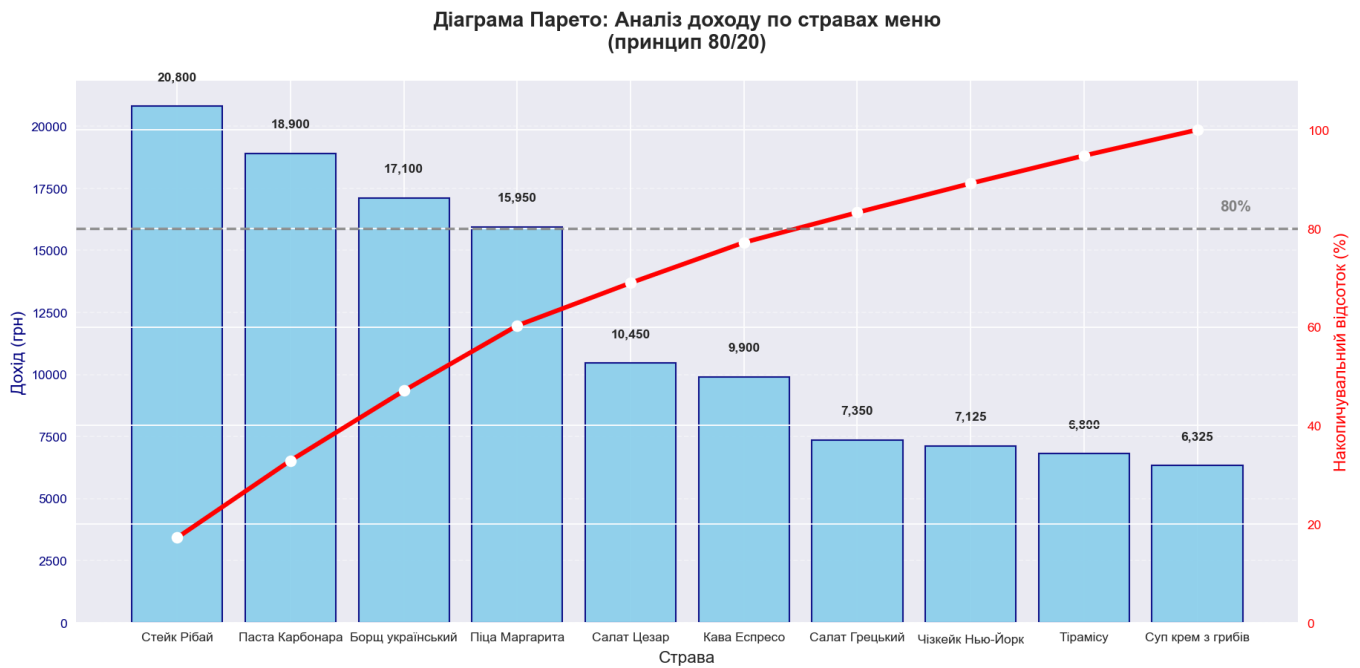


Рисунок 2.2 – Діаграма Парето для класифікації страв за принципом 20/80

На рисунку 2.2 сині стовпчики відображають індивідуальний дохід кожної страви, а червона лінія – динаміку накопичувального відсотка доходу. Пунктирна лінія на рівні 80 % візуалізує ключову межу принципу Парето.

З рисунку 2.2 та таблиці 2.2 чітко видно, що перші шість страв (від «Стейка Рібай» до «Кави Еспресо») приносять приблизно 77,1 % загального доходу закладу. Це вказує на необхідність зосередження уваги саме на цій групі «А»: посилити контроль якості інгредієнтів, слідкувати за наявністю позиції, активно просувати страву. Страви категорії «В» формують проміжну групу з відносно стабільними показниками, що потребують підтримки та періодичного моніторингу. Решта страв, що формують категорію «С», потребують додаткового дослідження за допомогою матриці Касавани-Сміта для прийняття рішення щодо їх присутності в меню.

2.2.2 Модель Касавани-Сміта

На відміну від ABC-аналізу, модель Касавани-Сміта фокусується на взаємозв'язку між популярністю страви та її маржинальним прибутком [23] -

[25]. Маржинальний прибуток (CM_i) визначається як різниця між ціною продажу та собівартістю (2.4):

$$CM_i = P_i - C_i, \quad (2.4)$$

де P_i – ціна одиниці i -ї страви;

C_i – собівартість одиниці i -ї страви.

Класифікація страв у матриці відбувається шляхом порівняння показників кожної страви з середніми значеннями по всьому меню.

Показник популярності розраховується (2.5):

$$MM_i = \frac{Q_i}{\sum_{k=1}^n Q_k}, \quad (2.5)$$

де Q_i – кількість проданих порцій i -ї страви;

$\sum_{k=1}^n Q_k$ – загальна кількість проданих порцій усіх страв у категорії або меню.

Середня маржа закладу розраховується (2.6):

$$CM_{avg} = \frac{\sum(CM_i \times Q_i)}{\sum Q_i}, \quad (2.6)$$

де $\sum(CM_i \times Q_i)$ – загальний маржинальний прибуток від усіх проданих порцій усіх страв;

$\sum Q_i$ – загальна кількість проданих порцій усіх страв.

У класичному підході для визначення порогів популярності використовується середнє арифметичне значення [23]. Проте в умовах ресторанного бізнесу продажі часто є нерівномірними: окремі дуже популярні позиції можуть суттєво зміщувати середнє вгору, що призводить до завищеного порогу та викривлення класифікації [9]. Щоб зменшити вплив таких статистичних викидів, у системі «Scan & Savor» застосовується розрахунок медіани, яка відображає

типове значення ряду та забезпечує більш збалансоване визначення порогів популярності. Значення порогу Th розраховується (2.7):

$$Th = Median(Q_1, Q_2 \dots Q_n), \quad (2.7)$$

Це дозволяє більш коректно розподіляти страви за чотирма квадрантами (табл. 2.2).

Таблиця 2.2 – Класифікаційні групи матриці Касавани-Сміта [3]

Популярність	Маржинальний внесок	
	Високий	Низький
Висока	Зірки (Stars)	Робочі конячки (Plow Horses)
Низька	Загадки (Puzzles)	Собаки (Dogs)

Зірки (Stars). Це найкращі позиції в меню. Вони добре продаються і мають високий маржинальний внесок. Такі страви приносять найбільший прибуток. Їх не варто суттєво змінювати – потрібно лише підтримувати якість, розмір порції та презентацію на належному рівні. Іноді можна перевірити еластичність ціни, але загалом «зірки» краще залишати як є [26].

Робочі конячки (Plow Horses). Назва говорить сама за себе – це «працьовиті» позиції меню. Вони дуже популярні серед гостей, але маржа внеску в них низька. Такі страви важливі, бо забезпечують стабільні продажі. Їх теж можна тестувати на зміну ціни, але вони часто чутливі до підвищення вартості. Щоб покращити ситуацію, варто спробувати знизити собівартість (наприклад, замінити інгредієнти або зменшити порцію) або поєднувати їх з іншими стравами, щоб збільшити загальний прибуток [26].

Загадки (Puzzles). Ці позиції мають високу маржу внеску, але продаються погано. Через низьку популярність їх часто розглядають як кандидатів

на вилучення з меню. Можливі дії: знизити ціну, щоб стимулювати продажі, або активніше просувати страву (кращий опис, розташування в меню, рекомендації офіціантів). Іноді допустимо залишити одну-дві «загадки», щоб показати різноманітність або преміум-сегмент. Але їх не повинно бути багато. Проблема в тому, що такі страви «заморожують» гроші в запасах і можуть призводити до псування продуктів [26].

Собаки (Dogs). Це найслабші позиції: низька популярність і низька маржа внеску. Зазвичай такі страви рекомендується повністю прибирати з меню. Якщо з певних причин хочеться їх залишити (наприклад, для іміджу), можна значно підвищити ціну – тоді вони можуть перейти в категорію «загадок». Якщо страва все ж лишається, її інгредієнти бажано обирати з тривалим терміном придатності, щоб уникнути втрат і псування [26].

2.3 Математичне забезпечення системи рекомендацій MBA

Для реалізації модуля інтелектуальних рекомендацій у кошику клієнта використано методику аналізу споживчого кошика MBA. Основна мета – знайти асоціативні правила виду $X \rightarrow Y$, де X – страва, що вже є в кошику, а Y – рекомендована страва [4], [27], [28].

Для оцінки сили зв'язку використовуються два ключові показники.

Підтримка (Support): частота, з якою набір страв зустрічається у загальній кількості замовлень N (2.8):

$$Supp(X \rightarrow Y) = \frac{Count(X \cap Y)}{N}, \quad (2.8)$$

Довіра (Confidence): ймовірність того, що клієнт замовить страву Y , якщо він уже вибрав страву X (2.9):

$$Conf(X \rightarrow Y) = \frac{Supp(X \rightarrow Y)}{Supp(X)}, \quad (2.9)$$

У розробленому ПЗ алгоритм налаштований на пошук товарів з найвищим показником довіри, що дозволяє динамічно генерувати блок рекомендацій страв, спираючись на реальну історію транзакцій закладу. Це забезпечує математично обґрунтований підхід до збільшення середнього чека без нав'язливої реклами [4], [6]-[8], [27], [28].

2.4 Вибір та обґрунтування технологічного стека

Розробка ПЗ для ресторанного бізнесу потребує правильного вибору технологій. Система «Scan & Savor» поєднує дві складові: швидку обробку замовлень через QR-меню та модуль для роботи з великими масивами даних. Тому технологічний стек має забезпечувати баланс між продуктивністю веб-сервера та можливостями обробки даних.

2.4.1 Вибір мови програмування та серверного фреймворку

Серверна частина є ядром системи, оскільки саме тут відбувається обробка бізнес-логіки, взаємодія з базою даних та виконання складних математичних розрахунків.

Для реалізації серверної логіки розглядалися три найбільш популярні на сьогодні мови веб-програмування: PHP, JavaScript (платформа Node.js) та Python. Кожна з них має свої переваги, проте розроблюваний проєкт потребує особливих вимог до обробки даних. Для прийняття остаточного рішення було проведено порівняльний аналіз, результати якого наведено у таблиці 2.3.

Таблиця 2.3 – Порівняльний аналіз мов програмування

Критерій	PHP	JavaScript (Node.js)	Python
Основна сфера застосування	Класичні веб-сайти, CMS	Високонавантажені real-time застосунки, SPA	Веб-розробка, Data Science, машинне навчання
Інструментарій для роботи з даними	Слабкий	Обмежений	Найширший
Швидкість розробки	Висока	Середня	Висока
Асинхронність та продуктивність	Низька	Дуже висока	Середня
Читабельність та підтримка коду	Середня	Середня	Висока

З аналізу таблиці 2.3 видно, що Node.js має перевагу у швидкості роботи з великою кількістю з'єднань, а PHP залишається поширеним вибором для хостингів. Проте для розроблюваного проєкту беззаперечним лідером є Python завдяки великій кількості бібліотек для роботи з даними. Використання PHP чи Node.js вимагало б написання великої кількості низькорівневого коду для реалізації математичних алгоритмів, що збільшує ризик помилок. Python дозволяє реалізувати ці обчислення просто й оптимізовано завдяки вбудованим бібліотекам.

Після затвердження мови програмування необхідно було обрати фреймворк, який би забезпечив швидку і безпечну розробку. У екосистемі Python домінують три основні рішення: Flask, FastAPI та Django [29]-[31]. Порівняння цих фреймворків представлено в таблиці 2.4.

Таблиця 2.4 – Порівняльний аналіз веб-фреймворків мови Python

Критерій	Flask [29]	FastAPI [30]	Django [31]
Архітектурний підхід	Мікрофреймворк (мінімалізм)	Мікрофреймворк (сучасний, асинхронний)	Моноліт
Вбудована ORM (робота з БД)	Немає (потребує інтеграції)	Немає (потребує зовнішніх бібліотек)	Є вбудована ORM
Панель адміністратора	Відсутня	Відсутня	Генерується автоматично
Система аутентифікації та безпеки	Налаштовується вручну	Налаштовується вручну (OAuth2)	Вбудована, захист від CSRF, SQLi, XSS
Швидкодія API	Середня	Дуже висока	Середня

За результатами порівняння таблиці 2.4 серверним фреймворком було обрано Django. Його перевага полягає у готових інструментах, які дозволяють швидко створити систему керування рестораном без розробки базових компонентів з нуля [31]. Django надає дві ключові можливості для розроблюваного проєкту:

- перша це вбудована панель адміністратора Django Admin: забезпечує інтерфейс для управління меню, категоріями та замовленнями. Її гнучкість дозволяє інтегрувати власні дашборди з матрицями та графіками, створюючи єдиний зручний простір для керівника закладу [31];
- друга це потужна система ORM: дозволяє виконувати складні агрегаційні запити для алгоритмів рекомендацій MBA оптимізовано на рівні реляційної алгебри СКБД, що забезпечує високу швидкість формування рекомендацій [31].

Таким чином, поєднання Python та Django є оптимальним архітектурним рішенням, яке гарантує швидку розробку, надійність платформи та можливість інтеграції складних модулів підтримки прийняття рішень.

2.4.2 Вибір системи керування базами даних

Оскільки система «Scan & Savor» виконує складні обчислення в реальному часі, вибір СКБД є одним із ключових архітектурних рішень. У розроблюваному проєкті база даних виступає не просто сховищем, а й обчислювальним ядром для меню-менеджменту.

Для проєкту розглядалися три сучасні СКБД різних типів: MongoDB (NoSQL), MySQL (реляційна) та PostgreSQL (об'єктно-реляційна) [32]-[34].

Порівняльний аналіз цих систем за критеріями наведено у таблиці 2.5.

Таблиця 2.5 – Порівняльний аналіз систем керування базами даних

Критерій	MongoDB [32]	MySQL [33]	PostgreSQL [34]
Структура даних	Документо-орієнтована (JSON)	Таблична (Реляційна)	Таблична (Об'єктно-реляційна)
Підтримка ACID (надійність транзакцій)	Часткова	Повна	Повна
Складні аналітичні запити	Обмежені (потребується агрегація на сервері)	Середні	Відмінні
Підтримка математичних функцій	Відсутня	Відсутня	Присутня
Сумісність з Django ORM	Низька (потребує сторонніх драйверів)	Висока	Максимальна

Аналізуючи дані таблиці 2.5 необхідно було зробити вибір між реляційним (SQL) та нереляційним (NoSQL) підходами.

Хоча бази даних типу MongoDB демонструють високу швидкість запису неструктурованих даних, їх використання в цьому проєкті є недоцільним [32].

Дані ресторанного бізнесу мають чітку та строгу реляційну структуру: «Категорія» містить «Страви» які додаються в «Замовлення» (через проміжну таблицю «Позиції замовлення»). Відсутність строгих зв'язків через ключі у NoSQL базах створює ризик порушення цілісності фінансових даних. Крім того, для реалізації алгоритму спільних покупок MBA необхідно здійснювати пошук перетинів страв у різних замовленнях, що в MongoDB вимагатиме завантаження великих обсягів даних в оперативну пам'ять сервера для обробки [32].

Тому вибір звузився до двох основних реляційних систем – MySQL та PostgreSQL. Обидві є надійними та безкоштовними, проте для архітектури розроблюваного проєкту було обрано PostgreSQL. Це рішення має три ключові технічні переваги:

- статистичні функції: у розроблюваному ПЗ використовується медіана для визначення порогу популярності в матриці Касавани-Сміта. MySQL не має вбудованої функції розрахунку медіани, що змусило б виконувати обчислення у Python через Pandas. PostgreSQL підтримує сучасні віконні функції, що дозволяє базі даних швидко розраховувати медіану самостійно, зменшуючи навантаження на веб-сервер [34];
- оптимізація JOIN-запитів: для MBA необхідно аналізувати величезну кількість зв'язків «страва-страва» в межах одного замовлення. Планувальник запитів PostgreSQL ефективніше працює зі складними агрегаціями, ніж MySQL, що забезпечує швидке формування рекомендацій у реальному часі [34];
- сумісність із Django: офіційна документація фреймворку рекомендує PostgreSQL як оптимальну СКБД. Django ORM має модуль postgres, який відкриває доступ до специфічних типів даних та повнотекстового пошуку, що створює можливості для подальшого масштабування системи [34].

Таким чином, використання об'єктно-реляційної СКБД PostgreSQL забезпечує розроблюваній системі «Scan & Savor» абсолютну надійність збереження транзакцій клієнтів та надає потужний обчислювальний інструментарій для миттєвої генерації звітності керівництву закладу.

2.4.3. Інструменти оцінки та роботи з даними

Django та PostgreSQL забезпечують надійну основу для транзакційної роботи системи – збереження замовлень та керування меню. Однак для блоку меню-менеджменту стандартних можливостей Python та ORM-запитів недостатньо. Реалізація моделі Касавани-Сміта та багатовимірного ABC-аналізу потребує складних матричних обчислень, сортування масивів, розрахунку кумулятивних сум і статистичних показників на великих наборах минулих замовлень. Виконання тисяч записів через звичайні цикли `for` у Python є надто ресурсозатратним і може призвести до перевантаження сервера під час формування звітності [35].

Для вирішення цієї проблеми у стек технологій було використано спеціалізовану бібліотеку для роботи з даними – Pandas [36].

Переваги бібліотеки Pandas:

- векторизація обчислень: на відміну від стандартних списків Python, Pandas працює з `DataFrame` та `Series`, реалізованими на C та Cython. Це дозволяє виконувати математичні операції над цілими колонками даних без використання циклів, що значно прискорює розрахунки [36];
- агрегація та групування: метод `groupby` дає змогу швидко згрупувати продажі за періодами чи категоріями страв та отримати сумарний дохід і собівартість [36];
- статистичні функції: Pandas має готові методи для розрахунку кумулятивних відсотків та медіани, що спрощує реалізацію ABC-аналізу та матриці Касавани-Сміта [36];

– інтеграція з Django ORM: дані з PostgreSQL легко конвертуються у DataFrame. Це дозволяє використовувати базу лише для швидкого отримання транзакцій, а складні обчислення виконувати оптимізованим ядром Pandas у пам'яті сервера [36].

Для візуалізації даних використовується клієнтський рендеринг, адже серверна генерація графіків за допомогою бібліотеки Matplotlib створює додаткове навантаження на процесор і не забезпечує інтерактивності. Клієнтський підхід дозволяє реалізувати функції взаємодії, наприклад відображення деталей страви при наведенні курсору.

Таким чином, Pandas виступає аналітичним ядром системи, яке виконує складну математичну обробку даних, тоді як Django відповідає за маршрутизацію та безпеку.

2.4.4. Технології клієнтської частини

Архітектура клієнтської частини розроблюваної системи «Scan & Savor» поділяється на два інтерфейси з різними вимогами. Перший – це гостьове QR-меню, де критично важливою є швидкість завантаження та адаптивність під будь-які екрани смартфонів. Другий – це адміністративна панель з дашбордами керівника, де необхідна зручність роботи з великими таблицями та інтерактивними графіками.

Оскільки основою системи є фреймворк Django, що має механізм шаблонізації, використання важких JavaScript-фреймворків було неоптимальним рішенням. Тому для фронтенд-розробки було обрано класичний та оптимізований стек технологій.

Візуальне оформлення системи реалізоване за допомогою HTML5, CSS3 та фреймворку Bootstrap 5. Його використання забезпечує принцип Mobile-First, що критично важливо для QR-меню. Завдяки готовим компонен-

там, Bootstrap дозволяє швидко створювати інтерфейс і гарантує коректне відображення на різних мобільних пристроях без написання великої кількості власного CSS [37].

Клієнтська логіка реалізована на сучасному JavaScript без використання застарілих бібліотек [38]. Він відповідає за три основні процеси:

- керування кошиком: додавання страв, зміна кількості, підрахунок суми через localStorage;
- асинхронні запити до сервера для отримання рекомендацій MBA;
- керування QR-кодами для столиків у панелі адміністратора.

Для візуалізації результатів системи бізнес-процесів використовується легка бібліотека Chart.js, яка будує інтерактивні графіки на елементі <canvas>. Вона забезпечує зручну взаємодію з даними і дозволяє відмовитися від серверної генерації зображень [39].

Обраний стек клієнтських технологій забезпечує невеликий розмір файлів для швидкого завантаження через мобільний інтернет у закладах та створює сучасний, зручний інтерфейс без зайвого ускладнення архітектури проєкту.

2.5 Висновки за розділом 2

На основі аналізу проблемної області було сформульовано чітку постановку задачі та визначено ключові функціональні вимоги до програмного забезпечення як з боку гостей закладу, так і з боку адміністрації. Доведено, що система повинна не лише автоматизувати прийом замовлень, а й генерувати рекомендації для оптимізації меню.

Було детально описано та математично обґрунтовано використання методів оцінки рентабельності меню. Зокрема, розкрито алгоритм багатовимірного ABC-аналізу з використанням принципу Парето. Також наведено модифікацію матриці інжинірингу меню Касавани-Сміта: замість середнього ариф-

метичного для порогу популярності застосовується медіана. Це дозволяє зменшити вплив статистичних викидів і зробити класифікацію страв за квадрантами більш об'єктивною. Крім того, подано теоретичну базу для реалізації модуля MBA, який відповідає за автоматичну генерацію рекомендацій в кошику.

Для реалізації алгоритмів було проведено аналіз сучасних засобів розробки та визначено оптимальний технологічний стек. Серверна частина реалізована на Python з використанням Django, що забезпечує швидкість і безпеку. Для зберігання реляційних даних та виконання складних запитів застосовано PostgreSQL. Обробку великих масивів даних виконує бібліотека Pandas. Інтерфейс користувача реалізовано за допомогою HTML5, Bootstrap 5, JavaScript та Chart.js, що гарантує адаптивність і інтерактивність.

Результати цього розділу формують теоретичну основу, що дозволяє перейти до практичної розробки архітектури розроблюваного ПЗ, проектування бази даних та написання вихідного коду.

3 ОПИС ПРОГРАМИ

3.1 Проектування архітектури та структури бази даних

Для створення гнучкої системи автоматизації закладів громадського харчування використано клієнт-серверну архітектуру. Вона забезпечує централізоване зберігання даних та розділяє бізнес-логіку від відображення інтерфейсу.

Серверна частина на Django працює з базою даних PostgreSQL, обробляє запити, виконує автентифікацію та розрахунки для меню-менеджменту. Клієнтська частина складається з двох підсистем: мобільного QR-меню для відвідувачів та десктопного дашборду для керівництва.

Надійна робота модуля формування замовлень та модуля оцінки позицій в меню залежить від правильної структури бази даних. Для системи «Scan & Savor» створено нормалізовану реляційну схему, яка мінімізує дублювання та забезпечує швидкий доступ до історії замовлень для алгоритмів MBA і матриці Касавани-Сміта.

База даних складається з таких основних таблиць:

- таблиця категорії (Category) зберігає інформацію про розділи страв у меню. Це дозволяє групувати й структурувати страви у клієнтському інтерфейсі;
- таблиця страви (Dish) містить головні сутності асортименту. Містить такі поля: назва, опис, фотографія, ціна продажу та собівартість. Наявність собівартості є критично важливою для роботи модуля оцінки позицій меню, оскільки вона дозволяє розраховувати маржинальний прибуток;
- таблиця столиків (Table) зберігає інформацію про фізичні столики в закладі. Для кожного столика генерується унікальний ідентифікатор (UUID), на основі якого створюється персоналізований QR-код;
- замовлення (Order) – таблиця, що фіксує факт створення замовлення. Містить інформацію про час створення, статус та прив'язку до конкретного столика;

– позиції замовлення (OrderItem) – це проміжна таблиця, що реалізує зв’язок «багато-до-багатьох» між Order та Dish. Вона зберігає кількість замовлених порцій конкретної страви та фіксує ціну на момент продажу (це необхідно для коректної аналітики, оскільки ціна страви може змінюватися з часом).

Візуальне представлення структури бази даних та зв’язків між сутностями наведено на ER-діаграмі (рис. 3.1).

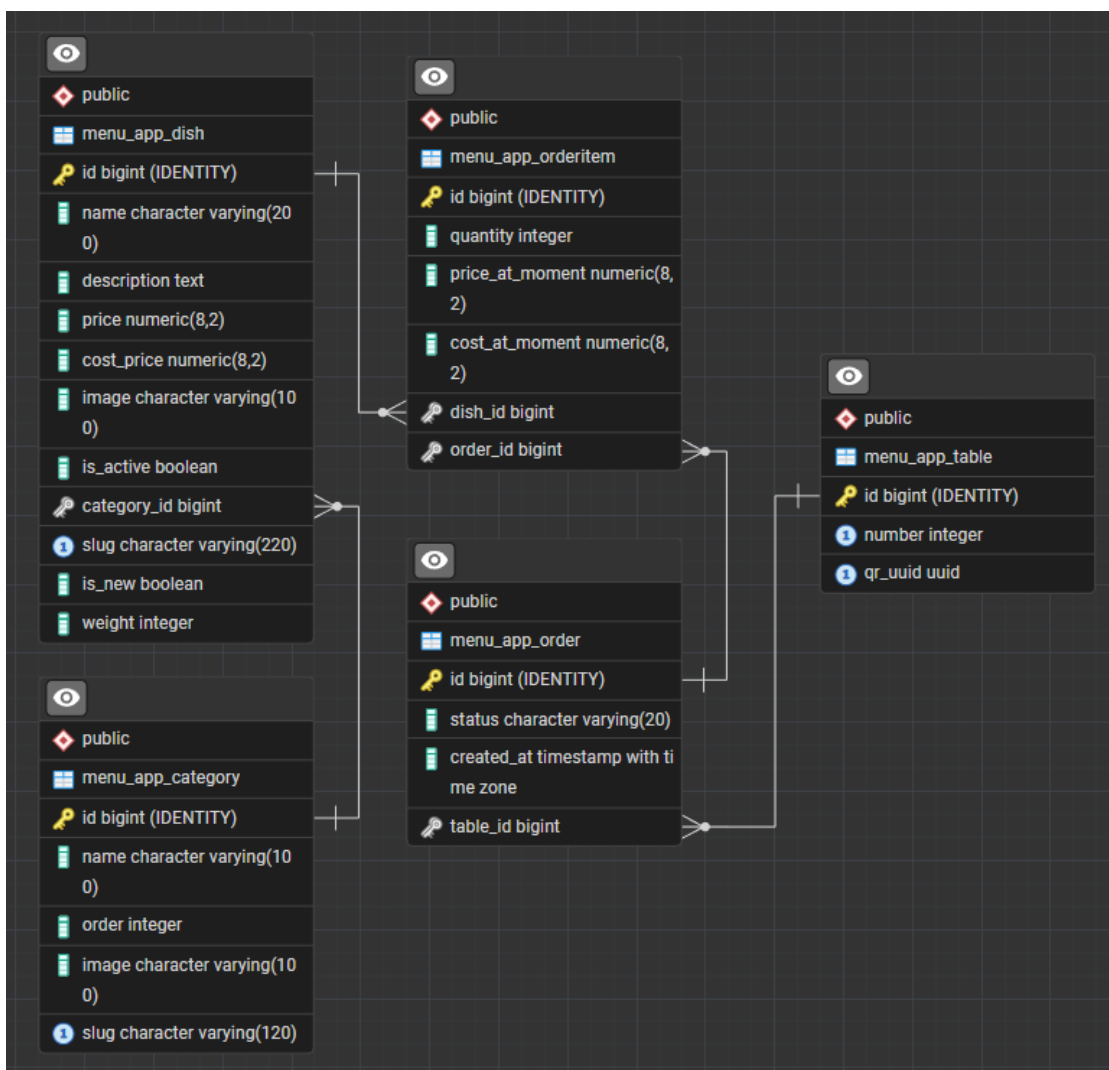


Рисунок 3.1 – ER-діаграма бази даних системи

Розроблена структура бази даних повністю відповідає вимогам нормалізації та забезпечує ефективне зберігання великих обсягів замовлень й позицій, необхідних для подальшої статистичної обробки за допомогою бібліотеки Pandas.

3.2 Реалізація серверної логіки

Серверна частина програмного комплексу побудована на основі фреймворку Django, що реалізує архітектурний патерн Model-View-Template. Основна увага при розробці була приділена створенню стійкої архітектури, яка забезпечує швидку обробку транзакцій замовлень та виконання складних розрахункових операцій без перевантаження сервера.

3.2.1 Структура програмного модуля

Проект має модульну структуру, де основна функціональність зосереджена в застосунку `menu_app`. Глобальні налаштування, конфігурація бази даних та маршрутизація верхнього рівня винесені в пакет `core`. Такий підхід відповідає стандартам чистої архітектури та дозволяє легко масштабувати програмний засіб.

Внутрішня структура проєкту та ієрархія його компонентів зазначені нижче.

Коренева директорія:

- `manage.py`. Головний скрипт для управління Django-проєктом, запуску локального сервера та виконання адміністративних команд;
- директорія конфігурації проєкту (`core/`);
- головний додаток системи (`menu_app/`);
- бізнес-логіка та сервіси (`menu_app/services/`);
- користувацькі команди (`menu_app/management/commands/`);
- директорія шаблонів сторінок (`templates/`).

Директорія конфігурації проєкту (core/):

- settings.py. Файл з основними конфігураціями проєкту, підключенням до бази даних PostgreSQL та ініціалізацією змінних оточення;
- urls.py. Головний маршрутизатор проєкту, який об'єднує всі URL-адреси додатків та панелі адміністратора.

Головний додаток системи (menu_app/):

- models.py. Описує структуру бази даних, включаючи моделі категорій, страв, столиків та позицій замовлень;
- views.py. Містить контролери для обробки запитів клієнтських сторінок, таких як перегляд меню та сканування QR-кодів;
- views_api.py. Реалізує API-ендпоінти для створення замовлень у базі даних та отримання рекомендацій страв для кошика;
- views_admin.py. Містить логіку обробки даних для відображення кастомної моніторингової панелі в інтерфейсі адміністратора;
- urls.py Визначає локальні маршрути (URL-адреси) для клієнтських сторінок, API та функцій адміністратора всередині додатку;
- admin.py. Налаштовує відображення моделей БД у стандартній адмін панелі Django та додає функціонал генерації QR-кодів;
- apps.py. Файл налаштувань проєкту, що визначає базові параметри додатку керування меню та моніторингу;
- tests.py. Містить модульні тести для перевірки правильності розрахунків у бізнес-моделях та безпеки API-запитів.

Бізнес-логіка та сервіси (menu_app/services/):

- dashboard.py. Реалізує математичні алгоритми для розрахунку матриці Касавани-Сміта, ABC-аналізу та динаміки продажів;
- recommendations.py. Містить алгоритм пошуку асоціативних правил МВА для рекомендації страв, які найчастіше купують разом;
- qr_code_generator.py. Відповідає за програмну генерацію графічних QR-кодів із вбудованим логотипом ресторану по центру.

Користувачькі команди (menu_app/management/commands/):

- generate_orders.py. Скрипт для автоматичної генерації випадкових тестових замовлень за останні 30 днів для перевірки роботи модуля оцінки позицій в меню;
- generate_qrs.py. Команда для масової генерації та збереження фізичних зображень QR-кодів для всіх створених столиків;
- load_menu.py. Скрипт для початкового автоматизованого наповнення бази даних тестовим асортиментом категорій та страв із розрахунком собівартості.

Директорія шаблонів сторінок (templates/):

- base.html. Базовий каркас веб-сторінок, який містить навігаційну панель, логіку роботи кошика та підключені стилі;
- index.html. Шаблон головної сторінки, що відповідає за візуальне відображення списку доступних категорій меню;
- category_detail.html. Шаблон сторінки конкретної категорії для виводу переліку страв із можливістю додавання їх у кошик;
- about.html. Інформаційна сторінка клієнтської частини з описом закладу, контактами та годинами роботи;
- admin/monitoring_dashboard.html. Шаблон кастомної панелі моніторингу меню з графіками рентабельності та статистикою продажів для адміністратора.

Структура проєкту у Visual Studio Code зображена на рисунку 3.2.

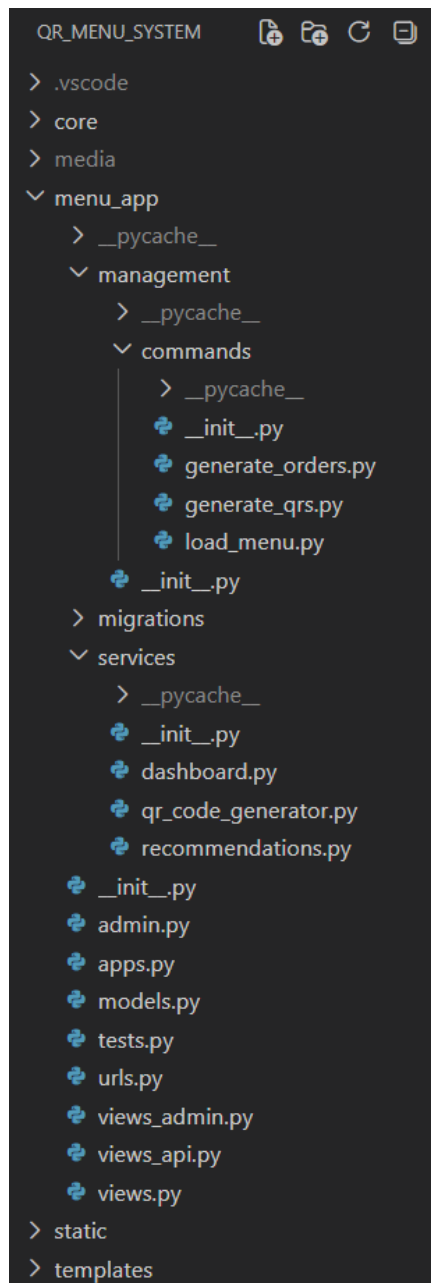


Рисунок 3.2 – Ієрархічна структура проєкту у Visual Studio Code

3.2.2 Програмна реалізація ORM

Для забезпечення цілісності даних використано Django ORM, що дозволяє працювати з сутностями бази даних як з об'єктами класів. Схему взаємодії програмного коду, Django ORM та PostgreSQL наведено на рисунку 3.3.

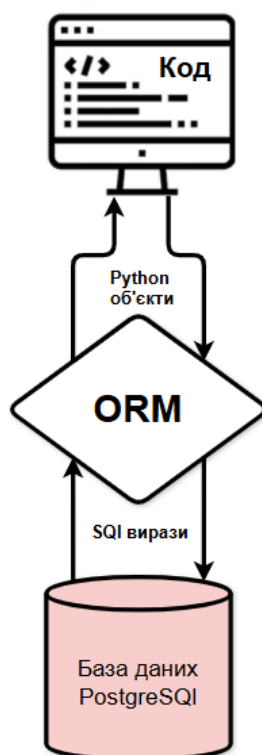


Рисунок 3.3 – Схема взаємодії програмного коду з БД через Django ORM

У файлі `models.py` реалізовано ієрархію класів, що описують предметну область:

- клас `Dish` містить атрибути для зберігання необхідних для оцінки рентабельності параметрів страв;
- клас `Order` та пов'язаний з ним `OrderItem` забезпечують цілісність транзакцій. При створенні `OrderItem` фіксується поточна ціна та собівартість страви. Це гарантує, що історичні записи залишаються незмінними для майбутніх розрахунків, навіть якщо характеристики страви згодом зміняться;
- клас `Category` містить атрибути необхідні для опису категорій страв та порядку їх відображення на головній сторінці;
- клас `Table` містить атрибут номеру столика та його унікальний UUID, що буде використовуватися для генерування посилань для QR-кодів.

Зв'язки між об'єктами реалізовані через зовнішні ключі з контролем цілісності на рівні бази даних.

3.2.3 Обробка запитів та контролери

Логіка керування Views може працювати у двох режимах. Перший це синхронний: сервер формує та відображає сторінки меню через шаблонізатор. Другий асинхронний: операції з кошиком виконуються без повного перезавантаження сторінки, дані оновлюються миттєво.

Для реалізації функціонала моніторингу та оцінки рентабельності розроблено окремий сервісний шар у папці services/. Це дозволяє винести складні математичні обчислення (ABC-сортування, розрахунок медіанних значень для класифікації страв) за межі контролерів.

Схема потоку даних для відображення інформації про позиції в меню у вигляді графіків в панелі адміністратора зображена на рисунку 3.4.

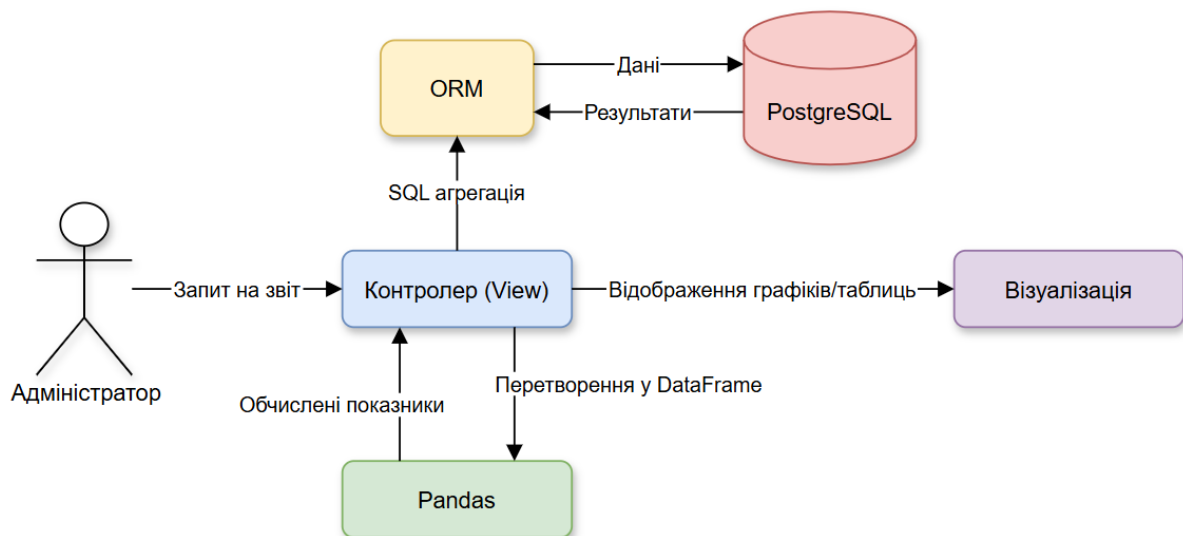


Рисунок 3.4 – Схема потоку даних для відображення даних позицій в меню

При відкритті розділу звітів адміністратором контролер виконує агрегаційні SQL-запити до PostgreSQL через ORM, потім перетворює дані у структури Pandas для швидких векторних обчислень і наприкінці повертає значення показників (маржинальність, популярність, категорія за матрицею Каса-вани–Сміта) для візуалізації.

3.2.4 Забезпечення безпеки та валідація даних

На рівні серверної логіки реалізовано декілька механізмів захисту від поширених вразливостей.

Захист від фальшивих замовлень: щоб відвідувачі не могли створювати фальшиві замовлення, знаходячись поза межами закладу, реалізовано механізм часу життя сесії. Після сканування QR-коду сесія прив'язується до столика лише на 2 години, після чого автоматично анулюється.

Безпечна обробка транзакцій: у модулі API для створення замовлення ідентифікатор столика витягується виключно із захищеної серверної сесії, а не з клієнтського запиту. Це унеможлиблює підміну ID столика з боку злоумисників.

Механізм зміни ідентифікаторів: для захисту скомпрометованих QR-кодів впроваджено функцію оновлення унікального ID столика в один клік через дії адміністратора в панелі Django. Це дозволяє миттєво регенерувати старий код і згенерувати новий без видалення столика з БД.

Використання Django як основного фреймворку дозволило зосередитися на розробці унікальних алгоритмів обробки показників ефективності меню, спираючись на надійні вбудовані інструменти роботи з сесіями та автентифікацією користувачів [31].

3.3. Програмна реалізація розрахункових модулів

Щоб система швидко обробляла великі масиви даних, розрахункову логіку винесено в окремий сервісний шар додатку `menu_app`. Це дозволило відокремити математичні алгоритми від контролерів і зробити код чистішим. Для роботи з даними використовується бібліотека `Pandas`, яка забезпечує ефективні векторні операції з таблицями `DataFrame` [35], [36].

3.3.1 Програмна реалізація ABC-аналізу

Процес розрахунку сегментів А, В та С реалізовано у вигляді методу, який послідовно виконує трансформацію сирих даних з бази даних у класифікований список. Алгоритм сегментації ABC-аналізу зображений на рисунку 3.5.

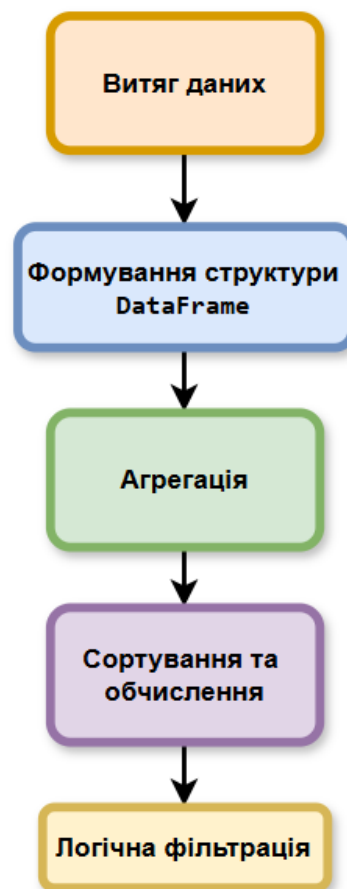


Рисунок 3.5 – Алгоритм сегментації ABC-аналізу

Опис етапів алгоритму сегментації ABC-аналізу.

Етап 1. Екстракція даних: за допомогою Django ORM формується набір даних QuerySet, що містить інформацію про продані позиції з таблиці OrderItem.

Етап 2. Формування структури DataFrame: отримані дані конвертуються в об'єкт Pandas для подальшої обробки.

Етап 3. Агрегація: дані групуються за ідентифікатором страви за допомогою методу `.groupby()`, де обчислюється сумарний дохід для кожної позиції.

Етап 4. Сортвання та кумулятивні обчислення: за допомогою вбудованої функції `.cumsum()` розраховується накопичувальний підсумок.

Етап 5. Логічна фільтрація: застосовуються умовні конструкції для присвоєння кожній страві відповідної категорії на основі розрахованих відсоткових порогів.

Використання векторизованих методів Pandas дозволяє уникнути ітераційних циклів на рівні Python, що критично важливо для продуктивності програмного засобу при збільшенні кількості замовлень.

3.3.2 Програмна реалізація методу Касавани-Сміта

Найбільш складною частиною розрахункового модуля є реалізація класифікації страв за параметрами популярності та маржинальності. У програмному коді реалізовано метод розрахунку порогів на основі значень медіан, що підвищує стійкість алгоритму до статистичних викидів.

Спочатку розраховується маржинальний дохід. На рівні `DataFrame` створюється обчислювальний стовпець, де для кожного рядка виконується операція віднімання собівартості від ціни продажу.

Далі виконується обчислення медіанних порогів. Замість стандартного середнього значення використовується функція `.median()`. Це дозволяє алгоритму автоматично адаптуватися під реальний розподіл попиту в закладі.

Наприкінці алгоритму відбувається розподіл страв за квадрантами. Реалізовано логічну функцію, яка порівнює показники кожної страви з медіанами популярності та прибутку і присвоює об'єкту відповідну категорію: «Зірка», «Конячка», «Загадка» або «Собака».

Фрагмент коду з обчисленням значення медіани для популярності, середнього значення для прибутковості та класифікацією за категорією зображено на рисунку 3.6.

```

# Для популярності використовується медіана
threshold_sold = df['total_sold'].median()

# Для прибутковості використовується середня питома маржа
threshold_margin = df['unit_margin'].mean()

def categorize_and_advise(row):
    is_high_sold = row['total_sold'] >= threshold_sold
    is_high_margin = row['unit_margin'] >= threshold_margin

    if is_high_sold and is_high_margin:
        return pd.Series(['Зірка (Star)', 'success', 'Зберегти якість. Рекламувати.'])
    elif is_high_sold and not is_high_margin:
        return pd.Series(['Конячка (Plow Horse)', 'warning', 'Обережно підняти ціну або зменшити порцію.'])
    elif not is_high_sold and is_high_margin:
        return pd.Series(['Загадка (Puzzle)', 'info', 'Високий потенціал. Додати в Рекомендації.'])
    else:
        return pd.Series(['Собака (Dogs)', 'danger', 'Видалити з меню або повністю змінити.'])

```

Рисунок 3.6 – Фрагмент коду модуля дашборду панелі адміністратора

3.3.3 Програмна реалізація модуля асоціативних правил MBA

Модуль рекомендацій MBA визначає, які страви найчастіше зустрічаються разом у замовленнях. Завантаження всієї історії транзакцій у пам'ять сервера було б неефективним і при великій базі даних призвело б до різкого падіння продуктивності. Тому обчислення винесено на рівень СКБД PostgreSQL через агрегаційні механізми Django ORM. Це дозволяє перетворити пошук асоціативних правил на оптимізований SQL-запит із JOIN та GROUP BY. Алгоритм реалізовано у функції `get_sell_recommendations`, яка приймає список ідентифікаторів страв із кошика користувача та повертає список рекомендованих позицій. Фрагмент коду модуля рекомендацій міститься на рисунку 3.7.

```
def get_sell_recommendations(cart_dish_ids, limit=4):
    # MVP алгоритм асоціативних правил (Market Basket Analysis)
    # Пошук товарів, які найчастіше купують разом із товарами в кошику

    if not cart_dish_ids:
        return []

    # Пошук всіх замовлень, в яких є хоча б одна страва з кошика
    orders_with_cart_items = OrderItem.objects.filter(
        dish_id__in=cart_dish_ids
    ).values('order')

    # Пошук інших страв в замовленнях, підрахунок їх кількості, сортування
    recommendations = Dish.objects.filter(
        orderitem__order__in=orders_with_cart_items,
        is_active=True # Для рекомендації позиція повинна бути в наявності
    ).exclude(
        id__in=cart_dish_ids # Якщо позиція вже в кошику - ігнорувати
    ).annotate(
        times_bought_together=Count('orderitem__order')
    ).order_by('-times_bought_together')[:limit]
```

Рисунок 3.7 – Фрагмент коду модуля рекомендацій

Процес алгоритмічної обробки розділено на три ключові етапи.

Етап 1. Створюється запит `orders_with_cart_items`, який фільтрує таблицю позицій замовлень і знаходить ідентифікатори тих чеків, що містять хоча б одну страву з поточного кошика.

Етап 2. Виконується запит до таблиці `Dish`. Метод `.filter()` залишає лише ті об'єкти, які пов'язані зі знайденими на попередньому етапі замовленнями, та перевіряє прапор `is_active`, щоб не рекомендувати страви, яких немає в наявності. Метод `.exclude()` гарантує, що клієнту не буде запропоновано те, що він уже додав. Ключовою інженерною оптимізацією є використання методу `.annotate()` разом із функцією `Count`. Він транслюється СКБД в операцію групування та підрахунку зв'язків, формуючи показник `times_bought_together`. Відразу після цього застосовується сортування за спаданням та обмеження вибірки (`limit`).

Етап 3. Отриманий QuerySet перетворюється на список словників. Перетворення типів гарантує безпроблемну подальшу конвертацію результату у формат JSON для передачі клієнтському застосунку через REST-інтерфейс.

Така організація дозволяє швидко обчислювати асоціативні зв'язки у реальному часі. Основне навантаження переноситься з Python на рівень реляційної алгебри бази даних [27], [31].

3.4 Проєктування та реалізація користувацьких інтерфейсів

Інтерфейси системи «Scan & Savor» спроектовані з урахуванням функціональності та зручності користування. Оскільки система має дві групи користувачів – гостей і адміністраторів – створено окремі інтерфейси, кожен із власними технічними вимогами.

3.4.1 Концепція архітектури клієнтської частини

Для побудови візуальної частини використано фреймворк Bootstrap 5, який дозволив реалізувати підхід Mobile-First. Це критично важливо для гостьового інтерфейсу, оскільки основний сценарій використання передбачає роботу зі смартфона після сканування QR-коду.

На рівні інтерфейсу реалізовано технічні рішення, що забезпечують адаптивність і швидку роботу системи. Завдяки сітці Grid картки страв коректно відображаються на будь-яких екранах – від смартфонів до моніторів з високою роздільною здатністю. Події кошика обробляються асинхронно за допомогою JavaScript, тому сторінки не перезавантажуються і інтерфейс працює швидше. Стан кошика зберігається у localStorage на стороні клієнта, що гарантує збереження вибраних позицій навіть при випадковому оновленні сторінки чи втраті з'єднання.

3.4.2 Реалізація інтерфейсу користувача

Гостьовий інтерфейс побудований так, щоб користувач швидко переходив від сканування QR-коду до оформлення замовлення. На головній сторінці відображається список карток страв (рис. 3.8).

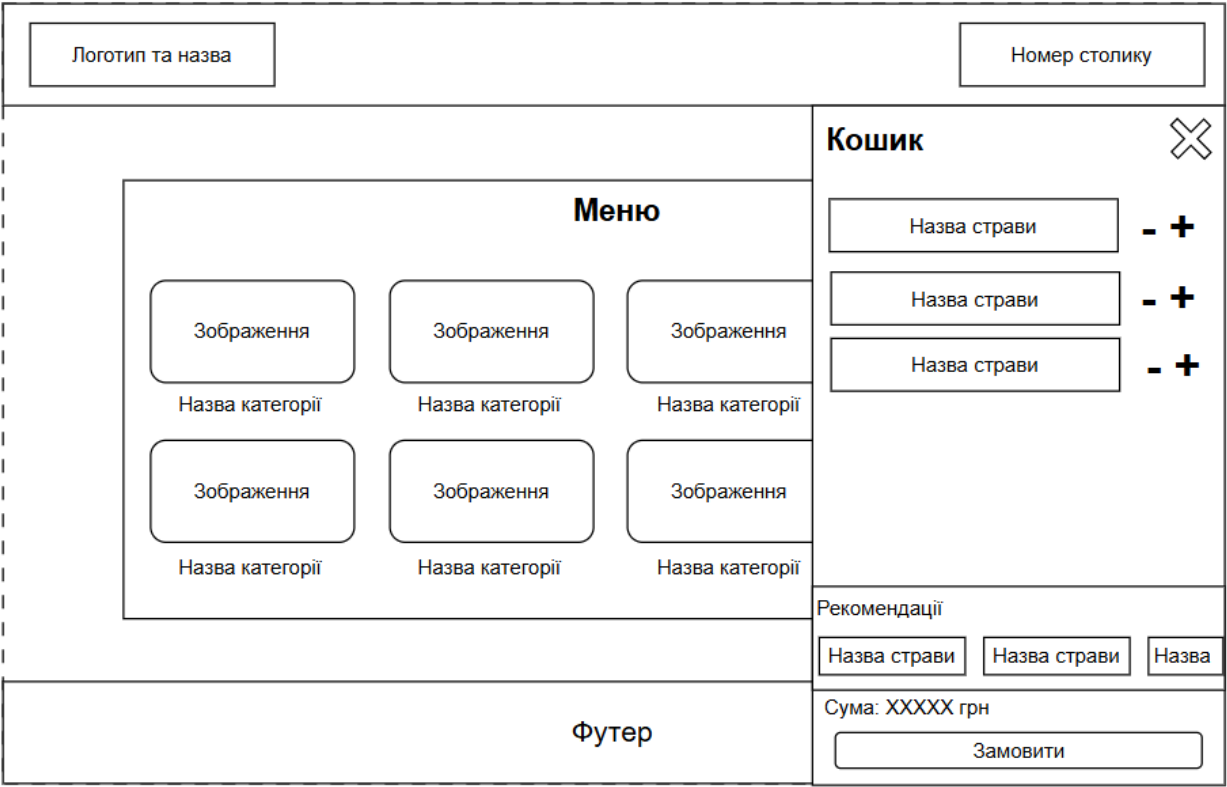


Рисунок 3.8 – Макет головної сторінки з картками категорій та кошиком

Кожна картка має власний ідентифікатор, який зчитується JavaScript скриптом при натисканні кнопки «Додати». У кошику передбачено блок інтелектуальних пропозицій: під час відкриття модального вікна виконується асинхронний запит до серверного методу `get_sell_recommendations`, що повертає дані для формування секції рекомендацій з cross-selling. HTML-структури цього блоку генерується динамічно на основі отриманого JSON-об'єкта.

3.4.3 Реалізація інтерфейсу дашборду в панелі адміністратора

Інтерфейс адміністратора призначений для керування меню та перегляду розрахованих показників ефективності асортименту. Основна складність полягала у візуалізації результатів роботи модулів ABC-класифікації та матриці Касавани-Сміта.

Для побудови графіків використано бібліотеку Chart.js. Серверний модуль обробляє дані за допомогою Pandas і передає у шаблон Django координати та мітки. JavaScript-скрипт ініціалізує об'єкт Chart, який відображає інтерактивну графіку на елементі <canvas>.

На рисунку 3.9 зображено макет панелі моніторингу з візуалізацією матриці класифікації страв. Кожна точка на графіку є інтерактивною: при наведенні курсору відображаються програмно розраховані значення популярності та маржинального прибутку для конкретного об'єкта Dish.

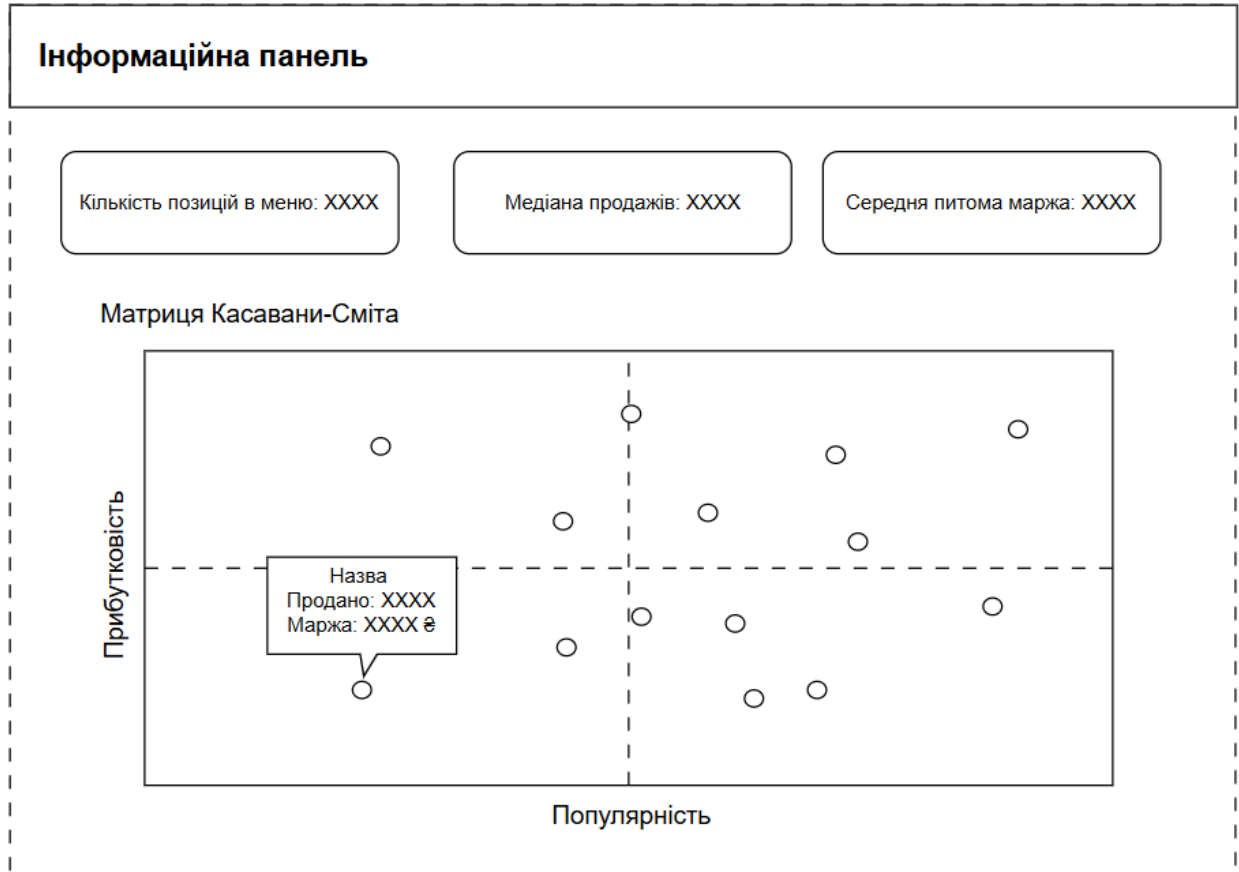


Рисунок 3.9 – Макет сторінки дашборду в панелі адміністратора

3.5 Висновки за розділом 3

Розроблено програмне забезпечення для автоматизації обслуговування клієнтів через QR-меню та алгоритмічної оцінки показників ефективності асортименту страв.

Запропоновано ієрархічну структуру програмного проєкту на базі фреймворку Django. Визначено, що основними програмними компонентами системи є модулі керування каталогом страв, обробки стану кошика замовлень та сервіс розрахункових алгоритмів.

Побудовано схеми потоків даних та взаємодії коду з базою через Django ORM. Вони показують роботу основних компонентів архітектури та демонструють, як кожен елемент бере участь у процесі обробки клієнтських запитів і відображення інформації про меню.

Розроблено структуру реляційної бази даних, яка відображає зв'язки між основними елементами предметної області ресторанного господарства. Основними сутностями бази даних обрано такі: Категорія (Category), Страва (Dish), Столик (Table), Замовлення (Order) та Позиція замовлення (OrderItem). Побудована ER-діаграма бази даних відображає логічні взаємозв'язки між основними таблицями та сутностями, що формують ядро транзакційної системи. На схемі відображено, як окремі елементи пов'язуються між собою для забезпечення повноти, точності та цілісності інформації, необхідної для роботи математичних алгоритмів.

У програмі реалізовано модулі для оцінки ефективності. Побудовано алгоритм сегментації методів моніторингу ABC-аналізу та матриці Касавани-Сміта.

Спроектовано та реалізовано інтерфейси взаємодії користувача з розробленим ПЗ. Створено макети головної сторінки з картками категорій і кошиком для гостя, а також макет сторінки інтерактивного дашборду в панелі адміністратора для візуалізації результатів алгоритмічної обробки даних.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення й умови застосування програми

Розроблювана програмна система «Scan & Savor» створена для автоматизації процесу прийому замовлень у закладах харчування за допомогою QR-меню та для алгоритмічної оцінки ефективності асортименту. Система забезпечує клієнтам безконтактний перегляд каталогу й формування кошика, а адміністрації надає дані щодо прибутковості та популярності кожної позиції завдяки вбудованим математичним алгоритмам. Таким чином, комплекс поєднує функції обслуговування гостей і засоби для меню-менеджменту.

Для стабільної роботи серверної частини визначено технічні вимоги до апаратного та програмного забезпечення. Оскільки ядром системи є фреймворк Django та реляційна база PostgreSQL, оптимальним середовищем розгортання є сервери під управлінням Linux. Мінімальна конфігурація включає двоядерний процесор із частотою від 2.0 ГГц, не менше 2 ГБ оперативної пам'яті для коректної роботи бібліотеки Pandas при обробці масивів даних та щонайменше 10 ГБ дискового простору. Програмне середовище повинно містити інтерпретатор Python версії 3.10 або вище.

Клієнтська частина не потребує потужних ресурсів, адже основна обробка даних виконується на сервері. Для взаємодії з гостьовим інтерфейсом достатньо смартфона на iOS чи Android із камерою для сканування QR-кодів. Обов'язковою умовою є доступ до Інтернету та сучасний веб-браузер з підтримкою JavaScript і HTML5. Адміністративна панель оптимізована для роботи на десктопах і планшетах із роздільною здатністю від 1024 пікселів по ширині, що забезпечує зручність перегляду даних та меню-менеджменту.

4.2 Методика та результати тестування програмного забезпечення

Тестування є важливою частиною життєвого циклу розробки, адже воно гарантує надійність системи та відповідність функціональним вимогам. Для перевірки працездатності «Scan & Savor» застосовано багаторівневий підхід: модульне, інтеграційне та системне тестування інтерфейсів.

На першому етапі було виконано модульне тестування алгоритмічних компонентів системи за допомогою вбудованого в Django фреймворку Test-Case. Основна увага приділялася перевірці коректності роботи математичних функцій у сервісному шарі. Було розроблено набір тестових сценаріїв для перевірки функції розрахунку медіанних значень, а також алгоритмів розподілу страв за категоріями. Тести підтвердили, що при подачі масивів даних зі статистичними викидами програмний модуль коректно обчислює медіану та безпомилково присвоює об'єктам відповідні категорії.

Інтеграційне тестування було спрямовано на перевірку коректності взаємодії між базою даних PostgreSQL, ORM та обчислювальним ядром на базі бібліотеки Pandas. На створених тестових замовленнях перевірялася транзакційна цілісність: збереження актуальної ціни страви у момент покупки та правильність передачі цих даних до агрегаційних функцій. Результати показали відсутність втрат або викривлень даних під час їх конвертації з QuerySet у структури DataFrame.

Завершальним етапом стало тестування користувацького інтерфейсу на кросбраузерність та адаптивність. Оскільки гостьова частина системи орієнтована на мобільні пристрої, її роботу було перевірено за допомогою інструментів розробника у браузерах Google Chrome, Opera та Microsoft Edge з емуляцією різних розмірів екранів. Використання сіткової системи Bootstrap 5 забезпечує коректне масштабування карток страв та модального вікна кошика. Асинхронні запити JavaScript відпрацьовують без затримок, забезпечуючи плавну взаємодію з користувачем.

4.3 Експериментальне дослідження роботи розрахункових алгоритмів

Для перевірки ефективності методів обробки даних проведено експеримент на згенерованому масиві тестових даних. У базу було завантажено інформацію про 30 страв різних категорій та створено 1500 транзакцій, що моделюють місячну роботу закладу.

Перший етап експерименту полягав у перевірці роботи модуля сегментації асортименту за методом ABC. Система успішно обробила завантажений масив даних, згрупувавши їх за критерієм сумарного доходу. Алгоритм коректно розрахував кумулятивний відсоток та виділив групу страв, які сумарно генерують 80% виручки. Отримані результати були відображені у таблиці аналізу страв в панелі адміністратора, що підтвердило здатність системи автоматично виявляти найбільш пріоритетні страви в меню (рис. 4.1).

НАЗВА СТРАВИ	ПРОДАНО (ШТ)	МАРЖА (€)	ЧАСТКА ОБОРОТУ	ABC КЛАС	КАТЕГОРІЯ (КАСАВАНА-СМІТ)	ПЛАН ДІЙ (ACTION PLAN)
Різотто з білими грибами	36	224,01 €	9,34%	Клас A	Зірка (Star)	Зберегти якість. Рекламувати.
Салат Цезар з куркою	36	204,99 €	8,02%	Клас A	Зірка (Star)	Зберегти якість. Рекламувати.
Піца Чотири сири	29	219,42 €	7,4%	Клас A	Зірка (Star)	Зберегти якість. Рекламувати.
Піца Вегетаріана	35	193,62 €	7,37%	Клас A	Зірка (Star)	Зберегти якість. Рекламувати.
Салат з прошуто та грушею	31	197,15 €	7,29%	Клас A	Зірка (Star)	Зберегти якість. Рекламувати.
Паста Карбонара	31	188,94 €	6,66%	Клас A	Зірка (Star)	Зберегти якість. Рекламувати.
Піца Прощуто е Фунгі	25	217,13 €	5,98%	Клас A	Загадка (Puzzle)	Високий потенціал. Додати в Рекомендації.
Паста Амарічіана	25	201,80 €	5,47%	Клас A	Загадка (Puzzle)	Високий потенціал. Додати в Рекомендації.
Піца Мисливська гостра	22	213,42 €	5,08%	Клас A	Загадка (Puzzle)	Високий потенціал. Додати в Рекомендації.
Салат Грецький	29	148,41 €	4,94%	Клас A	Конячка (Plow Horse)	Обережно підняти ціну або зменшити порцію.
Трамісу	30	130,60 €	4,38%	Клас A	Конячка (Plow Horse)	Обережно підняти ціну або зменшити порцію.
Паста Болоньезе	21	173,96 €	4,34%	Клас A	Загадка (Puzzle)	Високий потенціал. Додати в Рекомендації.

Рисунок 4.1 – Таблиця алгоритмічної оцінки страв

Другий етап був присвячений дослідженню модифікованої моделі Касавани-Сміта. Метою експерименту було доведення переваги використання медіани над середнім арифметичним. У тестовий набір свідомо було додано

одну позицію з аномально високою кількістю продажів. Розрахунок за класичною формулою через середнє значення призвів до хибного зміщення порогу популярності, через що більшість стабільних страв отримали статус об'єктів з низькою популярністю. Натомість програмний модуль «Scan & Savor», який використовує медіану, успішно нівелював цей статистичний викид, забезпечивши рівномірний та об'єктивний розподіл об'єктів по чотирьох квадрантах матриці.

Третій етап дослідження перевіряв працездатність модуля пошуку асоціативних правил. У кошик користувача тестового середовища було додано об'єкт «Капучино». База даних виконала обчислення частоти спільної зустрічальності, і алгоритм миттєво повернув JSON-відповідь із рекомендаціями, де більшість страв були з категорії «Десерти» (рис. 4.2).

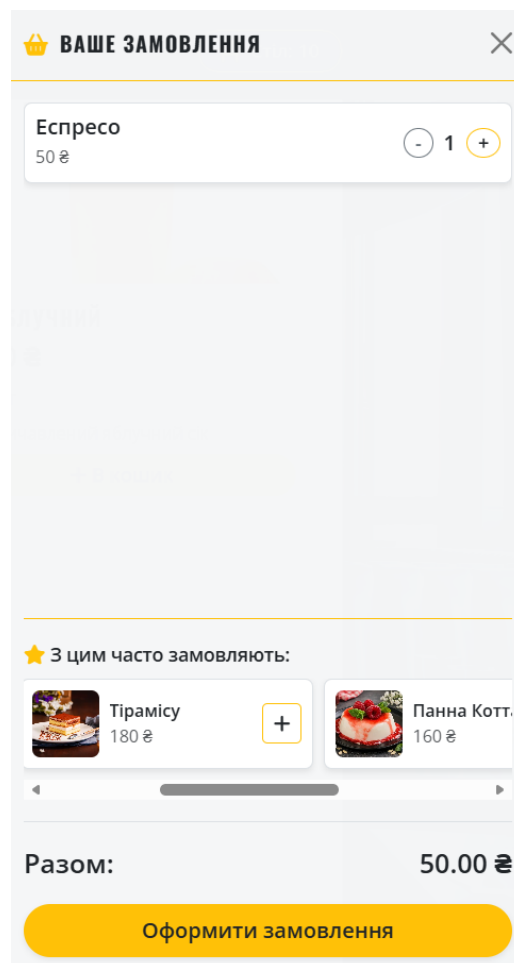


Рисунок 4.2 – Приклад роботи модуля пошуку асоціативних правил

Це підтверджує точність та швидкість роботи SQL-агрегацій, реалізованих за допомогою методів Django ORM.

4.4 Інструкція по експлуатації програми

4.4.1 Звернення до програми

Доступ до функціональних можливостей розроблюваного ПЗ здійснюється через веб-браузер. Для клієнта звернення до програми відбувається шляхом зчитування камерою мобільного пристрою унікального QR-коду, розміщеного на столику закладу. Ця дія активує перехід за URL-адресою, яка містить унікальний ідентифікатор столика, що дозволяє системі програмно ідентифікувати джерело запиту та автоматично прив'язати майбутнє замовлення до конкретного об'єкта в БД.

Для адміністративного персоналу звернення до системи виконується через стандартний інтерфейс авторизації за адресою керуючого модуля (/admin). Доступ надається після успішної перевірки даних автентифікації на рівні серверної логіки Django. Після авторизації програмний засіб надає доступ до розділів керування асортиментом та перегляду результатів алгоритмічної обробки показників.

4.4.2. Вхідні й вихідні дані

Вхідними даними для клієнтської частини є вибір користувачем страв та їхня кількість, що накопичуються в локальному сховищі браузера. На рівні адміністративного модуля вхідними даними є параметри карток страв: назва, категорія, ціна продажу та собівартість. Також до вхідних даних відносяться історичні масиви транзакцій, що вилучаються з бази даних PostgreSQL для подальшої математичної обробки.

Також вихідними даними системи є динамічно сформовані HTML-сторінки з актуальним меню, а також об'єкти замовлень, що передаються на сервер у форматі JSON. Результатом роботи розрахункових модулів є вихідні масиви даних, що містять розподіл страв за категоріями прибутковості та популярності. Ці дані візуалізуються через інтерфейсні компоненти у вигляді інтерактивних графіків та таблиць, що відображають програмно обчислені показники ефективності кожної позиції асортименту.

4.4.3 Повідомлення

Взаємодія програми з користувачем супроводжується системою програмних сповіщень, що відображають стан обробки запитів. При успішному виконанні операцій, таких як відправка замовлення на сервер, система видає інформаційні повідомлення для підтвердження дії. У випадку виникнення помилок система виводить діалогові вікна з описом проблеми. Адміністративна панель додатково містить повідомлення валідації на рівні Django, які сигналізують про некоректне введення числових параметрів ціни чи собівартості при редагуванні бази даних.

4.4.4 Виконання програмного забезпечення

Процес виконання програми розділений на два основні функціональні цикли. Перший цикл охоплює взаємодію гостя з меню: від моменту ініціалізації сесії через QR-код до завершення оформлення замовлення. На кожному етапі JavaScript-сценарії обробляють зміну стану інтерфейсу, виконують асинхронні запити для отримання рекомендацій та передають фінальний масив даних на сервер для збереження Django ORM.

Другий цикл виконання стосується роботи адміністративного модуля. При завантаженні сторінки показників ефективності серверна частина ініціює виконання обчислювальних алгоритмів. Бібліотека Pandas виконує читання

транзакцій, проводить векторні операції з даними та повертає структуровані результати для відображення за допомогою графіків. Цей процес відбувається автоматично при кожному зверненні до відповідного розділу, забезпечуючи актуальність відображених результатів на основі останніх записів у базі даних.

4.5 Висновки за розділом 4

У четвертому розділі проведено дослідження роботи та надійності програмного комплексу «Scan & Savor». Визначено його призначення та умови використання, описано вимоги до серверного обладнання й клієнтських пристроїв, що забезпечують стабільну роботу системи в закладах харчування. В ході тестування визначено, що модулі розроблюваного ПЗ, зокрема модуль обробки замовлень та модуль обчислювальних алгоритмів, є працездатними.

Експеримент на тестовому масиві з 1500 транзакцій показав перевагу використаних методів обробки. Визначено, що застосування медіани у розрахунках методу Касавани-Сміта робить систему стійкою до статистичних викидів і забезпечує об'єктивність оцінки меню. Також підтверджено роботу алгоритму формування асоціативних зв'язків між стравами, який використовується для створення рекомендацій.

Отримані результати тестування підтвердили відповідність продукту поставленим завданням і вимогам, що дозволяє рекомендувати його для практичного використання в закладах індустрії HoReCa.

ВИСНОВКИ

У ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес розроблення програмного забезпечення для автоматизації сервісних операцій та програмної реалізації обчислювальних методів оцінки характеристик асортименту страв.

Визначено, що сучасне програмне забезпечення для сегмента HoReCa це спеціалізовані системи, які не лише забезпечують безконтактний доступ до цифрових каталогів через QR-коди, але й збирають та обробляють великі об'єми даних про замовлення. За допомогою таких програмних сервісів оптимізується процес прийому замовлень: як показав аналіз, впровадження безконтактного QR-меню дозволяє скоротити загальний час взаємодії з клієнтом під час одного циклу обслуговування на 54% (з 27 до 12,5 хвилин). Така система зменшує навантаження на обслуговуючий персонал, прискорює обіг столиків та зводить до мінімуму вплив людського фактора при формуванні чека. Такі програмні інструменти дозволяють чітко структурувати та використовувати корисну інформацію, що дозволяє керівництву закладів приймати обґрунтовані рішення щодо управління асортиментом на основі математичних розрахунків.

За результатами проведеного аналізу предметної області визначено, що існує досить багато програмних засобів для автоматизації ресторанів. Але більшість існуючих рішень зосереджені тільки на веденні складського обліку або візуальному відображенні меню, не надаючи адміністраторам закладу вбудовані інструменти для глибокої обробки та оцінки даних про продажі. Також відсутність автоматизованих модулів оцінки страв та алгоритмів пошуку асоціативних правил обмежує можливості автоматизованого обчислення параметрів рентабельності в реальному часі та знижує загальну ефективність роботи закладу. У результаті виникає потреба в системах, які поєднують клієнтський

функціонал та обчислювальне ядро. Тому актуальною є розробка комплексного програмного забезпечення, здатного виконувати математичну оцінку рентабельності меню.

Сформульовано функціональні вимоги до програмного забезпечення, які включають адаптивний гостьовий інтерфейс з підтримкою локального кошика та адміністративну панель з вбудованими інструментами моніторингу і генерації QR-кодів.

Для реалізації серверної частини програмного забезпечення було обрано мову програмування Python та фреймворк Django. Django забезпечує високу швидкість розробки, надійність та можливість легко інтегрувати складні розрахункові модулі. Використання системи керування базами даних PostgreSQL забезпечило цілісність збереження даних та надало інструментарій для агрегації інформації. Для виконання складних математичних операцій, векторних обчислень та статистичної обробки масивів замовлень використано бібліотеку Pandas.

Програмне забезпечення побудовано на основі архітектурного шаблону Model-View-Template. Визначено, що основними сутностями програмного забезпечення є категорія, страва, столик та замовлення. Побудовано схеми функціонування програмного забезпечення, які відображають взаємодію між контролерами, базою даних та обчислювальним ядром у загальному процесі обробки запитів.

Розроблено базу даних програмного забезпечення, яка відображає зв'язки між елементами у предметній області. Структура бази даних спроектована з урахуванням вимог нормалізації, що забезпечує швидкий доступ до історії транзакцій для роботи математичних алгоритмів.

У роботі реалізовано модифікований алгоритм класифікації об'єктів за матрицею Касавани-Сміта. При розрахунку порогів популярності використано значення медіани замість середнього арифметичного, що забезпечує стійкість системи до статистичних викидів та більш точний розподіл страв за квадрантами.

Був спроектований та реалізований інтерфейс взаємодії користувача з програмним забезпеченням. Створено адаптивний клієнтський інтерфейс з використанням Bootstrap 5 та інтерактивний адміністративний дашборд із використанням бібліотеки Chart.js для візуалізації результатів обробки даних.

Виконано тестування розробленого програмного забезпечення. Результати експериментального дослідження, проведеного на згенерованому масиві з 1500 транзакцій для 30 тестових позицій меню, підтвердили високу обчислювальну продуктивність системи. Доведено, що програмна реалізація ABC-аналізу успішно агрегує дані та виділяє пріоритетну групу страв, яка генерує 80% усього доходу. Оптимізація запитів до бази даних дозволила алгоритму пошуку асоціативних правил обробляти історичні транзакції та формувати персоналізовані рекомендації для кошика.

Подальший розвиток програмної системи «Scan & Savor» може включати розширення його можливостей. Серед основних напрямів – створення REST API для інтеграції з комерційними POS-системами з метою синхронізації складських залишків; використання складніших моделей машинного навчання замість базового алгоритму асоціативних правил для більш персоналізованих рекомендацій; розробка модуля прогнозування попиту на інгредієнти; а також доповнення клієнтської частини візуальними елементами, наприклад створенням списку найпопулярніших страв меню.

В ході роботи створено програмне забезпечення «Scan & Savor», яке автоматизує процес обслуговування клієнтів у закладах харчування та надає керівництву інструменти для обробки даних продажів. Отримані результати можуть бути використані підприємствами ресторанного господарства для оптимізації асортименту страв, а розроблені програмні рішення – у майбутніх проєктах високонавантажених інформаційних систем.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Wong C. C., Chong L. Y., Chong S. C. QR Food Ordering System with Data Analytics. *Journal of Informatics and Web Engineering*. 2023. Vol. 2, № 2. P. 249–272. URL: <https://journals.mmupress.com/index.php/jiwe/article/view/650/414> (дата звернення: 27.04.2026).
2. Thummar J. B., Upadhyay B. R., Satote B. R. DineWise RMS: A QR Code-Based Restaurant Management. *RSP Science Hub*. URL: <https://rspsciencehub.com/index.php/journal/article/view/983/827> (дата звернення: 27.04.2026).
3. Іваненко В. Інжиніринг-меню як дієвий аналітичний інструмент підвищення ефективності діяльності ресторану. *Економіка, управління та інновації*. URL: <https://eui.zu.edu.ua/article/view/346879/333853> (дата звернення: 27.04.2026).
4. Fahreza M. Z., Jaman J. H., Maulana I. Penerapan Market Basket Analysis Untuk Rekomendasi Paket Menu Menggunakan Algoritma FP-Growth (Studi Kasus : Kafe Shans Juice). *Jurnal Informatika dan Teknik Elektro Terapan*. 2025. Vol. 13, № 3S1. URL: <https://journal.eng.unila.ac.id/index.php/jitet/article/view/7527/3565> (дата звернення: 27.04.2026).
5. Widodo A. E., Wati F. F., Widayanto A. Caffé Menu Ordering Information System Using QR-Code. *J-INTECH*. URL: <https://jurnal.ubhinus.ac.id/index.php/J-INTECH/article/view/865/585> (дата звернення: 27.04.2026).
6. Cross-Selling and Upselling: The Ultimate Guide. *HubSpot Blog*. URL: <https://blog.hubspot.com/sales/cross-selling> (дата звернення: 27.04.2026).
7. How to increase average check in restaurant. *Upmenu*. URL: <https://www.upmenu.com/blog/how-to-increase-average-check-in-restaurant/> (дата звернення: 27.04.2026).

8. How to increase average check in restaurant. *GloriaFood*. URL: <https://www.gloriafood.com/how-to-increase-average-check-in-restaurant> (дата звернення: 27.04.2026).
9. Examining a menu on the basis of the Kasavana - Smith model in a Hungarian restaurant / Z. Ivancsóné Horváth et al. *Deturope*. 2022. Vol. 14, № 1. P. 111–127. URL: <https://www.deturope.eu/pdfs/det/2022/01/06.pdf> (дата звернення: 27.04.2026).
10. Олійник О. В., Шестакова А. В., Ярмолюк Д. І. Напрями цифровізації ресторанного бізнесу. *Економіка управління та адміністрування*. 2023. URL: <https://www.researchgate.net/publication/369893798> (дата звернення: 27.04.2026).
11. Лизанець А., Русин О. Цифрові тренди розвитку ресторанного бізнесу. *Збірник тез доповідей Всеукраїнської науково-практичної інтернет-конференції* (27–28 вересня 2024 року). 2024. URL: http://dspace.s.msu.edu.ua:8080/bitstream/123456789/12493/1/Digital_trends_in_the_development_of_the_restaurant_business.pdf (дата звернення: 27.04.2026).
12. How Digital is Changing the Rules of Menu Engineering. *Simplot Food Blog*. 2023. URL: <https://www.simplotfood.com/blog/how-digital-is-changing-the-rules-of-menu-engineering> (дата звернення: 27.04.2026).
13. Маркетингові та організаційні механізми повоєнного розвитку галузі гостинності та туризму України. *Тези доповідей III Міжнародної науково-практичної конференції* (12 листопада 2025 року, м. Харків). Частина 2. 2025. URL: <https://repository.kpi.kharkov.ua/server/api/core/bitstreams/cc7f9d34-f403-438e-812d-e046ce71a077/content?page=263> (дата звернення: 27.04.2026).
14. Полотай Б. Я., Жмур-Клименко Б. В. Ресторанний бізнес під час війни. *Вісник Львівського торговельно-економічного університету. Гуманітарні науки. Серія: Туризм та готельно-ресторанна справа*. 2022. Вип. 29. С. 132–137. URL: <https://journals-lute.lviv.ua/index.php/tourism/article/view/1250/1177> (дата звернення: 27.04.2026).

15. Насонова О. Ресторанний бізнес у кризу. *Економічна правда*. 2021. 17 лют. URL: <https://www.epravda.com.ua/columns/2021/02/17/671122/> (дата звернення: 27.04.2026).
16. Bojnec Š. Ascertaining Restaurant Financial Sustainability by Analyzing Menu Performance. *Tourism and Hospitality*. 2025. Vol. 6, № 1. Art. 47. URL: <https://www.mdpi.com/2673-5768/6/1/47> (дата звернення: 27.04.2026).
17. Menu Engineering: How to Increase Restaurant Profits. *Lightspeed Blog*. 2022. URL: <https://www.lightspeedhq.com/blog/menu-engineering> (дата звернення: 27.04.2026).
18. Menu Engineering as an Effective Analytical Tool for Enhancing Restaurant Performance. *Eastern European Business and Management Journal*. 2019. № 18. С. 49–56. URL: https://www.easterneurope-bbm.in.ua/journal/18_2019/49.pdf (дата звернення: 27.04.2026).
19. Тренди ресторанного бізнесу 2025. *Poster Blog*. 2025. URL: <https://joinposter.com/ua/blog/marketing/restaurant-trends> (дата звернення: 27.04.2026).
20. Poster – хмарна каса для кафе, ресторанів та магазинів. URL: <https://joinposter.com/> (дата звернення: 27.04.2026).
21. ChoiceQR. URL: <https://choiceqr.com/> (дата звернення: 27.04.2026).
22. Me-Menu. URL: <https://me-menu.com/> (дата звернення: 27.04.2026).
23. Kasavana M., Smith D., Schmidgall R. Menu Engineering. A Practical Guide to Menu Analysis. Okemos : Hospitality Publications Inc., 1990. 167 p.
24. Menu Engineering on Main Course to Increase Sales / N. K. Saraswati et al. *International Journal of Glocal Tourism*. 2021. Vol. 1, № 1. P. 51–60. URL: <https://ejournal.sidyanusa.org/index.php/injogt/article/view/17/14> (дата звернення: 27.04.2026).
25. Fedosova K. Development of an effective restaurant menu. Research and recommendations. *Technology Audit and Production Reserves*. 2022. Vol. 1, № 4 (63). P. 32–35. URL: <https://journals.uran.ua/tarp/article/view/253040/250711> (дата звернення: 27.04.2026).

26. Kiiski R. Haaga-Helian opinnäytetyöohje. URL: https://www.theseus.fi/bitstream/handle/10024/796224/Kiiski_Riina.pdf (дата звернення: 27.04.2026).
27. Agrawal R., Srikant R. Fast algorithms for mining association rules. *Proc. 20th int. conf. very large data bases, VLDB*. 1994. Vol. 1215. P. 487–499. URL: <https://www.vldb.org/conf/1994/P487.PDF> (дата звернення: 27.04.2026).
28. Wati T., Sahvitri V., Wardani K. R. N. Implementasi Metode Market Basket Analysis Pada Sistem Penjualan Dan Pemasaran Pada Butik Nazwa Fashion. *Jurnal Pengembangan Sistem Informasi dan Informatika*. 2020. Vol. 1, No. 3. С. 184–196. URL: <https://journal.jis-institute.org/index.php/jpsii/article/view/181> (дата звернення: 27.04.2026).
29. Flask Documentation. *Pallets Projects*. URL: <https://flask.palletsprojects.com/en/stable/> (дата звернення: 27.04.2026).
30. FastAPI Documentation. *Tiangolo*. URL: <https://fastapi.tiangolo.com> (дата звернення: 27.04.2026).
31. Django documentation. *Django Project*. URL: <https://docs.djangoproject.com/en/5.0/> (дата звернення: 27.04.2026).
32. MongoDB Documentation. *MongoDB Inc.* URL: <https://www.mongodb.com/docs/> (дата звернення: 27.04.2026).
33. MySQL Documentation. *Oracle Corporation*. URL: <https://dev.mysql.com/doc/> (дата звернення: 27.04.2026).
34. PostgreSQL: The World's Most Advanced Open Source Relational Database. *PostgreSQL Global Development Group*. URL: <https://www.postgresql.org/docs/> (дата звернення: 27.04.2026).
35. McKinney W. Data Structures for Statistical Computing in Python. *Proceedings of the 9th Python in Science Conference*. 2010. P. 51–56. URL: <https://pub.curvenote.com/01908378-3686-7168-a380-d82bbf21c799/public/mckinney-57fc0d4e8a08cd7f26a4b8bf468a71f4.pdf> (дата звернення: 27.04.2026).

36. Pandas Documentation. *Pandas Project*. URL: <https://pandas.pydata.org/docs> (дата звернення: 27.04.2026).

37. Bootstrap Documentation. Getting Started. *Bootstrap Official Site*. URL: <https://getbootstrap.com/docs/5.0/getting-started/introduction/> (дата звернення: 27.04.2026).

38. JavaScript Documentation. *DevDocs*. URL: <https://devdocs.io/javascript> (дата звернення: 27.04.2026).

39. Chart.js Documentation. *Chart.js Project*. URL: <https://www.chartjs.org/docs/latest/> (дата звернення: 27.04.2026).

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення системи QR-меню та алгоритмічної оцінки асортименту «Scan & Savor» може використовуватися в галузі ресторанного господарства для автоматизації процесів обслуговування клієнтів закладів харчування та виконання алгоритмічної оцінки рентабельності меню на основі реальних даних продажів.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу бакалавра на тему «Розроблення та дослідження програмного забезпечення автоматизації закладів харчування», затверджене наказом Національного університету «Запорізька політехніка» № 139 від 07 квітня 2026 р.

A.2 Призначення розробки

Програмний продукт призначений для прийому замовлень через QR-коди та математичної обробки показників ефективності асортименту страв.

A.3 Основні вимоги до програми, що розробляється

A.3.1 Вимоги до функціональних характеристик

Програмне забезпечення для автоматизації закладів харчування забезпечує виконання таких функцій:

- підтримка можливості сканування QR-коду та програмної ідентифікації столика закладу;
- можливість відображення переліку категорій та страв, інформація про які зберігається у базі даних застосунку;

- можливість відображення інформації про страву, зокрема, її опис, ціну, вагу та зображення;
- підтримка можливості формування кошика замовлень з динамічним виведенням розрахованих cross-selling рекомендацій;
- можливість обчислення маржинального доходу, класифікації страв за матрицею Касавани-Сміта та проведення алгоритмічної сегментації ABC-аналізу;
- підтримка відображення панелі адміністратора з інтерактивними графіками для візуалізації результатів обробки транзакційних даних.

Вхідними даними є запити користувачів, конфігураційні параметри страв та історичні транзакції. Вихідними даними є сформовані замовлення, HTML-сторінки меню та JSON-об'єкти з розрахованими показниками рентабельності.

А.3.2 Вимоги до надійності

Програмне забезпечення повинно забезпечити надійне функціонування, локальне збереження стану кошика користувача та транзакційну цілісність даних при збереженні замовлень у базі даних.

А.3.3 Умови експлуатації

Для експлуатації програмного забезпечення гостями необхідна наявність мобільного пристрою з фотокамерою, а для повноцінної роботи керівництва з панеллю моніторингу – наявність персонального комп'ютера.

А.3.4 Вимоги до складу та параметрів технічний засобів

Для роботи застосунку потрібні відповідні технічні ресурси. Сервер має бути оснащений процесором із тактовою частотою не нижче 2,0 ГГц. Рекомендується застосовувати багатоядерні процесори для стабільної роботи бази даних. Обсяг оперативної пам'яті повинен становити щонайменше 2 ГБ для базових конфігурацій або 4 ГБ і більше для коректної роботи бібліотек обробки даних при активному використанні системи. Для зберігання даних необхідний SSD-накопичувач об'ємом від 20 ГБ.

А.3.5 Вимоги до інформаційної та програмної сумісності

Програмне середовище сервера має включати сучасну операційну систему Linux. Для функціонування серверної частини необхідно встановити інтерпретатор мови Python версії 3.10 або вище з використанням фреймворку Django 5.x. База даних повинна працювати під управлінням СКБД PostgreSQL версії 14 або вище.

Для обміну даними між клієнтом та сервером використовується архітектура REST та формат передачі даних JSON. Користувацький інтерфейс має бути сумісним із сучасними веб-браузерами з обов'язковою підтримкою технологій HTML5, CSS3 та JavaScript.

Для захисту інформації передача даних має здійснюватися через захищений протокол HTTPS, а серверні форми повинні бути захищені від CSRF та SQL-ін'єкцій.

А.3.6 Вимоги до маркування і пакування

Програмне забезпечення передається в електронному вигляді або записується на Flash-накопичувач. На пакуванні або в супровідному текстовому

файлі README.md повинна бути зазначена назва програми – «Програмне забезпечення системи QR-меню та алгоритмічної оцінки асортименту», а також версія продукту.

A.4 Техніко-економічні показники

Орієнтована економічна ефективність від впровадження розробки досягається за рахунок скорочення витрат на друк паперових меню, зменшення часу обслуговування одного столика на 10-15 хвилин та оптимізації витрат на закупівлю інгредієнтів шляхом вилучення збиткових страв.

Економічною перевагою розробки порівняно з комерційними аналогами є відсутність щомісячної абонентської плати за використання базового функціонала та наявність унікального вбудованого математичного модуля Касавани-Сміта, який у комерційних системах вимагає інтеграції дорогих сторонніх аналітичних сервісів.

ДОДАТОК Б
Фрагмент тексту програми

Б.1 Текст файлу urls.py

```

from django.urls import path
from . import views, views_api, views_admin

app_name = 'menu'

urlpatterns = [
    # API
    path('api/create-order/', views_api.create_order, name='create_order'),

    # Сторінки адміністратора
    path('admin-dashboard/monitoring-dashboard/', views_admin.monitoring_dashboard, name='monitoring_dashboard'),
    path('download-qr/<int:table_id>/', views.download_qr, name='download_qr'),

    # Клієнтські сторінки
    path('', views.index, name='index'),
    path('scan/<uuid:qr_uuid>/', views.scan_qr, name='scan_qr'),
    path('category/<slug:slug>/', views.category_detail, name='category_detail'),
    path('about/', views.about, name='about'),
    path('api/recommendations/', views_api.get_recommendations, name='get_recommendations'),
]

```

Б.2 Текст файлу views.py

```

from django.shortcuts import render, get_object_or_404, redirect
from .models import Category, Table
from django.conf import settings
from menu_app.services.qr_code_generator import generate_qr_with_logo
from django.http import HttpResponseRedirect
from django.contrib.admin.views.decorators import staff_member_required
import os

```

```

@staff_member_required
def download_qr(request, table_id):
    table = get_object_or_404(Table, id=table_id)

    # url для QR-коду
    base_url = request.build_absolute_uri('/')
    url = f"{base_url}scan/{table.qr_uuid}/"

    logo_path = os.path.join(settings.BASE_DIR, 'static', 'logo.webp')
    qr_img = generate_qr_with_logo(url, logo_path)

    response = HttpResponse(content_type="image/png")

    # Завантаження зображення QR-коду
    response['Content-Disposition'] = f'attachment; filename="qr_table_{table.number}.png"'
    qr_img.save(response, "PNG")

    return response

def scan_qr(request, qr_uuid):
    # Пошук столу за UUID
    table = get_object_or_404(Table, qr_uuid=qr_uuid)

    # Зберігання ID та номеру столу в сесію браузера клієнта
    request.session['table_id'] = table.id
    request.session['table_number'] = table.number

    # Встановлення терміну дії сесії на 2 години (7200 секунд)
    request.session.set_expiry(7200)

    # Перенаправлення на головну сторінку меню
    return redirect('menu:index')

def index(request):
    categories = Category.objects.all().order_by('order', 'name')
    return render(request, 'index.html', {'categories': categories})

```

```
def category_detail(request, slug):
    category = get_object_or_404(Category, slug=slug) # Отримання
    категорії за слагом
    dishes = category.dishes.filter(is_active=True) # Всі активні страви
    категорії
    return render(request, 'category_detail.html', {'category': cate-
    gory, 'dishes': dishes})

def about(request):
    return render(request, 'about.html')
```

Б.3 Текст файлу tests.py

```
import json
from django.test import TestCase, Client
from django.urls import reverse
from menu_app.models import Table, Category, Dish

class MenuAppTests(TestCase):
    def setUp(self):
        # Створення тестової БД
        self.client = Client()

        # Створення стілу
        self.table = Table.objects.create(number=1)

        # Створення категорії
        self.category = Category.objects.create(name="Основні страви",
        slug="osnovni")

        # Створення страви
        self.dish = Dish.objects.create(
            category=self.category,
            name="Борщ український",
            slug="borshch",
            price=150.00,
            cost_price=50.00,
            is_active=True
```

```

    )

def test_dish_creation(self):
    """
    Тест №1: Перевірка цілісності бізнес-моделей (ORM)
    Перевірка коректності розрахунку маржи та збереження даних
    """

    dish = Dish.objects.get(name="Борщ український")
    self.assertEqual(dish.price, 150.00)
    self.assertTrue(dish.is_active)
    self.assertEqual(str(dish), "Борщ український")

def test_api_empty_cart_validation(self):
    """
    Тест №2: Перевірка безпеки API
    Перевірка відповіді сервера на спробу створити замовлення з по-
    рожнім кошиком
    """

    # Імітація сесії клієнта (з відсканованим QR-кодом)
    session = self.client.session
    session['table_id'] = self.table.id
    session.save()

    # post-запит на створення замовлення з порожнім кошиком
    url = reverse('menu:create_order')
    response = self.client.post(
        url,
        data=json.dumps({'cart': {}}),
        content_type='application/json'
    )

    # Перевірка відповіді сервера (400 Bad Request)
    self.assertEqual(response.status_code, 400)

    # Перевірка повідомлення
    response_data = json.loads(response.content)
    self.assertEqual(response_data['status'], 'error')

```

```
self.assertEqual(response_data['message'], 'Кошик порожній')
```

Б.4 Текст файлу models.py

```
import uuid
from django.db import models

# Модель категорії страви
class Category(models.Model):
    name = models.CharField(max_length=100, verbose_name="Назва
категорії")
    slug = models.SlugField(max_length=120, unique=True, ver-
bose_name="Слаг (URL)")
    image = models.ImageField(upload_to='categories/', blank=True,
null=True, verbose_name="Фото категорії")
    order = models.PositiveIntegerField(default=0, ver-
bose_name="Порядок відображення")

    class Meta:
        verbose_name = "Категорія"
        verbose_name_plural = "Категорії"
        ordering = ['order']

    def __str__(self):
        return self.name

# Модель страви
class Dish(models.Model):
    category = models.ForeignKey(Category, related_name='dishes',
on_delete=models.CASCADE, verbose_name="Категорія")
    name = models.CharField(max_length=200, verbose_name="Назва
страви")
    slug = models.SlugField(max_length=220, unique=True, ver-
bose_name="Слаг (URL)")
    description = models.TextField(blank=True, verbose_name="Опис")

    price = models.DecimalField(max_digits=8, decimal_places=2, ver-
bose_name="Ціна продажу")
```

```

cost_price = models.DecimalField(max_digits=8, decimal_places=2,
verbose_name="Собівартість")

image = models.ImageField(upload_to='dishes/', blank=True,
null=True, verbose_name="Фото страви")

is_active = models.BooleanField(default=True, verbose_name="Активна
(в меню)")

weight = models.PositiveIntegerField(verbose_name="Вага (г)",
null=True, blank=True)

is_new = models.BooleanField(default=False, verbose_name="Новинка")

class Meta:
    verbose_name = "Страва"
    verbose_name_plural = "Страви"

def __str__(self):
    return self.name

# Модель столу в ресторані
class Table(models.Model):
    number = models.PositiveIntegerField(unique=True, ver-
bose_name="Номер столика")
    # Унікальний UUID
    qr_uuid = models.UUIDField(default=uuid.uuid4, editable=False,
unique=True)

    class Meta:
        verbose_name = "Столик"
        verbose_name_plural = "Столики"

    def __str__(self):
        return f"Стіл №{self.number}"

# Модель замовлення
class Order(models.Model):
    STATUS_CHOICES = (
        ('new', 'Нове'),
        ('cooking', 'Готується'),

```

```

        ('ready', 'Готово'),
        ('paid', 'Оплачено та закрито'),
    )

    table = models.ForeignKey(Table, related_name='orders', on_delete=models.SET_NULL, null=True, verbose_name="Столик")
    status = models.CharField(max_length=20, choices=STATUS_CHOICES, default='new', verbose_name="Статус")

    created_at = models.DateTimeField(auto_now_add=True, verbose_name="Час створення")

    class Meta:
        verbose_name = "Замовлення"
        verbose_name_plural = "Замовлення"

    def __str__(self):
        return f"Замовлення #{self.id} (Стіл {self.table.number if self.table else 'Видалено'})"

# Модель позиції замовлення
class OrderItem(models.Model):
    order = models.ForeignKey(Order, related_name='items', on_delete=models.CASCADE)
    dish = models.ForeignKey(Dish, on_delete=models.SET_NULL, null=True, verbose_name="Страва")
    quantity = models.PositiveIntegerField(default=1, verbose_name="Кількість")

    # Ціни на момент замовлення
    price_at_moment = models.DecimalField(max_digits=8, decimal_places=2, editable=False)
    cost_at_moment = models.DecimalField(max_digits=8, decimal_places=2, editable=False)

    class Meta:
        verbose_name = "Позиція замовлення"
        verbose_name_plural = "Позиції замовлень"

```

```

def save(self, *args, **kwargs):
    # Якщо нове, скопіювати актуальні ціни зі страви
    if not self.pk and self.dish:
        self.price_at_moment = self.dish.price
        self.cost_at_moment = self.dish.cost_price
    super().save(*args, **kwargs)

def __str__(self):
    return f"{self.quantity}x {self.dish.name if self.dish else
'Видалена страва'}"

class AnalyticsProxy(models.Model):
    # Модель для додавання посилання на сторінку моніторингу у адмін-
панелі

    class Meta:
        managed = False # Не створювати таблицю в БД
        verbose_name = "Information panel"
        verbose_name_plural = "Information panel"

```

Б.5 Текст файлу admin.py

```

import zipfile, io, uuid, os
from django.contrib import admin, messages
from django.http import HttpResponseRedirect, HttpResponse
from .models import Category, Dish, Table, Order, OrderItem
from django.urls import reverse
from django.utils.html import format_html
from .models import AnalyticsProxy
from django.conf import settings
from menu_app.services.qr_code_generator import generate_qr_with_logo

# Реєстрація моделей у панелі адміністратора для управління даними
@admin.register(Category)
class CategoryAdmin(admin.ModelAdmin):
    list_display = ('name', 'order')
    list_editable = ('order',)
    prepopulated_fields = {'slug': ('name',)}

```

```

@admin.register(Dish)
class DishAdmin(admin.ModelAdmin):
    list_display = ('name', 'category', 'price', 'cost_price', 'get_margin', 'weight', 'is_active', 'is_new')
    list_filter = ('category', 'is_active')
    search_fields = ('name', 'description')
    prepopulated_fields = {'slug': ('name',)}

    # Обчислювальний стовпець для маржі
    def get_margin(self, obj):
        return obj.price - obj.cost_price
    get_margin.short_description = 'Маржа (грн)'

@admin.action(description='Перегенерувати QR-коди')
def regenerate_qr_uuids(modeladmin, request, queryset):
    for table in queryset:
        table.qr_uuid = uuid.uuid4() # Генерація нового унікального ID
столика
        table.save()
    messages.success(request, f"Успішно оновлено QR-коди для {queryset.count()} столиків.")

@admin.action(description='Завантажити QR-коди архівом')
def download_qr_archive(modeladmin, request, queryset):
    buffer = io.BytesIO()
    with zipfile.ZipFile(buffer, 'w') as zip_file:
        base_url = request.build_absolute_uri('/')
        logo_path = os.path.join(settings.BASE_DIR, 'static', 'logo.webp')

        for table in queryset:
            url = f"{base_url}scan/{table.qr_uuid}/"
            img = generate_qr_with_logo(url, logo_path)

            img_bytes = io.BytesIO()
            img.save(img_bytes, format="PNG")
            img_bytes.seek(0)
            zip_file.writestr(f"qr_table_{table.number}.png",
img_bytes.read())

```

```

        buffer.seek(0)
        response = HttpResponse(buffer, content_type="application/zip")
        response['Content-Disposition'] = 'attachment; file-
name=qr_codes.zip'
        return response

@admin.register(Table)
class TableAdmin(admin.ModelAdmin):
    list_display = ('number', 'qr_uuid', 'qr_download_btn')

    # Додавання дій для генерації нових qr-кодів та завантаження їх
    архівом
    actions = [regenerate_qr_uuids, download_qr_archive]

    def qr_download_btn(self, obj):
        url = reverse('menu:download_qr', args=[obj.id])
        return format_html('<a class="button" href="{}" style="back-
ground-color: #28a745; color: white; padding: 5px 10px; border-radius:
4px; text-decoration: none;">Завантажити</a>', url)

    qr_download_btn.short_description = 'QR-код'

# Замовлення та позиції замовлення будуть відображатися разом
class OrderItemInline(admin.TabularInline):
    model = OrderItem
    extra = 0
    readonly_fields = ('price_at_moment', 'cost_at_moment')

@admin.register(Order)
class OrderAdmin(admin.ModelAdmin):
    list_display = ('id', 'table', 'status', 'created_at', 'get_to-
tal_items')
    list_filter = ('status', 'created_at', 'table')
    inlines = [OrderItemInline]

    def get_total_items(self, obj):
        return obj.items.count()

```

```

get_total_items.short_description = 'К-сть позицій'

@admin.register(AnalyticsProxy)
class AnalyticsProxyAdmin(admin.ModelAdmin):
    # Ця модель використовується для відображення вкладки (відсутня в
    Базі Даних)
    def has_add_permission(self, request):
        return False

    def has_change_permission(self, request, obj=None):
        return False

    def changelist_view(self, request, extra_context=None):
        return HttpResponseRedirect(reverse('menu:monitoring_dash-
board'))

```

Б.6 Текст файлу dashboard.py

```

import pandas as pd
from django.db.models import Sum, Avg
from menu_app.models import Order, OrderItem
from django.db.models.functions import TruncDate
from django.db.models import F, Count

def calculate_kasavana_smith(start_date=None, end_date=None):
    # Алгоритм Касавани-Сміта з медіаною для популярності та питомою
    маржею для прибутковості

    qs = OrderItem.objects.filter(order__status='paid')

    # Валідація дат
    if start_date and end_date and start_date > end_date:
        raise ValueError("Початкова дата не може бути більшою за кін-
цеву.")

    if start_date:
        qs = qs.filter(order__created_at__gte=start_date)
    if end_date:

```

```

qs = qs.filter(order__created_at__lte=end_date)

# Агрегація: обирається середня ціна та собівартість (якщо ціни
зміняться)
aggregated_data = qs.values(
    'dish__id', 'dish__name'
).annotate(
    total_sold=Sum('quantity'),
    avg_price=Avg('price_at_moment'),
    avg_cost=Avg('cost_at_moment')
)

empty_result = {'averages': {}, 'items': []}
if not aggregated_data:
    return empty_result

df = pd.DataFrame(list(aggregated_data))

df['avg_price'] = df['avg_price'].astype(float)
df['avg_cost'] = df['avg_cost'].astype(float)

# Розрахунок питомої маржі та сумарного доходу
df['unit_margin'] = df['avg_price'] - df['avg_cost']
df['total_revenue'] = df['avg_price'] * df['total_sold']

# Додавання відсотку від загального обороту
total_system_revenue = df['total_revenue'].sum()
if total_system_revenue > 0:
    df['revenue_percentage'] = (df['total_revenue'] /
total_system_revenue * 100).round(2)
else:
    df['revenue_percentage'] = 0.0

# ABC-аналіз (За правилом Парето 80/15/5)

df = df.sort_values(by='total_revenue', ascending=False)

# Обрахунок накопичувальної суми відсотків
df['cum_percentage'] = df['revenue_percentage'].cumsum()

```

```

# Призначення класів (А - топ 80% доходу, В - 15%, С - 5%)
def assign_abc(cum_pct):
    if cum_pct <= 80:
        return 'A'
    elif cum_pct <= 95:
        return 'B'
    else:
        return 'C'

df['abc_class'] = df['cum_percentage'].apply(assign_abc)

# Для популярності використовується медіана
threshold_sold = df['total_sold'].median()

# Для прибутковості використовується середня питома маржа
threshold_margin = df['unit_margin'].mean()

def categorize_and_advise(row):
    is_high_sold = row['total_sold'] >= threshold_sold
    is_high_margin = row['unit_margin'] >= threshold_margin

    if is_high_sold and is_high_margin:
        return pd.Series(['Зірка (Star)', 'success', 'Зберегти як-ість. Рекламувати.'])
    elif is_high_sold and not is_high_margin:
        return pd.Series(['Конячка (Plow Horse)', 'warning', 'Обережно підняти ціну або зменшити порцію.'])
    elif not is_high_sold and is_high_margin:
        return pd.Series(['Загадка (Puzzle)', 'info', 'Високий потенціал. Додати в Рекомендації.'])
    else:
        return pd.Series(['Собака (Dogs)', 'danger', 'Видалити з меню або повністю змінити.'])

df[['category_name', 'color_class', 'action_plan']] = df.apply(categorize_and_advise, axis=1)

result = {

```

```

        'averages': {
            'threshold_sold': int(threshold_sold),
            'threshold_margin': round(threshold_margin, 2)
        },
        'items': df.to_dict(orient='records')
    }

    return result

# Функція для отримання динаміки продажів за певний період
def get_sales_dynamics():
    qs = Order.objects.filter(status='paid').annotate(
        date=TruncDate('created_at')
    ).values('date').annotate(
        total_revenue=Sum(F('items__price_at_moment')
F('items__quantity')),
        orders_count=Count('id', distinct=True)
    ).order_by('date')

    dates = [item['date'].strftime('%d.%m.%Y') for item in qs]
    revenues = [float(item['total_revenue'] or 0) for item in qs]

    return {
        'dates': dates,
        'revenues': revenues
    }

```

Б.7 Текст файлу qr_code_generator.py

```

import qrcode
from PIL import Image, ImageDraw

def generate_qr_with_logo(url, logo_path):
    # Логотип
    logo = Image.open(logo_path).convert("RGBA")

    # Перетворення всіх непрозорих пікселів на чорні (якщо потрібно)
    pixels = logo.load()

```

```

for y in range(logo.size[1]):
    for x in range(logo.size[0]):
        r, g, b, a = pixels[x, y]
        if a > 0: # непрозорий піксель
            pixels[x, y] = (0, 0, 0, a)

# Створення QR-коду
qr = qrcode.QRCode(
    version=2,
    error_correction=qrcode.constants.ERROR_CORRECT_H,
    box_size=12,
    border=4
)
qr.add_data(url)
qr.make(fit=True)

qr_img = qr.make_image(fill_color="black",
back_color="white").convert("RGBA")

# Масштабування логотипу
box_size = qr_img.size[0] // 4
logo_resized = logo.resize((box_size, box_size))

# Центр
pos = ((qr_img.size[0] - logo_resized.size[0]) // 2,
      (qr_img.size[1] - logo_resized.size[1]) // 2)

# Білий фон під логотипом
draw = ImageDraw.Draw(qr_img)
padding = 10
bg_box = [pos[0]-padding, pos[1]-padding,
          pos[0]+logo_resized.size[0]+padding,
          pos[1]+logo_resized.size[1]+padding]
draw.rounded_rectangle(bg_box, fill="white")

# Вставка логотипу
qr_img.paste(logo_resized, pos, mask=logo_resized)

return qr_img

```

Б.8 Текст файлу recommendations.py

```

from django.db.models import Count
from menu_app.models import Dish, OrderItem

def get_sell_recommendations(cart_dish_ids, limit=4):
    # MVP алгоритм асоціативних правил (Market Basket Analysis)
    # Пошук товарів, які найчастіше купують разом із товарами в кошику

    if not cart_dish_ids:
        return []

    # Пошук всіх замовлень, в яких є хоча б одна страва з кошика
    orders_with_cart_items = OrderItem.objects.filter(
        dish_id__in=cart_dish_ids
    ).values('order')

    # Пошук інших страв в замовленнях, підрахунок їх кількості,
    сортування
    recommendations = Dish.objects.filter(
        orderitem__order__in=orders_with_cart_items,
        is_active=True # Для рекомендації позиція повинна бути в наяв-
ности
    ).exclude(
        id__in=cart_dish_ids # Якщо позиція вже в кошику - ігнорувати
    ).annotate(
        times_bought_together=Count('orderitem__order')
    ).order_by('-times_bought_together')[:limit]

    # Список словників для json
    result = []
    for dish in recommendations:
        result.append({
            'id': dish.id,
            'name': dish.name,
            'price': float(dish.price),
            'image_url': dish.image.url if dish.image else '',

```

```

    })

    return result

```

Б.9 Текст файлу monitoring_dashboard.html

```

{# Сторінка дашборду в панелі адміністратора #}
{% extends "admin/base_site.html" %}
{% load static %}

{% block extrastyle %}
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/@fortawesome/fontawesome-free@6.0.0/css/all.min.css">
<style>
    .dashboard-container {
        padding: 20px;
        font-family: sans-serif;
    }
    .metrics-row {
        display: flex;
        gap: 20px;
        margin-bottom: 30px;
        flex-wrap: wrap;
    }
    .metric-card {
        background: #fff;
        border: 1px solid #ddd;
        padding: 20px;
        border-radius: 8px;
        flex: 1; min-width: 250px;
        box-shadow: 0 2px 4px rgba(0,0,0,0.05);
    }
    .metric-card h3 {
        margin: 0 0 10px 0;
        color: #666;
        font-size: 14px;
    }

```

```

        text-transform: uppercase;
        font-weight: bold;
    }
    .metric-card .value {
        font-size: 28px;
        font-weight: bold;
        color: #111;
    }

    .chart-container {
        background: #fff;
        padding: 20px;
        border: 1px solid #ddd;
        border-radius: 8px;
        margin-bottom: 30px;
        height: 60vh;
        margin: 0 auto 30px auto;
        max-width: 1000px;
    }

    .results-table { width: 100%; border-collapse: collapse; background:
    #fff; box-shadow: 0 2px 4px rgba(0,0,0,0.05); font-size: 1.05em; }
    .results-table th, .results-table td { border: 1px solid #ddd; padding:
    14px 12px; text-align: left; }
    .results-table th { background-color: #f8f9fa; font-weight: 800;
    color: #000; text-transform: uppercase; font-size: 0.9em; letter-spacing:
    0.5px; }
    .results-table td { color: #e1e1e1; }

    .badge { padding: 6px 12px; border-radius: 20px; font-size: 13px;
    font-weight: bold; color: white; display: inline-block; }
    .bg-success { background-color: #28a745; }
    .bg-warning { background-color: #ffc107; color: #212529; }
    .bg-info { background-color: #17a2b8; }
    .bg-danger { background-color: #dc3545; }

    .abc-A { color: #28a745 !important; font-weight: 800; }
    .abc-B { color: #ffc107!important; font-weight: 800; }
    .abc-C { color: #dc3545!important; font-weight: 800; }

```

```

body {
    background-color: var(--body-bg, #121212) !important;
    color: var(--body-fg, #ccc) !important;
}

#content {
    background-color: transparent !important;
}

#methodologyAccordion ul li {
    list-style: none !important;
    margin-left: 0 !important;
}

#methodologyAccordion ul li::before {
    display: none !important;
    content: none !important;
}

.nav-tabs, .nav-tabs .nav-item {
    list-style: none !important;
    list-style-type: none !important;
    padding-left: 0 !important;
}

.nav-tabs .nav-item::before {
    display: none !important;
}

.nav-tabs .nav-link {
    user-select: none !important;
    -webkit-user-select: none !important;
    cursor: pointer !important;
    outline: none !important;
}

.nav-tabs .nav-link:not(.active) {
    color: #888 !important;
}

```

```

</style>
{% endblock %}

{% block content %}
<div class="dashboard-container">
    <h2 class="mb-4">Панель управління та Моніторингу</h2>

    <ul class="nav nav-tabs mb-4" id="dashboardTabs" role="tablist">
        <li class="nav-item" role="presentation">
            <button class="nav-link active fw-bold text-dark" id="sppr-
tab" data-bs-toggle="tab" data-bs-target="#sppr" type="button"
role="tab" aria-selected="true">
                <i class="fas fa-chart-scatter text-primary me-1"></i>
Матриця рентабельності
            </button>
        </li>
        <li class="nav-item" role="presentation">
            <button class="nav-link fw-bold text-dark" id="sales-tab"
data-bs-toggle="tab" data-bs-target="#sales" type="button" role="tab"
aria-selected="false">
                <i class="fas fa-chart-line text-success me-1"></i>
Динаміка продажів
            </button>
        </li>
    </ul>

    <div class="tab-content" id="dashboardTabsContent">
        <div class="tab-pane fade show active" id="sppr" role="tabpanel"
aria-labelledby="sppr-tab">
            <p class="text-muted" style="color: white !im-
portant;">Комплексний огляд рентабельності меню на базі матриці
Касавани-Сміта та АВС-аналізу.</p>
            <p class="small text-info mb-4"><i class="fas fa-search-
plus"></i> Використовуйте колесо миші для масштабування графіка.
Затисніть ліву кнопку миші для переміщення.</p>

        </div>
    </div>
</div>

{# Методологія #}

```

```

<div class="accordion mb-4" id="methodologyAccordion">
  <div class="accordion-item border-0 shadow-sm rounded-3 overflow-
hidden">
    <h2 class="accordion-header" id="headingOne">
      <button class="accordion-button collapsed bg-light fw-bold
text-dark border-0 py-3 px-4"
        type="button" data-bs-toggle="collapse"
        data-bs-target="#collapseOne" aria-expanded="false"
        aria-controls="collapseOne">
        <i class="fas fa-info-circle text-primary me-3"></i>
        Як читати цей дашбورد? (Методологія)
      </button>
    </h2>
    <div id="collapseOne" class="accordion-collapse collapse"
      aria-labelledby="headingOne" data-bs-parent="#methodolog-
yAccordion">
      <div class="accordion-body bg-white p-4">
        <div class="row g-4">
          <div class="col-md-6">
            <h6 class="fw-bold text-dark mb-3 d-flex align-
items-center">
              <i class="fas fa-chart-pie text-success me-
2"></i>
              Матриця Касавани-Сміта
            </h6>
            <ul class="list-unstyled ps-2">
              <li class="mb-2">
                <span class="list-unstyled badge bg-
success">Зірки (Stars)</span> –
                Високий прибуток, високі продажі. Ваші
найкращі позиції.
              </li>
              <li class="mb-2">
                <span class="badge bg-warning">Конячки
(Plow Horses)</span> –
                Низький прибуток, високі продажі. При-
носять потік клієнтів.
              </li>
              <li class="mb-2">

```

```

        <span class="badge bg-info">Загадки
(Puzzles)</span> –
        Високий прибуток, низькі продажі. По-
требують агресивної реклами.
    </li>
    <li class="mb-2">
        <span class="badge bg-danger">Собаки
(Dogs)</span> –
        Низький прибуток, низькі продажі. Кан-
дидати на вилучення.
    </li>
</ul>
</div>
<div class="col-md-6 border-start border-light">
    <h6 class="fw-bold text-dark mb-3 d-flex align-
items-center">
        <i class="fas fa-sitemap text-warning me-
2"></i>
        ABC-Аналіз (Закон Парето)
    </h6>
    <ul class="list-unstyled ps-2">
        <li class="mb-2">
            <strong class="text-success">Клас
A</strong> –
            Товари, що приносять ~80% загального до-
ходу.
        </li>
        <li class="mb-2">
            <strong class="text-warning">Клас
B</strong> –
            Наступні ~15% доходу.
        </li>
        <li class="mb-2">
            <strong class="text-danger">Клас
C</strong> –
            Останні ~5% доходу (найменш важливі
фінансово).
        </li>
    </ul>

```

```

        </div>
    </div>
</div>
</div>

{% if data.items %}

{# Вивід ключових метрик в картках #}
<div class="metrics-row">
    <div class="metric-card">
        <h3>Позицій в меню</h3>
        <div class="value">{{ data.items|length }} шт.</div>
    </div>
    <div class="metric-card">
        <h3>Медіана продажів (Попір)</h3>
        <div class="value">{{ data.averages.threshold_sold }}
шт.</div>
    </div>
    <div class="metric-card">
        <h3>Середня питома маржа (Попір)</h3>
        <div class="value">{{ data.averages.threshold_margin }}
%</div>
    </div>
</div>

{# Графік розподілу страв за популярністю та рентабельністю #}
<div class="chart-container">
    <canvas id="matrixChart"></canvas>
</div>

{# Таблиця з результатами #}
<table class="results-table">
    <thead>
        <tr>
            <th>Назва страви</th>
            <th>Продано (шт)</th>

```

```

        <th>Маржа (€)</th>
        <th>Частка обороту</th>
        <th>ABC Клас</th>
        <th>Категорія (Касавана-Сміт)</th>
        <th>План дій (Action Plan)</th>
    </tr>
</thead>
<tbody>
    {% for item in data.items %}
    <tr>
        <td><strong>{{ item.dish__name }}</strong></td>
        <td>{{ item.total_sold }}</td>
        <td>{{ item.unit_margin|floatformat:2 }} €</td>
        <td>{{ item.revenue_percentage }}%</td>
        <td class="abc-{{ item.abc_class }}">Клас {{
item.abc_class }}</td>
        <td>
            <span class="badge bg-{{ item.color_class }}">{{
item.category_name }}</span>
        </td>
        <td>{{ item.action_plan }}</td>
    </tr>
    {% endfor %}
</tbody>
</table>
</div>
{# Вкладка перегляду щоденної виручки #}
<div class="tab-pane fade" id="sales" role="tabpanel" aria-la-
belledby="sales-tab">
    <div class="d-flex justify-content-between align-items-cen-
ter mb-3 flex-wrap gap-2">
        <p class="mb-0">Моніторинг щоденної виручки
закладу.</p>

        <div class="d-flex gap-2 align-items-center">
            <select id="yearFilter" class="form-select form-se-
lect-sm" style="width: auto; cursor: pointer;">
                <option value="">Всі роки</option>
                <option value="2025">2025</option>

```

```

        <option value="2026">2026</option>
    </select>

    <select id="monthFilter" class="form-select form-
select-sm" style="width: auto; cursor: pointer;">
        <option value="">Всі місяці</option>
        <option value="01">Січень</option>
        <option value="02">Лютий</option>
        <option value="03">Верезень</option>
        <option value="04">Квітень</option>
        <option value="05">Травень</option>
        <option value="06">Червень</option>
        <option value="07">Липень</option>
        <option value="08">Серпень</option>
        <option value="09">Вересень</option>
        <option value="10">Жовтень</option>
        <option value="11">Листопад</option>
        <option value="12">Грудень</option>
    </select>
</div>
</div>

<div class="chart-container">
    <canvas id="salesChart"></canvas>
</div>
</div>
</div>
{% else %}
<div class="metric-card" style="text-align: center; color:
#dc3545;">
    <h3>Немає даних для обробки</h3>
    <p>Спочатку потрібно отримати оплачені замовлення.</p>
</div>
{% endif %}
</div>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/boot-
strap.bundle.min.js"></script>

```

```

<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
<script src="https://cdn.jsdelivr.net/npm/hammerjs@2.0.8"></script>
<script          src="https://cdn.jsdelivr.net/npm/chartjs-plugin-zoom@2.0.1/dist/chartjs-plugin-zoom.min.js"></script>
<script          src="https://cdn.jsdelivr.net/npm/chartjs-plugin-annotation@2.0.1/dist/chartjs-plugin-annotation.min.js"></script>
<script>
    {% if data.items %}
    document.addEventListener("DOMContentLoaded", function() {
        const rawData = {{ data.items|safe }};
        const avgSold = parseInt({{ data.averages.threshold_sold|safe
    }}});

        const avgMargin = {{ data.averages.threshold_margin|safe }};

        // Легенда для графіку
        const groupedData = {
            'Зірка (Star)': { color: '#28a745', points: [] },
            'Конячка (Plow Horse)': { color: '#ffc107', points: [] },
            'Загадка (Puzzle)': { color: '#17a2b8', points: [] },
            'Собака (Dogs)': { color: '#dc3545', points: [] }
        };

        rawData.forEach(item => {
            if (groupedData[item.category_name]) {
                groupedData[item.category_name].points.push({
                    x: item.total_sold,
                    y: item.unit_margin,
                    label: item.dish__name
                });
            }
        });

        // Формування масиву datasets
        const datasets = Object.keys(groupedData).map(category => {
            return {
                label: category,
                data: groupedData[category].points,
                backgroundColor: groupedData[category].color,
                pointRadius: 6,
            }
        })
    }

```

```

        pointHoverRadius: 8
    };
});

const ctx = document.getElementById('matrixChart').getContext('2d');
new Chart(ctx, {
    type: 'scatter',
    data: { datasets: datasets },
    options: {
        responsive: true,
        maintainAspectRatio: false,
        plugins: {
            legend: {
                position: 'right',
                labels: {
                    font: { size: 14, weight: 'bold' },
                    padding: 20
                }
            },
            tooltip: {
                callbacks: {
                    label: function(ctx) {
                        const raw = ctx.raw;
                        // Округлення маржі
                        const marginRounded = parseFloat(raw.y).toFixed(2);
                        return raw.label + ': Продано ' + raw.x
                        + ' шт, Маржа ' + marginRounded + ' €';
                    }
                }
            },
            zoom: {
                pan: { enabled: true, mode: 'xy' },
                zoom: {
                    wheel: { enabled: true },
                    pinch: { enabled: true },
                    mode: 'xy'
                }
            }
        }
    }
});

```

```

    },
    annotation: {
      annotations: {
        verticalLine: {
          type: 'line',
          xMin: avgSold,
          xMax: avgSold,
          borderColor: '#666666',
          borderWidth: 2,
          borderDash: [6, 6]
        },
        horizontalLine: {
          type: 'line',
          yMin: avgMargin,
          yMax: avgMargin,
          borderColor: '#666666',
          borderWidth: 2,
          borderDash: [6, 6]
        }
      }
    },
    scales: {
      x: {
        min: Math.min(...rawData.map(d => d.to-
tal_sold)) - 1,
        max: Math.max(...rawData.map(d => d.to-
tal_sold)) + 1,
        title: {
          display: true,
          text: 'Популярність (Кількість проданих
порцій)',
          font: { size: 14, weight: 'bold' },
          color: '#000'
        },
        ticks: {
          font: { weight: 'bold', size: 12 },
          color: '#333'
        }
      },

```

```

        grid: {
            color: (ctx) => ctx.tick.value === avgSold
? 'rgba(0, 0, 0, 0.8)' : 'rgba(0, 0, 0, 0.1)',
            lineWidth: (ctx) => ctx.tick.value === avg-
Sold ? 2 : 1
        }
    },
    y: {
        title: {
            display: true,
            text: 'Прибутковість (Питома маржа, %)',
            font: { size: 14, weight: 'bold' },
            color: '#000'
        },
        ticks: {
            font: { weight: 'bold', size: 12 },
            color: '#333'
        },
        grid: {
            color: (ctx) => ctx.tick.value === avgMargin
? 'rgba(0, 0, 0, 0.8)' : 'rgba(0, 0, 0, 0.1)',
            lineWidth: (ctx) => ctx.tick.value ===
avgMargin ? 2 : 1
        }
    }
}

});

// Графік динаміки продажів
{% if sales_data %}
const allSalesDates = {{ sales_data.dates|safe }};
// "DD.MM.YYYY"
const allSalesRevenues = {{ sales_data.revenues|safe }};

const ctxSales = document.getElementById('salesChart').getCon-
text('2d');
let salesChart = new Chart(ctxSales, {
    type: 'bar',

```

```

data: {
  labels: allSalesDates,
  datasets: [{
    label: 'Щоденна виручка (€)',
    data: allSalesRevenues,
    backgroundColor: 'rgba(40, 167, 69, 0.7)',
    borderColor: 'rgba(40, 167, 69, 1)',
    borderWidth: 1,
    borderRadius: 4
  }]
},
options: {
  responsive: true,
  maintainAspectRatio: false,
  plugins: { legend: { display: false } },
  scales: {
    y: { beginAtZero: true, title: { display: true,
text: 'Виручка (€)', font: { weight: 'bold' } } },
    x: { title: { display: true, text: 'Дата', font: {
weight: 'bold' } } }
  }
}
});

function filterSalesChart() {
  const selectedYear = document.getElementById('yearFil-
ter').value;
  const selectedMonth = document.getElementById('monthFil-
ter').value;

  let filteredDates = [];
  let filteredRevenues = [];

  for (let i = 0; i < allSalesDates.length; i++) {
    const dateStr = allSalesDates[i]; // "DD.MM.YYYY"
    const parts = dateStr.split('.');
    const mMonth = parts[1];
    const mYear = parts[2];

```

```

        const matchYear = (selectedYear === "" || selectedYear
=== mYear);

        const matchMonth = (selectedMonth === "" || selected-
Month === mMonth);

        if (matchYear && matchMonth) {
            filteredDates.push(dateStr);
            filteredRevenues.push(allSalesRevenues[i]);
        }
    }

    salesChart.data.labels = filteredDates;
    salesChart.data.datasets[0].data = filteredRevenues;
    salesChart.update();
}

    document.getElementById('yearFilter').addEventListener('change', filterSalesChart);
    document.getElementById('monthFilter').addEventListener('change', filterSalesChart);
    {% endif %}

});
{% endif %}
</script>
{% endblock %}

```

ДОДАТОК В
Слайди презентації

Національний університет «Запорізька політехніка»
Факультет комп'ютерних наук та технологій
Кафедра програмних засобів

Дипломна робота на тему:
Розроблення та дослідження
програмного забезпечення
автоматизації закладів харчування

Виконав: ст. гр. КНТ-132

Роман МАРЯХІН

Керівник:
к.ф.-м.н. доцент

Олена ПОДКОВАЛІХІНА

Рисунок В.1 – Слайд 1

Актуальність, об'єкт, предмет та мета

2

Актуальність: необхідність цифрової трансформації HoReCa для зниження витрат та автоматизації прийому замовлень.

Об'єкт дослідження: процес розроблення програмного забезпечення для автоматизації сервісних операцій та програмної реалізації методів обробки даних у ресторанному бізнесі.

Предмет: методи програмної реалізації QR-інтерфейсів та алгоритми автоматизованого розрахунку параметрів асортименту.

Мета: підвищення швидкості обробки замовлень через впровадження QR-технологій та автоматизація отримання результатів оцінки рентабельності меню.

Рисунок В.2 – Слайд 2

Аналіз предметної області та огляд існуючих рішень

3

Функціональна можливість	Poster [1]	ChoiceQR [2]	MeMenu [3]
Електронне QR-меню	Так	Так	Так
Матриця Касавани-Сміта	Ні	Ні	Ні
АВС-аналіз	Так	Ні	Ні
Система розумних рекомендацій	Ні	Так	Так
Розрахунок динаміки продажів у реальному часі	Так	Так	Ні

Висновок: існуючі рішення зосереджені на касі або візуальному меню, але не мають вбудованих обчислювальних алгоритмів для глибокої класифікації страв.

Рисунок В.3 – Слайд 3

Вибір мови програмування

4

Критерій	PHP	JavaScript (Node.js)	Python
Основна сфера застосування	Створення CMS, класичних веб-сайтів	Високонавантажені real-time застосунки	Web-розробка, Data Science, машинне навчання
Інструментарій для обробки даних	Обмежений	Обмежений	Максимальний (Pandas, NumPy)
Швидкість розробки обчислювальних модулів	Середня	Низька	Висока

Висновок: обрано Python завдяки оптимізованим бібліотекам для матричних обчислень.

Рисунок В.4 – Слайд 4

Вибір серверного фреймворку

Критерій	Flask	FastAPI	Django
Архітектурний підхід	Мікрофреймворк	Мікрофреймворк	Моноліт (MVT)
Вбудована ORM	Немає	Немає	Є
Панель адміністратора	Відсутня	Відсутня	Створюється автоматично

Висновок: Обрано **Django** як основу для створення комплексної системи.

Рисунок В.5 – Слайд 5

Архітектура та структура бази даних

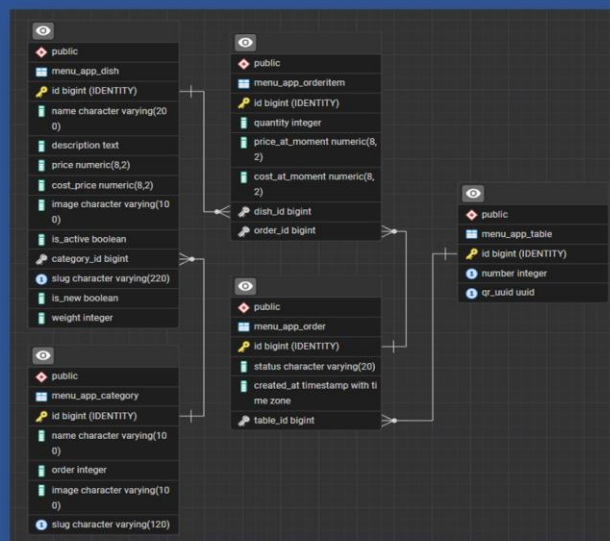


Рисунок В.6 – Слайд 6

Обчислювальний алгоритм ABC-аналізу

7

1. Дохід страви:

$$D_i = K_i \times P_i,$$

K_i – кількість проданих порцій i -ї страви;

P_i – ціна продажу i -ї страви.

2. Кумулятивний дохід страви:

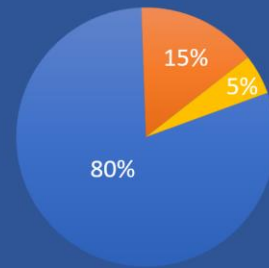
$$Cum D_i = \sum_{i=1}^j D_i \quad (j = 1 \dots n).$$

3. Кумулятивний відсоток:

$$Cum \%_i = \frac{Cum D_i}{D_{total}} \times 100,$$

$D_{total} = \sum_{i=1}^n D_i$ – загальний дохід від усіх проданих страв за обраний період.

ABC-аналіз



- Категорія А: найбільш цінні позиції (80% доходу).
- Категорія В: проміжні позиції (15% доходу).
- Категорія С: найменш значущі позиції (5% доходу).

Рисунок В.7 – Слайд 7

Обчислювальний алгоритм Касавани-Сміта

8

1. Маржинальний прибуток страви:

$$CM_i = P_i - C_i,$$

P_i – ціна одиниці i -ї страви,

C_i – собівартість одиниці i -ї страви.

2. Показник популярності:

$$MM_i = \frac{Q_i}{\sum_{k=1}^n Q_k},$$

Q_i – кількість проданих порцій i -ї страви;

$\sum_{k=1}^n Q_k$ – загальна кількість проданих страв.

3. Середня маржа закладу:

$$CM_{avg} = \frac{\sum (CM_i \times Q_i)}{\sum Q_i}.$$

4. Поріг популярності:

$$Th = Median(Q_1, Q_2 \dots Q_n).$$

Класифікаційні групи

Популярність	Маржинальний внесок	
	Високий	Низький
Висока	Зірки (Stars)	Робочі конячки (Plow Horses)
Низька	Загадки (Puzzles)	Собаки (Dogs)

«Зірки» — популярні та прибуткові позиції, ядро меню;

«Конячки» — популярні страви з низьким прибутком;

«Загадки» — прибуткові позиції з низьким попитом;

«Собаки» — позиції з низьким прибутком та популярністю.

Рисунок В.8 – Слайд 8

Модуль асоціативних зв'язків (МБА)

```
def get_sell_recommendations(cart_dish_ids, limit=4):
    # MVP алгоритм асоціативних правил (Market Basket Analysis)
    # Пошук товарів, які найчастіше купують разом із товарами в кошику

    if not cart_dish_ids:
        return []

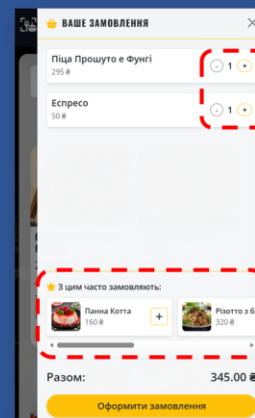
    # Пошук всіх замовлень, в яких є хоча б одна страва з кошика
    orders_with_cart_items = OrderItem.objects.filter(
        dish_id__in=cart_dish_ids
    ).values('order')

    # Пошук інших страв в замовленнях, підрахунок їх кількості, сортування
    recommendations = Dish.objects.filter(
        orderitem__order__in=orders_with_cart_items,
        is_active=True # Для рекомендації позиція повинна бути в наявності
    ).exclude(
        id__in=cart_dish_ids # Якщо позиція вже в кошику - ігнорувати
    ).annotate(
        times_bought_together=Count('orderitem__order')
    ).order_by('-times_bought_together')[:limit]
```

Рисунок В.9 – Слайд 9

Програмна реалізація клієнтських сервісних модулів

Функціональна можливість	Poster	ChoiceQR	MeMenu	Scan & Savor
Рекомендації	Ручні (статичні)	Ручні (статичні)	Відсутні	Так
Архітектура клієнта	Важкі фреймворки	Веб-контейнер	Веб-контейнер	Vanilla JS
Збереження кошика	Cookies / Сесії	Cookies	Локально	Локально
Швидкість відгуку	Середня	Середня	Низька	Висока



Vanilla JavaScript
(Event Listeners)

Asynchronous
Fetch API (JSON
exchange)

Рисунок В.10 – Слайд 10

Візуалізація результатів алгоритмічної обробки даних у модулі адміністратора

11

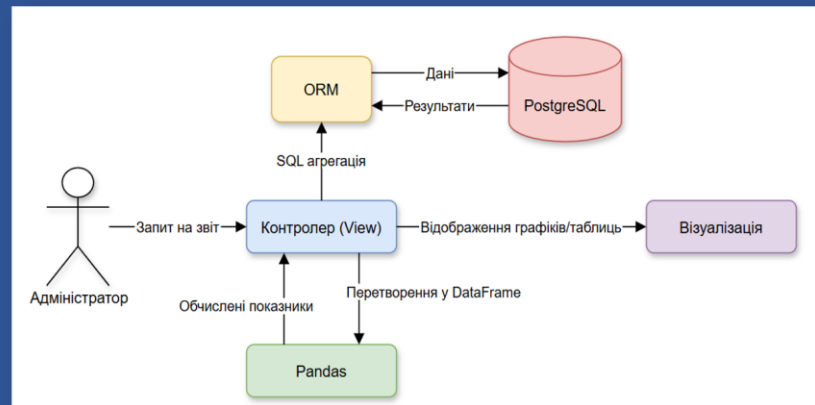


Схема потоку даних для відображення розрахованих даних позицій в меню

Рисунок В.11 – Слайд 11

Візуалізація результатів алгоритмічної обробки даних у модулі адміністратора (Касавани-Сміта)

12

Функціональна можливість	Poster	ChoiceQR	MeMenu	Scan & Savor
Загальний дохід	✓	✓	✓	✓
«Зірки»	✗	✗	✗	✓ 6 страв
«Конячки»	✗	✗	✗	✓ 5 страв
«Загадки»	✗	✗	✗	✓ 6 страв
«Собаки»	✗	✗	✗	✓ 4 страви



Висновок: алгоритм визначив 4 збиткових, 5 популярних, 6 прибуткових та 6 пріоритетних позицій. Це дає змогу приймати рішення на основі точних даних, а не припущень.

Рисунок В.12 – Слайд 12

Візуалізація результатів алгоритмічної обробки даних у модулі адміністратора (ABC-аналізу)

13

Параметри	Poster	ChoiceQR	MeMenu	Scan & Savor
Наявність ABC-звіту	✓	✗	✗	✓
Тип аналізу	Статичний звіт	✗	✗	Динамічний звіт
Обчислювальне ядро	Хмарний скрипт	✗	✗	Pandas
Категоризація	✓	✗	✗	✓

НАЗВА СТРАВИ	ПРОДАНО (ШТ)	МАРЖА (€)	ЧАСТКА ОБОРОТУ	ABC КЛАС	КАТЕГОРІЯ (КАСАВАНА-СМІТ)	ПЛАН ДІЙ (ACTION PLAN)
Рістко з білими грибами	35	224,01 €	9,34%	Клас А	Stops (Star)	Зберігти якість, Рекламувати.
Салат Цезар з куркою	35	204,99 €	8,02%	Клас А	Stops (Star)	Зберігти якість, Рекламувати.
Піца Чотири сир	29	219,42 €	7,4%	Клас А	Stops (Star)	Зберігти якість, Рекламувати.
Піца Вегетариан	35	193,62 €	7,37%	Клас А	Stops (Star)	Зберігти якість, Рекламувати.
Салат з провансу та грушею	31	197,15 €	7,29%	Клас А	Stops (Star)	Зберігти якість, Рекламувати.
Паста Карбонара	31	188,94 €	6,66%	Клас А	Stops (Star)	Зберігти якість, Рекламувати.
Піца Прованс в Фурні	25	217,13 €	5,98%	Клас А	Stops (Star)	Високий потенціал. Додати в Рекламувати.
Паста Амарічана	25	201,80 €	5,47%	Клас А	Stops (Star)	Високий потенціал. Додати в Рекламувати.

Висновок: 12 страв формують 80% доходу, 5 страв – 15%, а 4 страви – лише 5%. Алгоритм виділив ключові позиції (клас «А»), що складають ядро прибутку закладу.

Рисунок В.13 – Слайд 13

Оцінка ефективності розробленої системи

14

Порівняння часових витрат на обслуговування

Етап	Звичайний спосіб обслуговування	Система «Scan & Savor»	Економія часу
Бронювання столика	3 хв	0.5 хв	2.5 хв
Час у черзі	15 хв (фізична черга)	10 хв (віртуальна черга)	5 хв
Оформлення замовлення	5 хв	1 хв	4 хв
Обробка платежу	4 хв	1 хв	3 хв
Загальний час взаємодії	27 хв	12.5 хв	15.5 хв

Рисунок В.14 – Слайд 14

Висновки

15

Задачі виконано, мету досягнуто.

Результати:

1. Програмно реалізовано модифіковані алгоритми класифікації (матриця Касавани-Сміта) із застосуванням медіанних обчислень для підвищення точності розподілу.
2. Реалізовано обчислювальне ядро на базі Pandas для генерації рекомендацій (МВА) шляхом аналізу асоціативних зв'язків у реальному часі.
3. Розроблено та впроваджено модулі безконтактного обслуговування на основі QR-технологій (час обробки замовлення скорочено на 54%).

Практична цінність: створено комплексний програмний продукт, що поєднує клієнтські QR-сервіси з інструментами алгоритмічної підтримки прийняття рішень для галузі HoReCa.

Рисунок В.15 – Слайд 15