МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій

«До захисту допущено»
В.о. завідувача кафедри

_____Світлана ВАЩЕНКО

____ _____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 «Комп'ютерні науки,

освітньо-професійної програми «Інформаційні технології проектування» на тему: «Мобільний додаток підтримки ведення ресторанного бізнесу. Підсистема обліку замовлень»_____

Здобувача групи ІТ-03-2 Федорченка Олександра Віталійовича .
(шифр групи) (прізвище, ім'я, по батькові)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Олександр ФЕДОРЧЕНКО
(підпис) (Ім'я та ПРІЗВИЩЕ здобувача)

Керівник доцент кафедри ІТ, к.т.н., доцент Юлія ПАРФЕНЕНКО
(посада, науковий ступінь, вчене звання, Ім'я та ПРІЗВИЩЕ) _____ (підпис)

Суми – 2024

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
В. о. зав. кафедри ІТ

_____Світлана ВАЩЕНКО
«__» _____ 2024 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА СТУДЕНТУ

Федорченко Олександр Віталійович

1 Тема роботи *Додаток «*Мобільний додаток підтримки ведення ресторанного бізнесу. Підсистема обліку замовлень»

керівник роботи *Парфененко Юлія Вікторівна, к.т.н., доцент*,

затверджені наказом по університету від « *07* » травня 2024 р. №0482-VI

2 Строк подання студентом роботи « 26 » травня 2024 р.

3 Вхідні дані до роботи перелік вимог по розробці вебдодатку «Мобільний додаток підтримки ведення ресторанного бізнесу. Підсистема обліку замовлень», графічні матеріали для наповнення додатку

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) аналіз предметної області, моделювання та проектування вебдодатку, програмна реалізація додатку

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) актуальність, мета, задачі, функціональні вимоги, аналіз аналогів, порівняльна таблиця аналогів, контекстна діаграма ведення щоденника IDEF0, діаграма варіантів використання, приклади документів бази даних, діаграма послідовностей, засоби реалізації, демонстрація вебдодатку, висновки, апробація.

6. Консультанти розділів роботи:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7.Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області	11.04.2024 – 12.04.2024	виконано
2	Планування робіт	13.04.2024 - 18.04.2024	виконано
3	Аналіз продуктів-аналогів	19.04.2024 - 20.04.2024	виконано
4	Складання технічних завдань	21.04.2024 - 23.04.2024	виконано
5	Модельована та проектування вебдодатку	25.04.2024 - 01.05.2024	виконано
6	Розробка вебдодатку	02.05.2024 - 21.05.2024	виконано
7	Тестування вебдодатку	22.05.2024	виконано
8	Оформлення пояснювальної записки	23.05.2024 - 25.05.2024	виконано

Студент _____ Олександр ФЕДОРЧЕНКО

(підпис)

Керівник роботи _____ Юлія ПАРФЕНЕНКО

(підпис)

АНОТАЦІЯ

Тема кваліфікаційної роботи бакалавра «Мобільний додаток підтримки ведення ресторанного бізнесу. Підсистема обліку замовлень».

Пояснювальна записка складається зі вступу, 3 розділів, висновків, списку використаних джерел із 12 найменувань, додатків. Загальний обсяг роботи − 101 сторінка, у тому числі 64 сторінки основного тексту, 2 сторінок списку використаних джерел, 29 сторінкок додатків.

Актуальність роботи ґрунтується на потребі оптимізації процесів ведення ресторанного бізнесу. Особливо це стосується складнощів, пов’язаних з управлінням замовленнями, що вимагає автоматизації для підвищення ефективності та зниження людських помилок. Така проблематика актуальна, особливо в контексті зростаючих вимог до швидкості обслуговування та якості сервісу у ресторанному бізнесі.

Мета роботи: розробка мобільного додатку для ресторанів, який включатиме підсистему обліку замовлень. Додаток покликаний спростити процеси прийняття замовлень, їх обробки та відстеження, що зробить управління ресторанним бізнесом більш ефективним і зменшить час на обслуговування клієнтів. Це підвищить загальну продуктивність роботи ресторану та поліпшить досвід користувачів.

Ключові слова: мобільний додаток, підсистема обліку замовлень, автоматизація процесів, react native.

ЗМІСТ

Вступ

Ресторанний бізнес у сучасному світі зазнає стрімких змін, де вимоги до якості, ефективності та інновацій постійно зростають. Задовольнити клієнтів вишуканими стравами - це лише частина завдань, з якими стикаються власники та менеджери ресторанів. Управління ресторанним бізнесом потребує не лише глибокого розуміння гастрономічної культури, але й високої організаційної компетентності, а також вміння використовувати передові технології [1].

У цифрову епоху, коли технології розвиваються стрімко, мобільні додатки стають невід'ємним інструментом для ефективного управління ресторанним бізнесом. Однією з ключових підсистем у таких додатках є система обліку замовлень. Вона не лише спрощує процес замовлення для клієнтів, але й забезпечує чітку організацію роботи кухні, оптимізує керування персоналом та контроль запасів.

Розробка та впровадження мобільного додатку для обліку замовлень у ресторанному бізнесі стає важливим кроком у забезпеченні конкурентоспроможності в динамічному середовищі ринку. Цей інструмент дозволяє ресторанам автоматизувати ключові процеси, такі як прийняття замовлень, оплата, підтвердження та оновлення статусу. Крім того, він оптимізує управління ресурсами, включаючи контроль персоналу, замовлень та запасів, а також надає аналітичні звіти для прийняття обґрунтованих рішень. Головною метою дослідження є розробка та впровадження цього додатку з акцентом на систему обліку замовлень. Впровадження такого інструменту не тільки покращить ефективність роботи ресторану, але й забезпечить підвищену конкурентоспроможність завдяки кращому обслуговуванню, оптимізації ресурсів та відповіді на потреби клієнтів та тенденції галузі. Це також дозволить ресторанам динамічно реагувати на зміни на ринку та підтримувати високий рівень обслуговування за

допомогою швидкого прийняття замовлень, зручного інтерфейсу та персоналізованих пропозицій для клієнтів.

Метою кваліфікаційної роботи бакалавра є розробка та впровадження мобільного додатку для підтримки ресторанного бізнесу з особливим акцентом на систему обліку замовлень [2].

Для досягнення мети потрібно вирішити такі задачі:

-провести аналіз предметної області розроблення мобільних додатків для ведення ресторанного бізнесу та обліку замовлень;

-виконати постановку задачі та планування робіт з розроблення мобільного додатку;

-спроєктувати підсистему обліку замовлень мобільного додатку підтримки ведення ресторанного бізнесу;

-розробити мобільний додаток підтримки ведення ресторанного бізнесу з підсистемою обліку замовлень;

-провести тестування підсистеми обліку замовлень;

Впровадження такого додатку не лише забезпечить ефективну роботу ресторану, але й дозволить підвищити конкурентоспроможність: за рахунок кращого обслуговування, оптимізації ресурсів та аналітики даних, відповідати вимогам сучасного ринку: динамічно реагувати на потреби клієнтів та тенденції галузі.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд останніх досліджень і публікацій

У сучасному світі мобільні додатки стали невід'ємною частиною ресторанного бізнесу. Вони пропонують зручний спосіб для клієнтів розміщувати замовлення, відстежувати статус замовлення, отримувати лояльність та багато іншого. Підсистема обліку замовлень є однією з найважливіших компонентів мобільного додатку ресторану, оскільки вона дозволяє ресторанам ефективно керувати своїми замовленнями, запасами та персоналом.

Останні дослідження в галузі розробки мобільних додатків для ресторанного бізнесу фокусуються на інноваційних підходах до покращення підсистеми обліку замовлень. Один із ключових напрямків досліджень полягає у використанні штучного інтелекту (ШІ) та машинного навчання (ML) для прогнозування попиту та оптимізації планування персоналу. Це дозволяє ресторанам ефективно використовувати історичні дані та інші фактори для покращення планування робочих годин персоналу та забезпечення якісного обслуговування клієнтів [3].

Ще одним важливим напрямком є використання технологій геолокації для відстеження кур'єрів та забезпечення своєчасної доставки замовлень. Це дозволяє клієнтам в реальному часі відстежувати місцезнаходження свого замовлення, що покращує їх досвід користування додатком та збільшує лояльність.

Також активно досліджується можливість розробки персоналізованих рекомендацій для клієнтів на основі даних про їхні покупки. Це допомагає ресторанам пропонувати клієнтам індивідуально підібрані пропозиції, що сприяє збільшенню продажів та задоволеності клієнтів.

Останні публікації у цій галузі демонструють успішні приклади впровадження нових технологій у ресторанний бізнес. Наприклад, деякі дослідження зосереджені на персоналізації мобільних замовлень їжі за допомогою рекомендацій на основі контексту, що призводить до збільшення продажів. Інші дослідження досліджують використання технології blockchain для забезпечення прозорості та безпеки ланцюжка постачання, або розробляють чат-ботів на основі ШІ для покращення обслуговування клієнтів.

Ось деякі з ключових напрямків досліджень:

Дослідження в роботі [9] пропонує підхід до прогнозування попиту та планування персоналу в ресторанах за допомогою машинного навчання. Автори розробили модель, яка використовує історичні дані про продажі, такі як сезонність, час доби, погодні умови та спеціальні заходи, для прогнозування майбутнього попиту. Ці прогнози потім використовуються для оптимізації планування робочого часу персоналу, що гарантує, що в ресторані буде достатньо персоналу для задоволення попиту в будь-який час [9].

Стаття [10] описує метод прогнозування попиту та планування персоналу в ресторанах з використанням глибокого навчання з підкріпленням. Автори розробили систему, яка може навчатися на даних про історичні продажі та оптимізувати планування персоналу для максимізації прибутку або мінімізації витрат.

Система диспетчеризації замовлень розроблена з урахуванням місця розташування для служб доставки продуктів харчування на основі навчання з підкріпленням. Дослідження пропонує систему розподілу замовлень для служб доставки їжі, яка використовує локаційну інформацію і навчання з підкріпленням для оптимізації маршрутів кур'єрів. Це може призвести до скорочення часу доставки та підвищення задоволеності клієнтів.

У статті [12] описується метод оптимізації маршрутів доставки їжі, який використовує локаційну інформацію та генетичний алгоритм. Це може

призвести до скорочення відстані, пройденої кур'єрами, та економії часу та коштів [12].

Ці дослідження свідчать про активний інтерес до вдосконалення систем обліку замовлень у ресторанному бізнесі за допомогою новітніх технологій та методів, що сприяє покращенню ефективності та конкурентоспроможності галузі.

1.2 Аналіз програмних продуктів - аналогів

Мобільний додаток пропонує підсистему обліку замовлень, яка дозволяє працівникам додавати, переглядати та відстежувати замовлення. Ця підсистема є важливою частиною вашого додатку, оскільки вона допомагає працівникам організувати свою роботу та відстежувати свій прогрес.

Переваги підсистеми обліку замовлень у мобільних додатках ресторанів дуже очевидні та значущі для підвищення ефективності та комфорту у роботі персоналу. Простота використання є ключовим аспектом, оскільки вона дозволяє швидко впроваджувати нових працівників та максимально використовувати можливості системи без додаткового навчання. Це особливо важливо для тих, хто не має досвіду роботи з мобільними додатками, забезпечуючи їм легкий доступ до необхідних інструментів [4].

Ефективність підсистеми проявляється у можливості швидко знаходити та відстежувати замовлення, що дозволяє працівникам бути більш продуктивними та оперативно реагувати на зміни в обстановці. Прозорість, яку забезпечує система, робить процес обліку замовлень більш зрозумілим та контрольованим для персоналу, адже вони мають можливість переглядати історію своїх замовлень та аналізувати статистику.

У той же час, є можливості для подальшого покращення підсистеми. Інтеграція з зовнішніми системами, такими як календарі або CRM, може полегшити координацію завдань та надати працівникам більше контексту для їхніх замовлень. Автоматизація певних процесів, таких як призначення замовлень чи генерація звітів, спростить робочі процеси та звільнить час для більш важливих завдань. Додаткова аналітика допоможе власникам ресторанів та менеджерам краще розуміти динаміку замовлень та вносити оптимальні зміни для підвищення ефективності та якості обслуговування.

Square for Restaurants: Цей додаток пропонує широкий спектр функцій, включаючи прийняття замовлень, відстеження замовлень, управління столами, обробку оплати та друк рахунків. Square також пропонує POS-систему, яка може бути інтегрована з мобільним додатком.

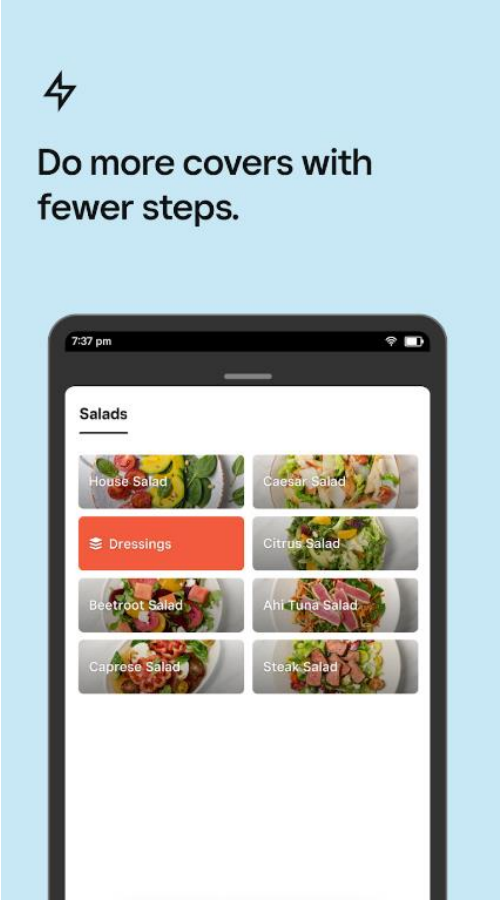

Рисунок 1.1 Square for Restaurants

Lightspeed Restaurant: Цей додаток пропонує комплексне рішення для ведення ресторанного бізнесу, яке включає мобільний додаток, POS-систему, веб-сайт та інструменти управління персоналом. Lightspeed є одним із найдорожчих варіантів, але він пропонує широкий спектр функцій, які можуть бути корисними для великих ресторанів.

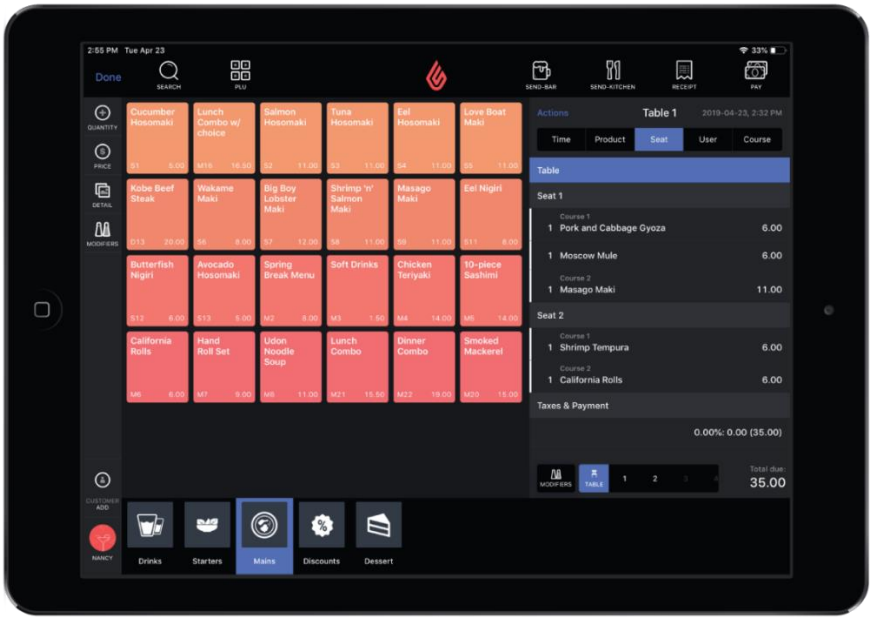

Рисунок 1.2 Lightspeed Restaurant

Toast POS: Цей додаток схожий на Square for Restaurants, але він пропонує деякі додаткові функції, такі як управління персоналом та аналітика даних. Toast також пропонує різні рівні ціноутворення, щоб відповідати потребам різних ресторанів.

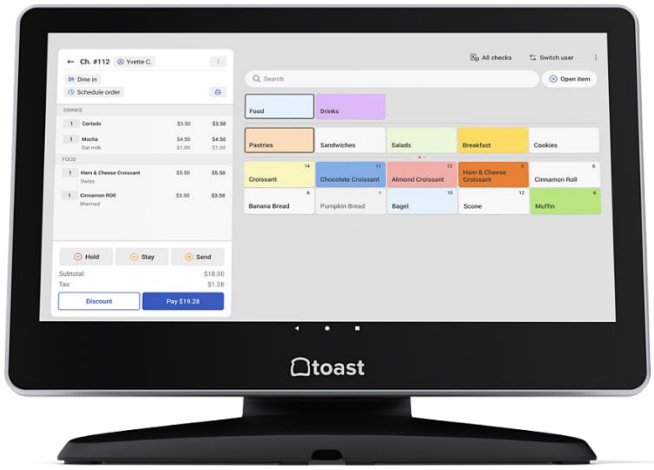

Рисунок 1.3 Toast POS

Програмний продукт	Переваги	Недоліки
Square for Restaurants	Простий у використанні, доступний, пропонує широкий спектр функцій	Не пропонує деяких додаткових функцій, які можуть бути корисними для великих ресторанів
Toast POS	Більше функцій, ніж Square for Restaurants, пропонує різні рівні ціноутворення	Дорожчий, ніж Square for Restaurants
Lightspeed Restaurant	Комплексне рішення, пропонує широкий спектр функцій	Найдорожчий варіант

1.3 Мета та задачі дослідження

Розробка та дослідження мобільного додатку, який допомагатиме ресторанам вести облік замовлень та покращувати управління бізнесом. Підсистема обліку замовлень буде ключовим компонентом цього додатку, надаючи офіціантам можливість вносити замовлення, переглядати історію замовлень та отримувати статистику [5].

Вивчення актуальних тенденцій у ресторанній галузі та аналіз сучасних мобільних додатків для ресторанного бізнесу є ключовим етапом у підготовці до розробки та впровадження мобільного додатку для обліку замовлень у ресторанах. Цей додаток має на меті автоматизувати процеси прийняття замовлень, оплати, підтвердження та оновлення статусу, що сприятиме оптимізації управління ресурсами, включаючи контроль персоналу,

замовлень та запасів. Інтеграція з існуючими системами ресторану дозволить забезпечити плавний перехід та взаємодію між різними компонентами бізнесу.

Впровадження мобільного додатку передбачає проведення тестування в реальних умовах та навчання персоналу його роботі. Окрім цього, запуск додатку для клієнтів відкриє нові можливості для покращення обслуговування, забезпечивши швидке прийняття замовлень та персоналізовані пропозиції.

Після запуску додатку, проводитиметься збір та аналіз даних про його використання для оцінки ефективності та визначення можливостей для подальшого вдосконалення. Очікується, що в результаті цього проекту ресторани зможуть підвищити свою конкурентоспроможність за рахунок автоматизації процесів, покращення якості обслуговування та збільшення лояльності клієнтів, що стане ключовим фактором у сучасному ринковому середовищі.

Завдання підсистеми обліку замовлень в мобільному додатку для ресторанів полягає в забезпеченні оптимальних умов для роботи персоналу та покращенні обслуговування клієнтів. Важливо, щоб офіціанти могли швидко та легко вносити різні типи замовлень через мобільний пристрій, включаючи їжу, напої та спеціальні запити, що сприятиме покращенню ефективності та точності в обробці замовлень.

Підсистема також має надавати можливість перегляду історії замовлень для офіціантів та менеджерів, що дозволить відстежувати тенденції, аналізувати дані та вдосконалювати обслуговування клієнтів. Доступна статистика по замовленнях також є важливою для прийняття обґрунтованих рішень щодо меню, ціноутворення та маркетингових стратегій.

Очікувані результати включають розробку мобільного додатку, який допоможе ресторанам підвищити ефективність у веденні обліку замовлень та управління бізнесом, що в свою чергу призведе до покращення

обслуговування клієнтів та їхньої задоволеності. Для успішної реалізації цих цілей, мобільний додаток має бути простим у використанні та інтуїтивно зрозумілим для офіціантів, надійним та безпечним для захисту даних клієнтів та оптимально масштабованим для використання в ресторанах різного розміру та типу. Впевнений, що ця підсистема стане цінним інструментом для ресторанів, допоможе їм покращити свою роботу та підвищити ефективність.

2 Моделювання та проектування

2.1 Моделювання інформаційної системи

Для моделювання підсистеми обліку замовлень було використано методологію функціонального моделювання IDEF0. Контекстну діаграму, модельовану в одному блоці, на рис. 2.1, показує основні входи, елементи керування, виходи та мехінізми, пов’язані з системою.

Рисунок 2.1 – Контекстна діаграма в нотації IDEF0

Рис. 2.2 демонструє ієрархічну декомпозицію, яка детально розбиває складний процес купівлі товару на складові функції.

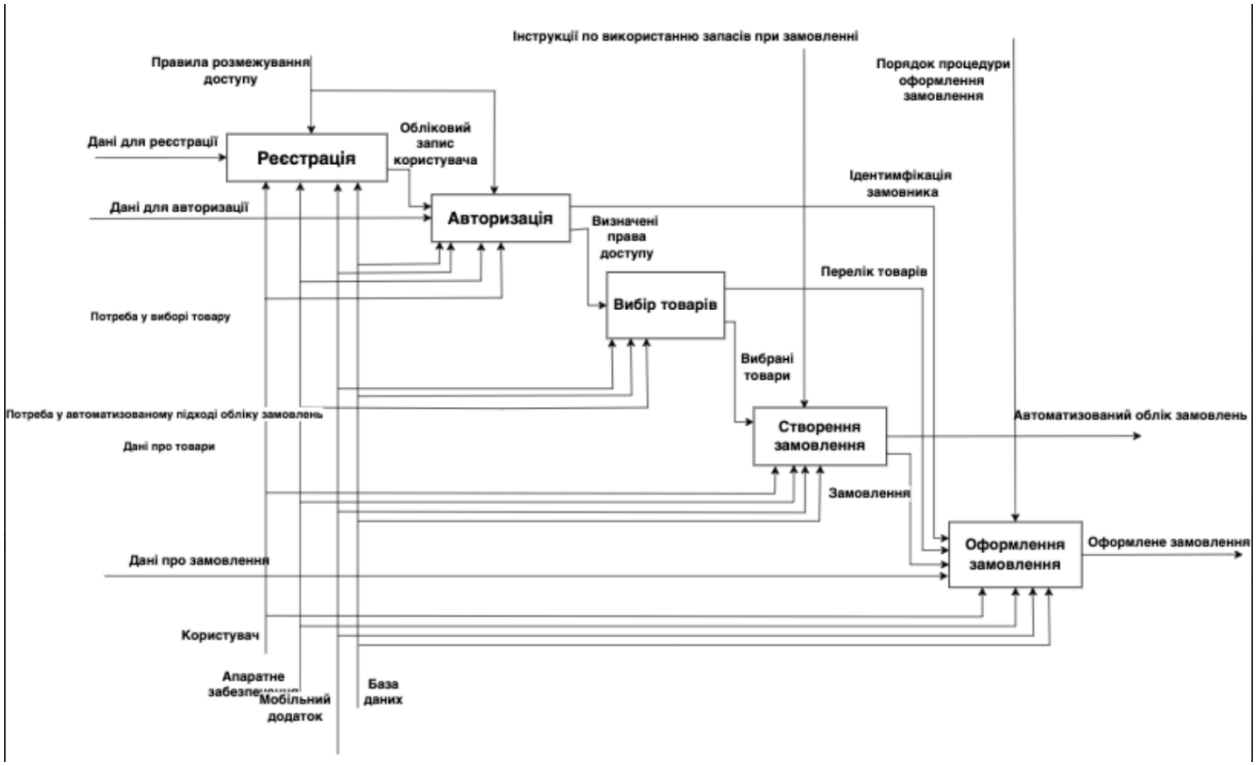

Рисунок 2.2 – Декомпозиція контекстної діаграми

2.2 Діаграма варіантів використання

На рис. 2.3 показано діаграму варіантів використання, яка показує безпосередньо варіанти використання, актори та їхні зв’язки.

Актори:

- Власник бізнесу – особа, яка може переглядати інформацію про замовлення, реєструватися, оформляти замовлення, обробляти замовлення, переглядати інформацію про витрачені товари

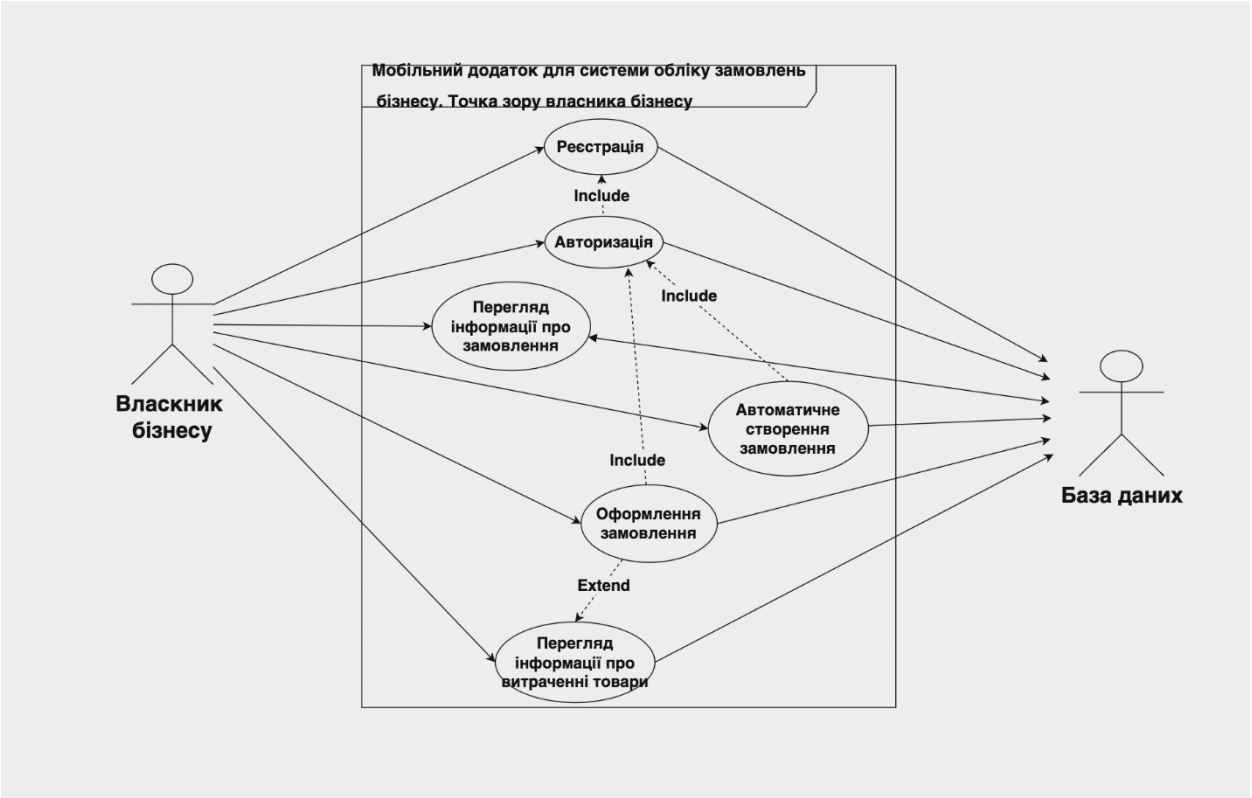

Рисунок 2.3 – Діаграма варіантів використання

Варіанти використання:

- ВВ Реєстрація та авторизація – надає дозвіл реєструватися на авторизовуватися;
- ВВ Робота з переглядом інформація про замовлення – дозволяє власнику переглядати усю інформацію про замовлення;
- ВВ Оформляти замовлення – надає дозвіл фласнику оформляти замовлення;
- ВВ Обробка замовлень – надає дозвіл власнику обробляти замовлення;
- ВВ Перегляд інформація про витрачені товари – надає дозвіл власнику переглядати інформацію про витрачені товари;

2.3 Проектування моделі бази даних

На етапі проектування бази даних необхідно розробити комплексний стратегічний план, щоб гарантувати ефективну реалізацію етапів життєвого циклу бази даних. На цьому етапі необхідно розглянути низку важливих питань, серед яких:

1. Аналіз існуючих інформаційних систем: вивчення та оцінка існуючих інформаційних систем, включаючи їхні переваги, недоліки та потенційні проблеми.

2. Доцільність створення нової інформаційної системи: визначення, чи потрібна нова система на основі оцінки потреб користувачів і недоліків існуючої системи. враховуючи організаційні, фінансові та технічні аспекти.

3. Вартість проекту та обсяг робіт і ресурсів: оцінка необхідного обсягу робіт, ресурсів (фінансових, людських і матеріальних) і вартості проекту.

4. Розробка методології збору даних і визначення їх формату: розробка стратегії збору даних, яка включає способи збору, організації та зберігання даних. Визначення найкращого формату даних для потреб інформаційної системи.

5. Визначення послідовності проектування та реалізації застосувань: встановлення послідовності етапів проектування та реалізації бази даних, включаючи вибір відповідних технологій, розробку моделей і інтерфейсів.

На основі отриманих даних було створено ERD-діаграму реляційної бази даних (рис. 2.4) і детальний опис таблиць інформаційної системи (табл. 2.1) і відповідних полів даних у таблицях (табл. 2.2), які відображають структуру та взаємозв’язки даних у системі.

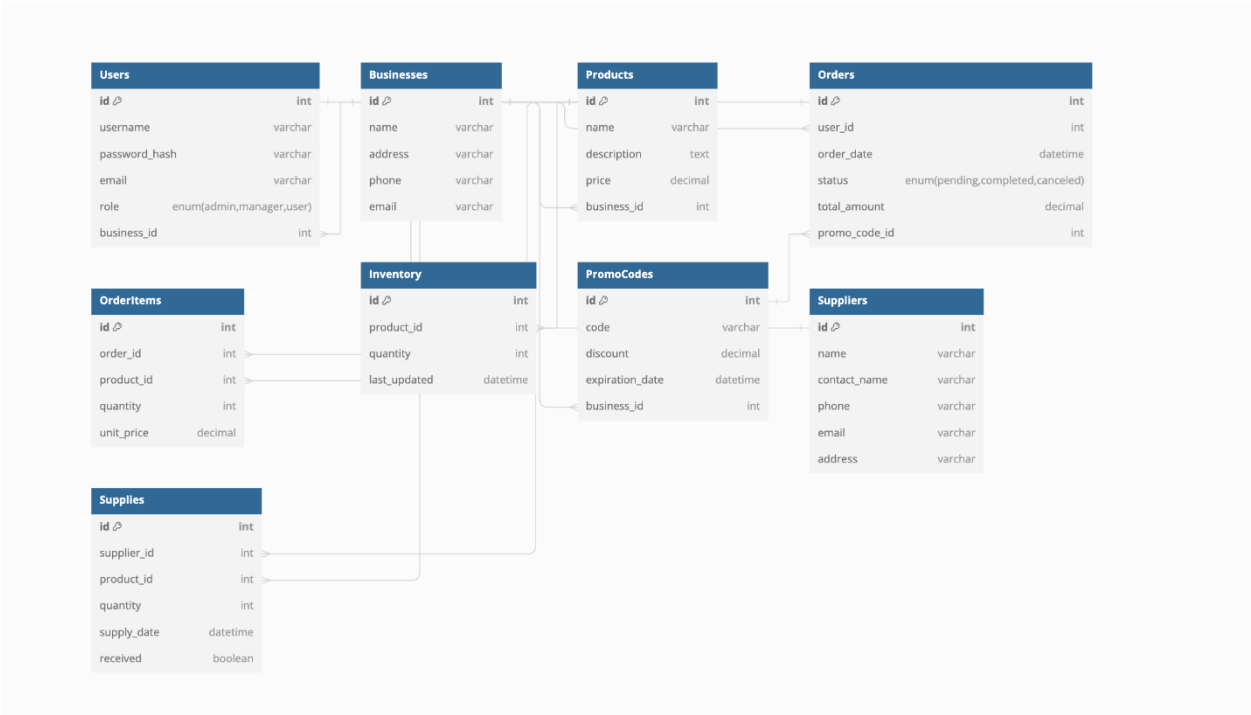

Рисунок 2.4 – ERD діаграма бази даних

Із загальної схеми бази даних мною було розроблено поля Products, Orders, OrderItems, PromoCodes.

Таблиця 2.1 – Опис таблиць бази даних

№	Назва таблиці	Призначення
1	Products	Зберігає інформацію про продукти.
2	Orders	Зберігає інформацію про замовлення.
3	OrderItems	Зберігає інформацію про продукти в замовленнях.
4	PromoCodes	Зберігає інформацію про промокоди.

Таблиця 2.2 – Опис полів бази даних

№	Таблиця	Поле	Зміст	Тип	Ключі	Обмеження
1	Products	id	Унікальний ідентифікатор	INT	PK	Не пустий
		name	Назва продукту	VARCHAR		Не пустий
		description	Опис	TEXT		Не пустий
		price	Ціна	DECIMAL		Не пустий
		business_id	Зовнішній ключ до Businesses	INT	FK	Не пустий
2	Orders	id	Унікальний ідентифікатор	INT	PK	Не пустий
		user_id	Зовнішній ключ до Users	INT	FK	Не пустий
		order_date	Дата замовлення	DATETIME		Не пустий
		status	Статус замовлення (pending, completed, canceled)	ENUM		Не пустий
		total_amount	Загальна сума	DECIMAL		Не пустий
		promo_code_id	Зовнішній ключ до PromoCodes (може бути NULL)	INT	FK	Не пустий
3	OrderItems	id	Унікальний ідентифікатор	INT	PK	Не пустий
		order_date	Зовнішній ключ до Orders	INT	FK	Не пустий
		product_id	Зовнішній ключ до Products	INT	FK	Не пустий
		quantity	Кількість	INT		Не пустий
		unit_price	Ціна за одиницю	DECIMAL		Не пустий
4	PromoCodes	id	Унікальний ідентифікатор	INT	PK	Не пустий
		code	Промокод	VARCHAR		Не пустий
		discount	Знижка	DECIMAL		Не пустий
		expiration_date	Дата закінчення дії	DATETIME		Не пустий
		business_id	Зовнішній ключ до Businesses	INT	FK	Не пустий

На рис. 2.4 представлено фізичну модель бази даних.

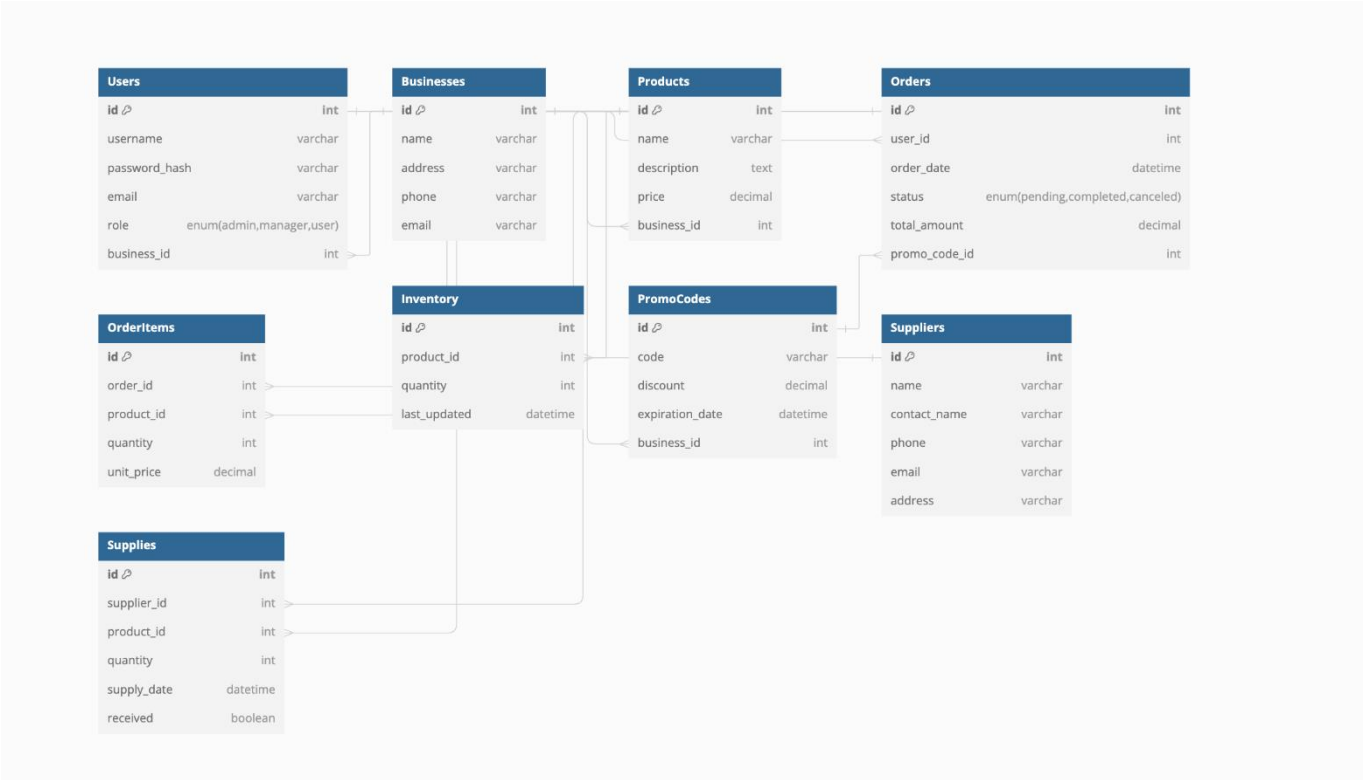

Рисунок 2.4 –База даних

3 Програмна реалізація

3.1 Архітектура додатку

Архітектура мобільного додатку, яка включає в себе підсистему обліку замовлень, показана на рис. 3.1.

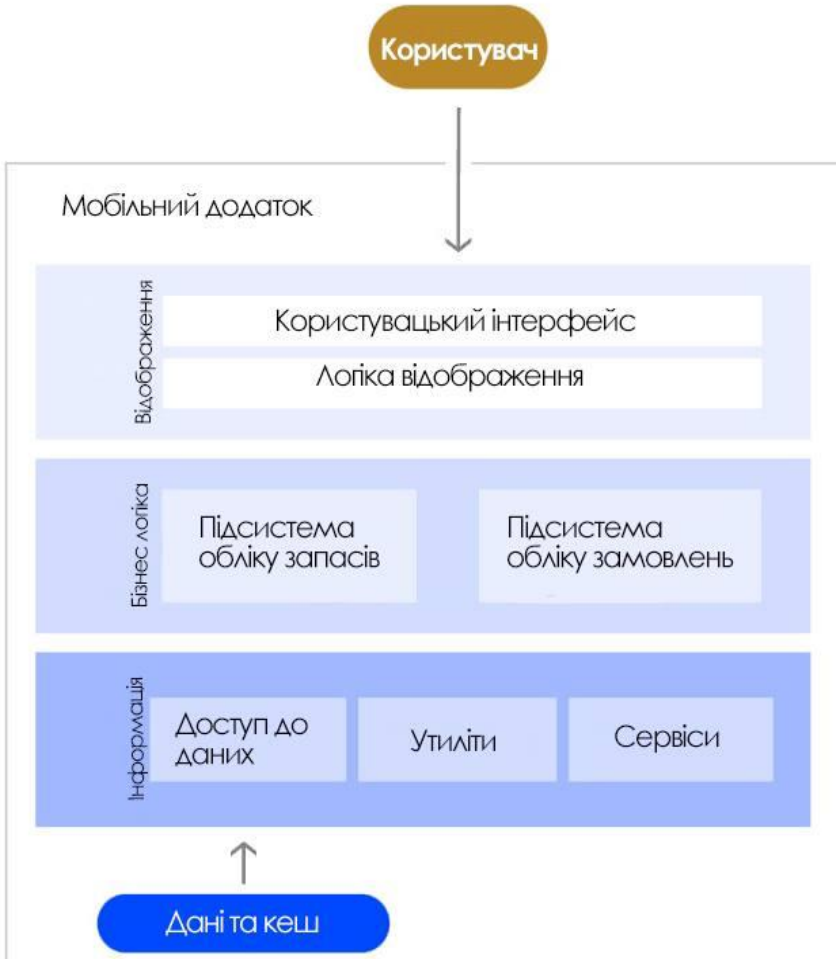

Рисунок 3.1 – Архітектура додатку

Користувацький інтерфейс є простим для користувача та у використанні. Можна швидко та логічно переходити між екранами та розділами, використовуючи його, щоб показати структуру програми. Таким чином, користувачі можуть легко знайти потрібну інформацію та користуватися додатком без зусиль [6].

Мобільний додаток призначений для допомоги підприємцям, які працюють у ресторанах. Підсистема обліку замовлень має відповідати трьом основним

категоріям стандартів: візуальне наповнення, контент і структура. Візуальне наповнення є частиною дизайну додатку, який є привабливим для очей користувачів, що призводить до більш позитивного досвіду користування. Наповнення, яке буде якісною та інформативною, а також функції та зручні способи використання будуть включені до контенту. Структура програми повинна бути логічною та легкою для використання, щоб користувачі могли швидко знайти та використовувати потрібну інформацію.

На верхньому рівні схеми знаходиться Користувач, який взаємодіє з мобільним додатком. Наступний рівень, позначений як Мобільний додаток, розділяється на дві основні компоненти: Відображення та Бізнес логіка. У межах підсистеми відображення виділяються Користувацький інтерфейс, який містить усі елементи, з якими користувач безпосередньо взаємодіє, та Логіка відображення, що відповідає за правильне відображення даних і забезпечує інтуїтивну взаємодію користувача з додатком. Підсистема бізнес логіки включає Підсистему обліку запасів, яка займається управлінням та контролем запасів, і Підсистему обліку замовлень, що відповідає за облік та адміністрування замовлень користувачів. Нижче розташована секція Інформація, яка складається з трьох ключових елементів: Доступ до даних, що забезпечує механізми доступу до баз даних та іншої інформації, Утиліти, які є допоміжними інструментами та програмами, що підтримують функціонування додатка, і Сервіси, які включають додаткові функції та послуги, необхідні для роботи додатка. На найнижчому рівні знаходиться компонент Дані та кеш, що забезпечує збереження і обробку даних, а також кешування для прискорення доступу до них. Кожен з цих шарів має своє специфічне призначення і функціює у взаємодії з іншими елементами системи, забезпечуючи цілісне та ефективне функціонування мобільного додатка.

Таблиця 3.1 містить додаткову інформацію про вимоги до додатку підтримки ведення ресторанного бізнесу. Вона детально розкриває характеристики та важливість кожної вимоги до візуального наповнення, контенту та структури додатку.

Таблиця 3.1 – Вимоги до додатку

№	Категорія	Назва	Опис
1	Структура	Хедер	На кожній сторінці у верхній частині повинен знаходитися хедер із назвою вашого бізнесу та логотипом додатку.
2		Навігаційне меню	Навігаційне меню представляє з себе перелік кнопо, які ведуть на потрібні користувачу сторінки.
3		Основний вміст	Блок на екрані, що займає більшу частину екрану та має інформаційне наповнення відповідно до тематики.
5	Візуальне наповнення	Гарна читабельність шрифтів	Для зручності використання застосунку потрібно використовувати шрифти які комфортно читати. Адже непридатні для читання шрифти ускладнюють зручність перегляду.
6		Палітра кольорів	При виборі палітри кольорів потрібно переконатися, що вони мають високий рівень доступності, буде легко сприйматись всіма користувачами.
7		Адаптивна верстка	При реалізації адаптивної верстки додатку будь-який користувач матиме можливість використовувати застосунок на різних телефонах та планшетах
8		Єдиний стиль	Для зручного використання також важливо підтримувати єдиний стиль для оформлення блоків з інформацією, картинок, іконок та іншого.
9	Контент	Вдалість використання	Текст повинен бути вдалим, потрібно підкріплювати його ілюстраціями.
10		Оформлення тексту	Потрібно приділити увагу оформленню тексту, адже це впливає на те, як звичайний користувач буде сприймати ту чи іншу інформацію.

Підтримка для ведення ресторанного бізнесу. Підсистема обліку замовлень складається з двох основних компонентів: перед авторизацією та після авторизацією. Цей розподіл дозволяє користувачам отримувати доступ до різних функціональних модулів і сторінок відповідно до статусу користувача.

Щоб розпочати подальшу роботу з додатком, необхідно пройти реєстрацію або авторизацію.

Користувач отримує додаткові можливості та привілеї після авторизації. Він може брати участь у виборі свого бізнесу та отримати доступ до персоналізованої інформації, такої як налаштування профілю, історія замовлень тощо. Завдяки цьому розподілу можна забезпечити відповідний рівень функціональності та доступу до інформації в залежності від потреб користувачів, а також створити середовище для користування, яке є зручним і адаптованим до їхніх потреб [7].

Перелік функціональних можливостей для незареєстрованого користувача:

- реєстрація, або авторизація;

Перелік функціональних можливостей для зареєстрованого користувача:

- вибор бізнесу;
- зробити нове замовлення;
- переглядати історію замовлень;
- промокоди;
- робітники;
- аналітика;
- налаштування.

Розглянемо детальніше, які модулі доступні для кожного типу акаунту в таблиці 3.2.

Таблиця 3.2 – Таблиця рівнів доступу до сторінок додатку

№	Сторінки	Гість	Користувач
1	«Вибор бізнесу»	-	+
2	«Історія замовлень»	–	+
3	«Зробити нове замовлення»	-	+
4	«Промокоди»	-	+
5	«Робітники	-	+

6	«Аналітика»	-	+
7	«Налаштування»	–	+
8	«Авторизація»	+	-

3.2 Програмна реалізація

Програмна реалізація додатку підтримки ведення ресторанного бізнесу виконана за допомогою JavaScript, а саме фреймворку React Native.

Створення проекту на фреймворку React Native може бути розділено на кілька етапів. Нижче наведено опис основних кроків розробки додатку.

1. Встановлення React Native:

Початковий крок – встановлення фреймворку React Native локально. Для цього можна використовувати команду npx react-native init MyReactNativeApp. Після успішної установки можна створити новий проект.

2. Створення маршрутів:

React Native використовує маршрути для визначення, який код виконувати при запитах до вашого додатку. Маршрути знаходяться в папці navigation.

Приклад маршрутів:

```
const AuthStack = createNativeStackNavigator();
export const AuthRouter: FC = () => {
  return (
    <AuthStack.Navigator
      initialRouteName="Introduction"
      screenOptions={{ headerShown: false, animation:
'slide_from_right' }}>
      <AuthStack.Screen name={'Introduction'} component={Introduction}
/>
      <AuthStack.Screen name={'SignIn'} component={SignIn} />
      <AuthStack.Screen name={'ForgotPassword'}
component={ForgotPassword} />
      <AuthStack.Screen name={'ResetPassword'}
component={ResetPassword} />
    </AuthStack.Navigator>
  );
};
```

3. Створення контролерів:

Контролери в React Native використовуються для групування логіки, пов'язаної з обробкою запитів. У контролері описуємо методи, які відповідають за обробку різних запитів. Наприклад:

```
export const loginOwnBusiness = data => {
  return        axiosInstance.post(`/business-api/business/login`,
data);
};

export const checkPromo = data => {
  return       axiosInstance.post(`/business-api/promocode/check`,
data);
};
export const deleteOwnBusiness = data => {
  return axiosInstance.delete(`/business-api/business/delete`, {
    data: { businessId: data },
  });
};

export const getBusinessInventory = data => {
  return           axiosInstance.get(`/business-api/inventory/get-
all/${data}`);
```

4. Використання структури та логіки:

React Native використовує структуру та логіку для відображення сторінок вашого додатку. Вони відображають загальну структуру сторінки, та відповідають за вміст конкретних сторінок. Вони знаходяться в директорії screens. Наприклад:

```
<KeyboardAware>
      <TouchableWithoutFeedback onPress={onPressDismiss}>
        <View style={styles.container}>
          <Divider height={60} />

          <View style={styles.logoWrapper}>
            <Icon name="logo" />
          </View>
```

Та приклад функціонування:

```
const dispatch = useDispatch();
const navigation = useNavigation<any>();
const { t } = useTranslation();

const [stage, setStage] = useState(true);
const [email, setEmail] = useState('');
```

3.3 Використання програмного додатку

Додаток підтримки ведення ресторанного бізнесу. Підсистема обліку замовлень складається з основної частини: клієнтської.

Клієнтська частина додатку призначена для перегляду інформації та доступна всім користувачам після авторизації. Тут користувачі можуть створити свій бізнес, переглянути історію замовлень, створити нове замовлення. Крім того, користувачі можуть користуватися промокодами, налаштовувати робітників, дивитися аналітику та користуватися налаштуваннями [8].

Для обліку замовлень спершу потрібно авторизуватися (рис. 3.3)

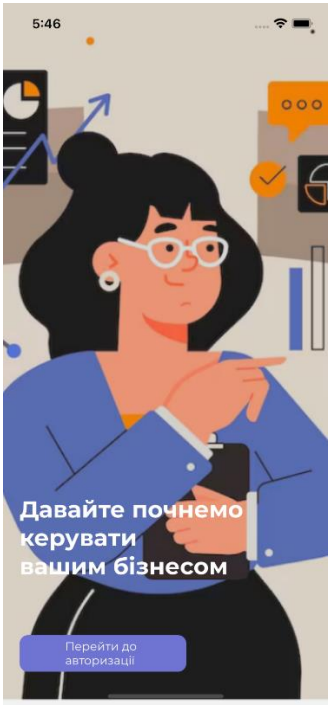

Рисунок 3.3 – Вікно авторизації у мобільному додатку

Після натискання на кнопку «Перейти до авторизації», відкриється сторінка реєстрації або авторизації.(рис. 3.4 та 3.5).

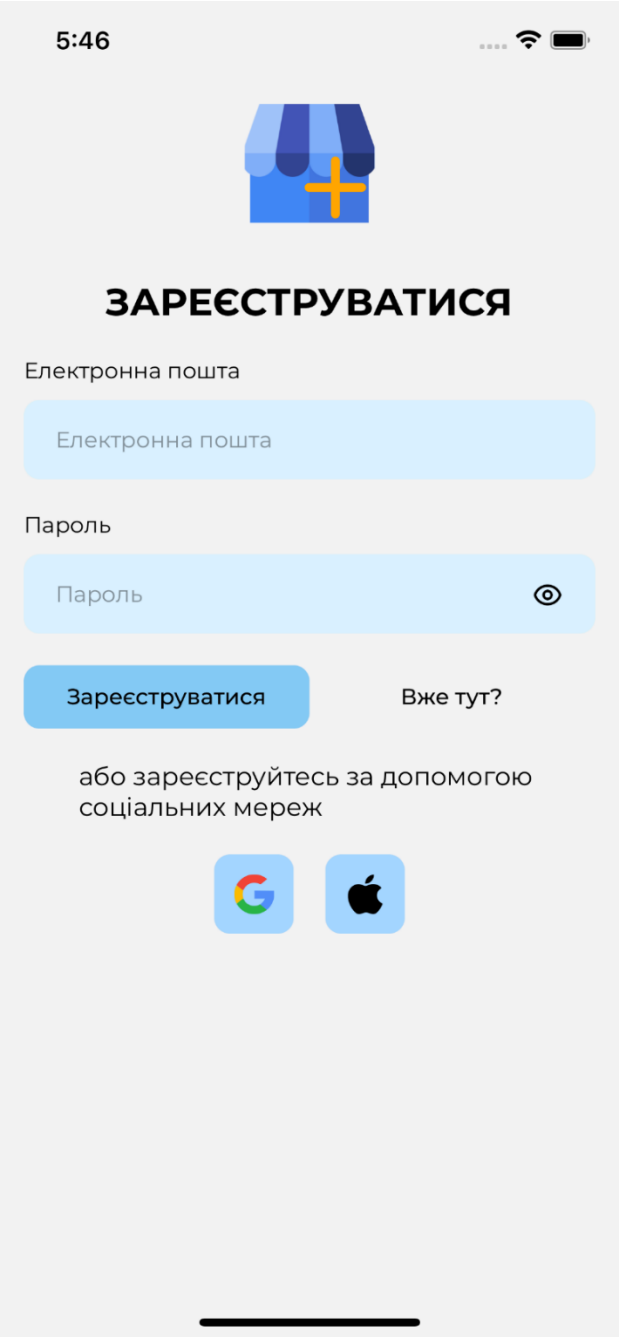

Рисунок 3.4 – Сторінка реєстрації

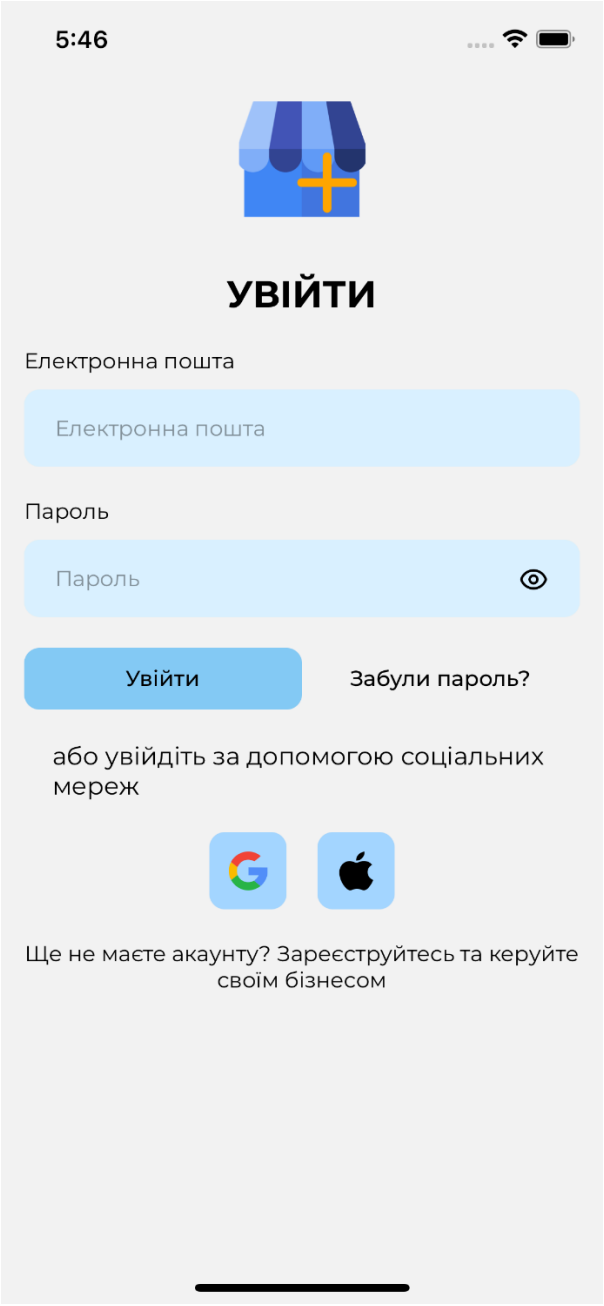

Рисунок 3.5 – Сторінка авторизації

Після успішної авторизації вам запропонується створити ваш бізнес. (рис 3.6)

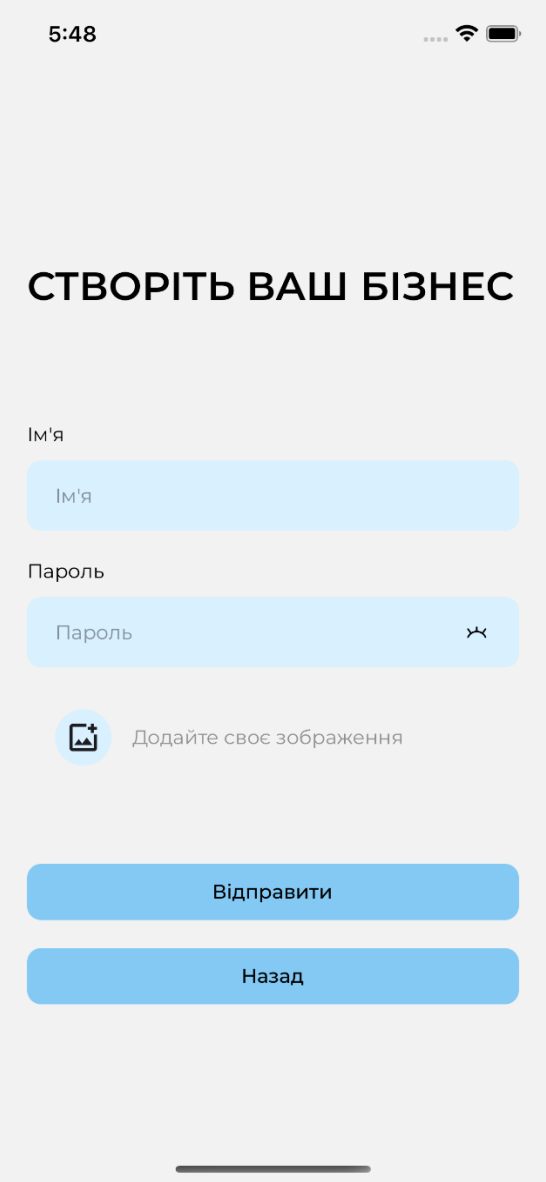

Рисунок 3.6 – Створення бізнесу

Коли ви створили потрібний бізнес, побачите екран с вибором бізнесів. (рис. 3.7)

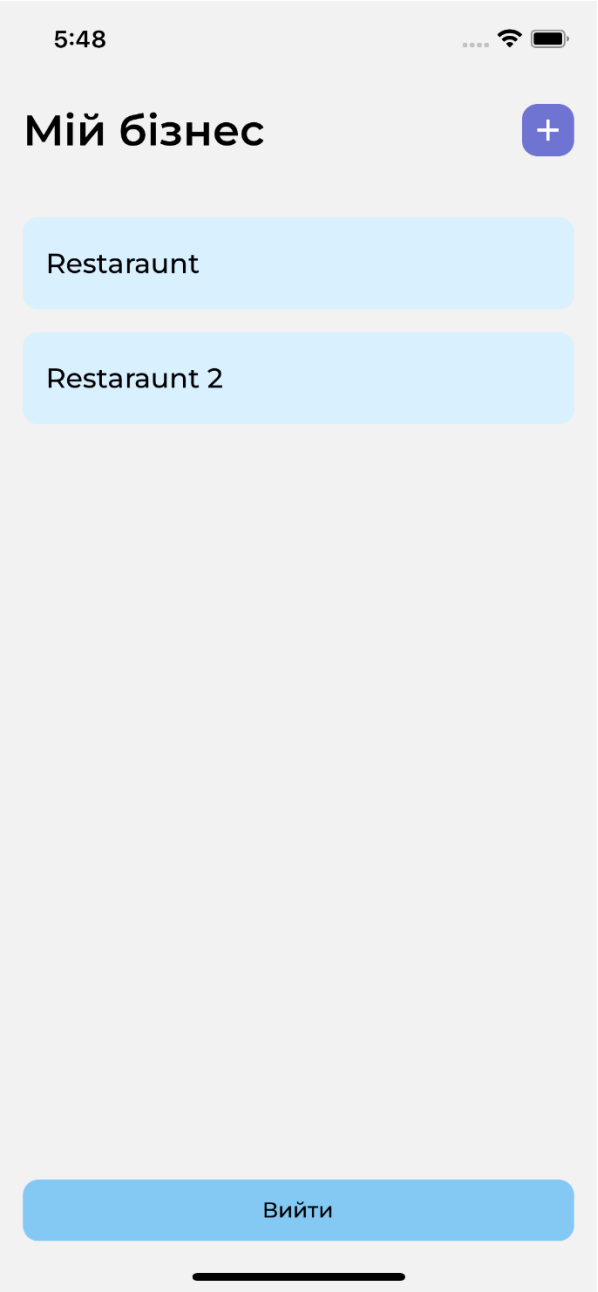

Рисунок 3.7 – Екран вибору бізнесів

Після вибору бізнесу з'являється увесь фунціонал, який ви можете побачити навігаційне меню (рис. 3.8)

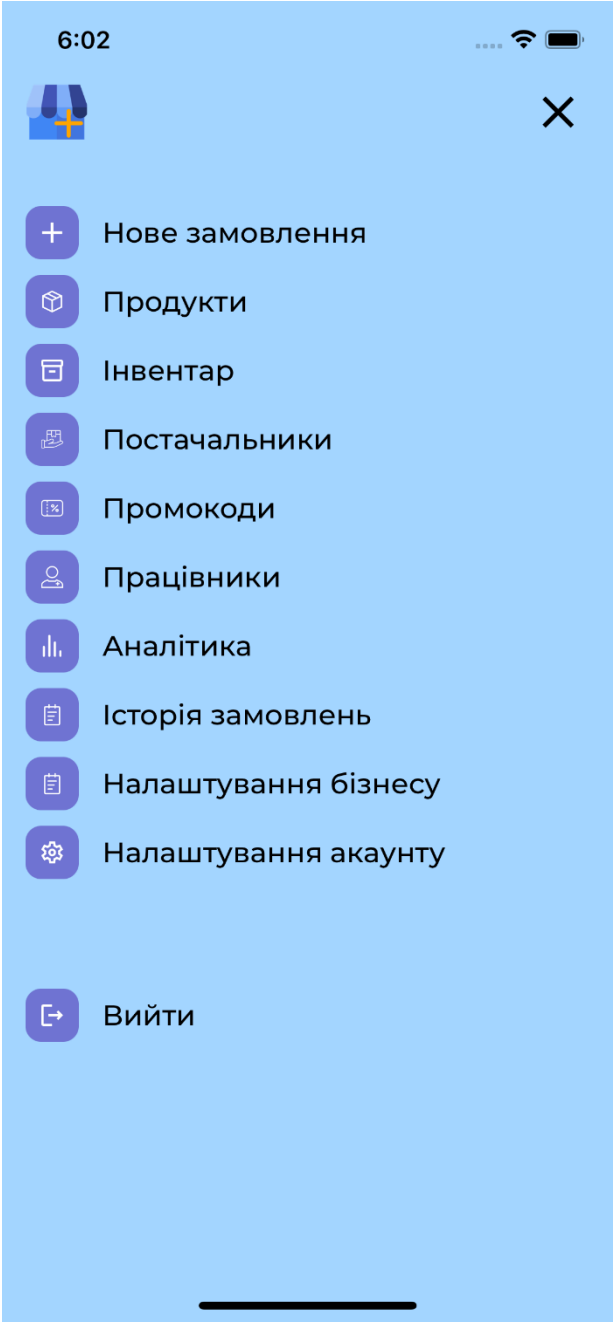

Рисунок 3.8 – Навігаційне меню

Меню включаю список з всього функціоналу додатку, перший з якого «Нове замовлення», де можна вибрати категорії продуктів, додати до замовлення потрібні та оформити його (рис 3.9)

Рисунок 3.9 – Екран з новим замовленням

Після того як натискання кнопки «Оформити замовлення» воно додається до екрану «Ваші позиції» (рис 3.10)

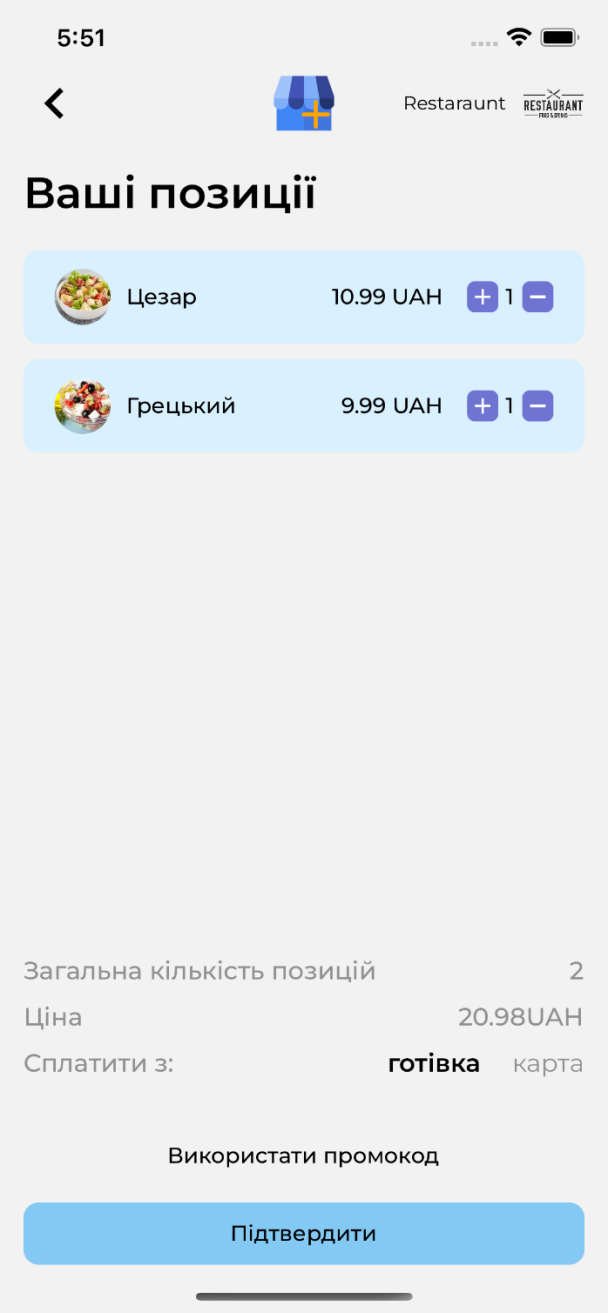

Рисунок 3.10 – Екран «Ваші позиції»

Також тут можно використати промокод, якщо він існує (рис. 3.11)

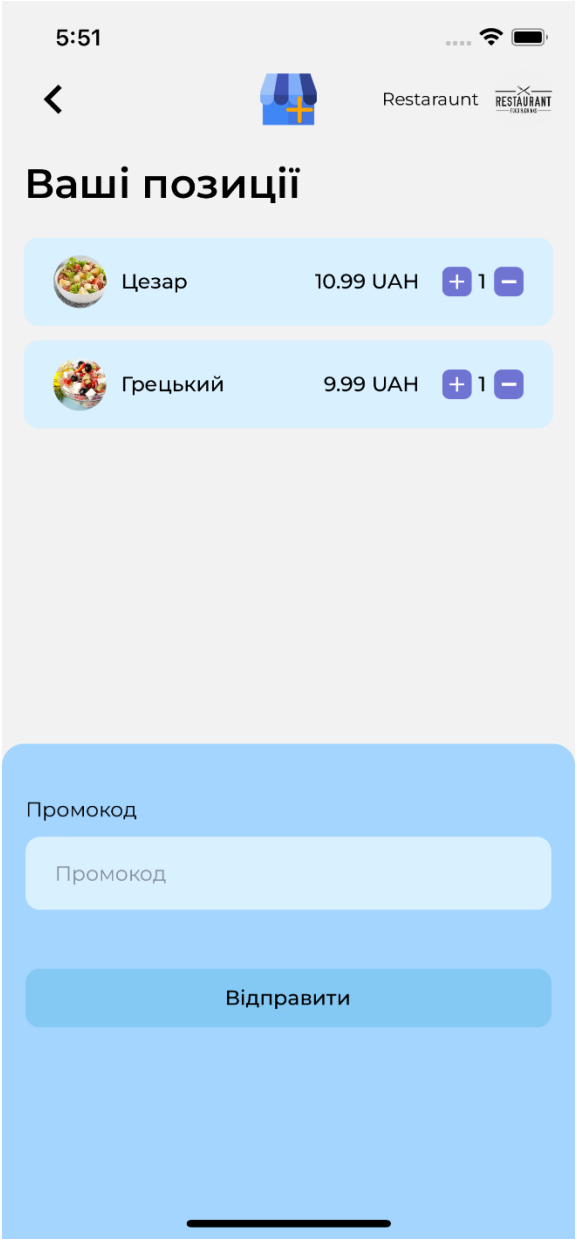

Рисунок 3.11 – Екран для використання промокоду.

Промокоди додаються на екрані який можна вибрати в меню під назвою «Промокоди» (рис 3.12)

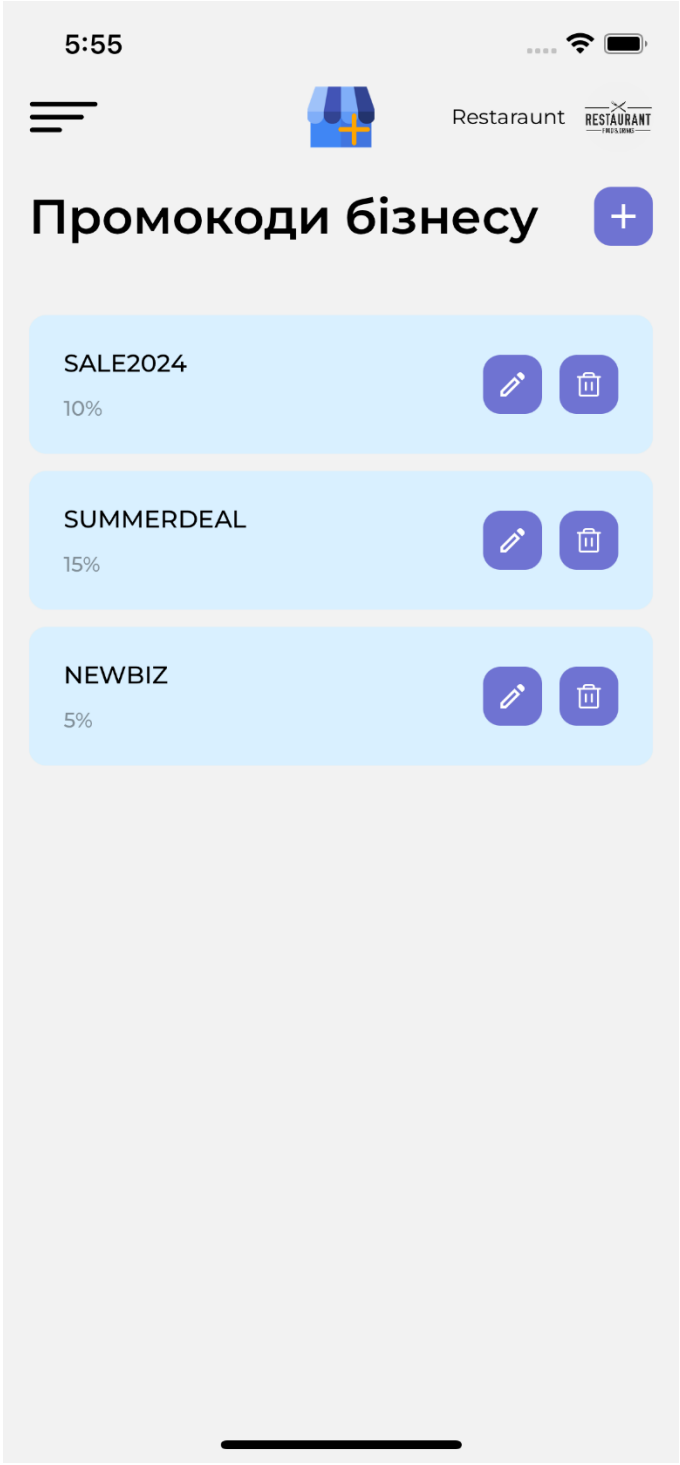

Рисунок 3.12 – Сторінка «Промокоди»

Користувач тут може додати новий, видалити або редагувати промокоди (рис 3.13)

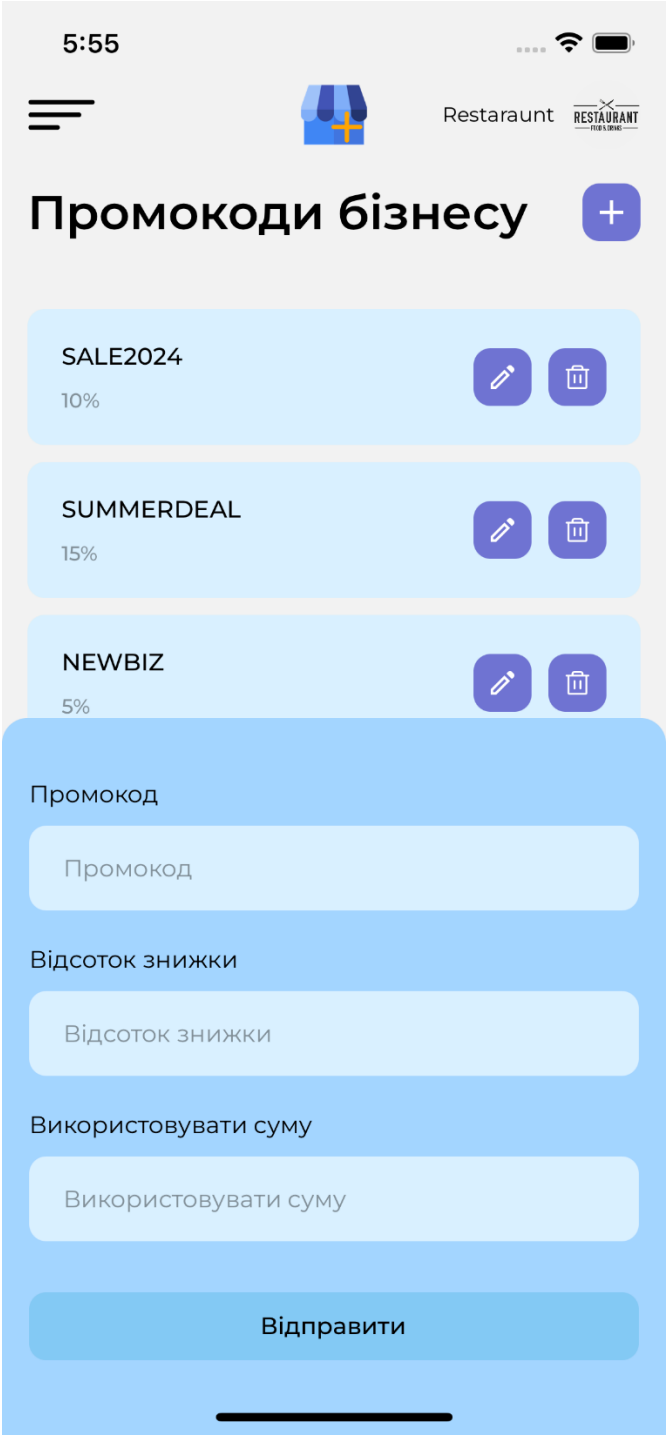

Рисунок 3.13 – Додати новий промокод

Після того як користувач оформив замовлення воно додається до Історії замовлень (рис. 3.14)

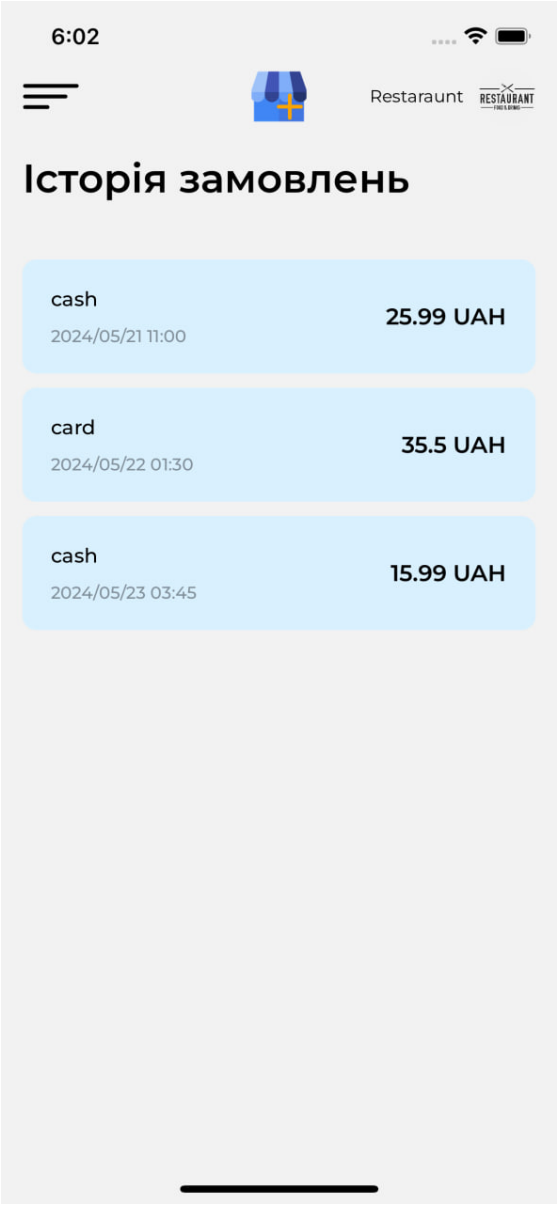

Рисунок 3.14 – Історія замовлень

Також власник бізнесу може дивитися аналітику доходів та витрат за тиждень. (рис 3.15)

Рисунок 3.15 – Аналітика

Користувач може налаштовувати вже створений бізнес, змінювати ім'я, валюту та зображення. (рис. 3.16)

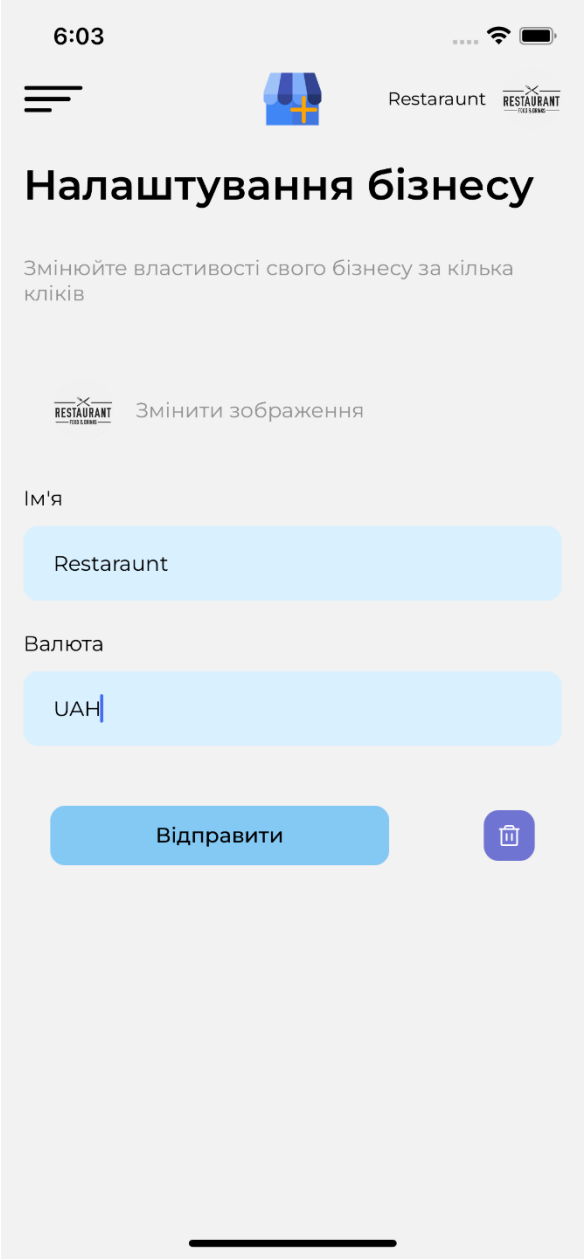

Рисунок 3.16 – Налаштування бізнесу

Окрім бізнесу є можливість налаштовувати і власник профіль (рис 3.17).

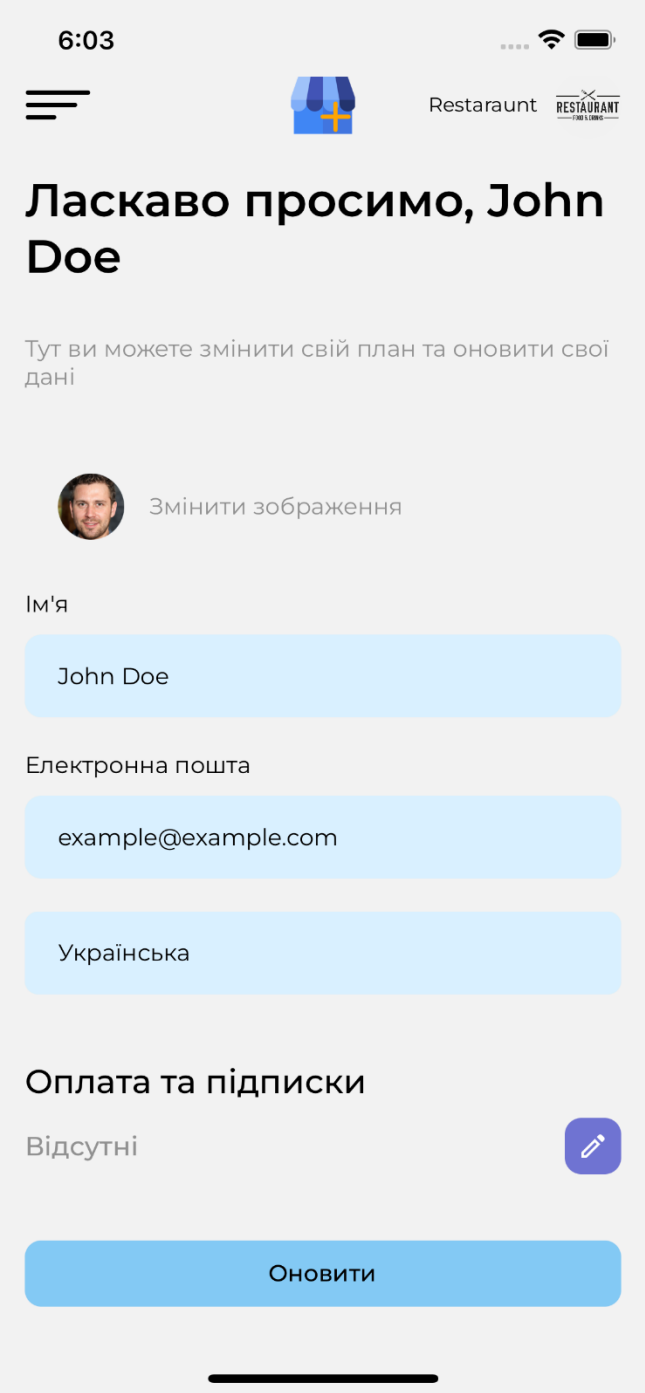

Рисунок 3.17 – Налаштування профілю

Додатково у формі налаштування профілю задано різні підписки, від котрих залежить можливості користувача (рис 3.18)

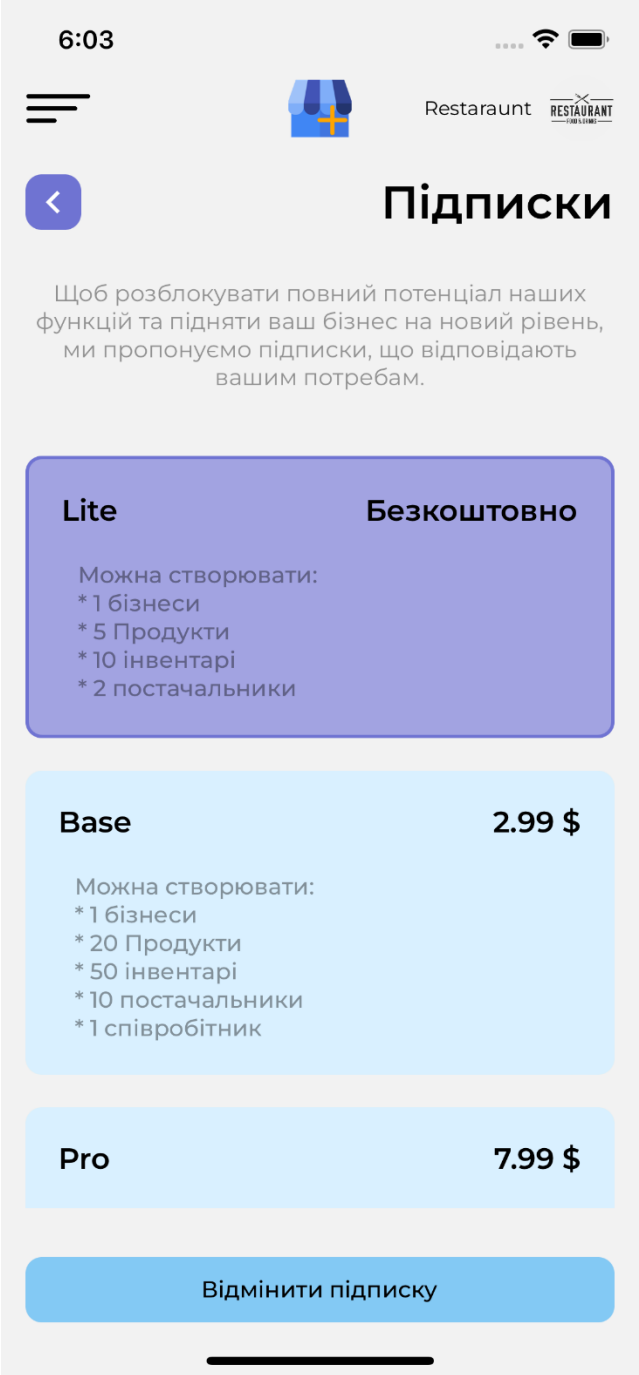

Рисунок 3.18 – Вибір підписки

ВИСНОВКИ

Під час виконання кваліфікаційної роботи бакалавра було виконано аналіз предметної області розроблення мобільних додатків для оформлення замовлень для ресторанного бізнесу та розроблено прототип мобільного додатку з підсистемою обліку замовлень, яка може допомогти ресторанам покращити ефективність ведення обліку замовлень та управління бізнесом, що в свою чергу призведе до покращення обслуговування клієнтів та їхньої задоволеності.

Підсистема обліку замовлень є цінним інструментом, який може допомогти ресторанам покращити свою операційну діяльність та підвищити задоволеність клієнтів. Додаток буде простим у використанні, матиме широкий спектр функцій та буде інтегруватися з іншими ресторанними системами. Розробка мобільного додатку буде складатися з декількох етапів, включаючи аналіз та проектування, розробку, тестування та впровадження. Очікується, що мобільний додаток матиме значний позитивний вплив на роботу ресторанів.

Практичні результати: проведено дослідження актуальних тенденцій у ресторанній галузі та проаналізовано сучасні мобільні додатки для ресторанного бізнесу, визначено ключові функції та можливості підсистеми обліку замовлень, розроблено прототип мобільного додатку з підсистемою обліку замовлень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Журнал "Mobile Apps for Restaurant Business: Trends and Innovations", [Електронний ресурс] – Режим доступу до ресурсу: http://www.restauranttechinsights.com/mobile-apps-for-restaurant-business-trends-and-innovations (дата звернення: 01.04.2023)

2. "Restaurant Order Management Systems: A Comprehensive Guide", Whitepaper by XYZ Consulting Group, 2022

3. Стаття "The Impact of Mobile Apps on Order Management in Restaurants", [Електронний ресурс] – Режим доступу до ресурсу: https://www.restaurantbusinessonline.com/technology/impact-mobile-apps-order-management-restaurants (дата звернення: 15.06.2023)

4. "Enhancing Customer Experience Through Mobile Apps: A Case Study of Restaurant Order Systems", Research Paper by ABC University, 2021

5. "Digital Transformation in Restaurant Order Processing: Challenges and Opportunities", Conference Proceedings, International Conference on Hospitality Technology, 2023

6. Книга "Mobile Technologies in the Hospitality Industry: Best Practices and Case Studies", автор Назва Автора, видавництво, 2020

7. "Effective Use of Mobile Apps for Inventory and Order Management in Restaurants", Industry Report by DEF Research Firm, 2022

8. "Optimizing Order Tracking and Delivery with Mobile Applications: Insights from Restaurant Industry Leaders", Panel Discussion Summary, National Restaurant Association Conference, 2023

9. Zanella, A., Foschini, L., Cavalieri, S., & Parisi, G. (2021). A machine learning approach for demand forecasting and staff scheduling in restaurants. International Journal of Hospitality Management, 97, 103062. [https://www.sciencedirect.com/science/article/pii/S2212827119301568]

10. Wu, D., Li, Z., & Wang, H. (2020). Demand forecasting and staff scheduling for restaurants using deep reinforcement learning. IEEE Transactions on Automation Science and Engineering, 17(3), 1476-1487. [https://ieeexplore.ieee.org/document/9066984]

11. Jiao, J., Wang, S., & Liu, F. (2022). [https://www.mdpi.com/2078-2489/14/11/597]

12. Long, Y., Zhang, Y., & He, Y. (2021). Research on the optimization of food delivery path based on location information and genetic algorithm. In 2021 3rd International Conference on Big Data and Software Engineering (ICBDSE), pp. 308-313. [https://ieeexplore.ieee.org/document/10082480/]

ДОДАТОК А. Технічне завдання

ТЕХНІЧНЕ ЗАВДАННЯ

на розробку додатку

«Мобільний додаток підтримки ведення ресторанного бізнесу. Підсистема обліку замовлень.»

ПОГОДЖЕНО:

Доцент кафедри інформаційних технологій

_____ Парфененко Ю.В.

Студент групи ІТ-03-2

_____ Федорченко О.В

Суми 2024

1. Призначення й мета створення додатку

1.1 Призначення додатку

Підсистема обліку замовлень у мобільному додатку підтримки ведення ресторанного бізнесу призначена для автоматизації процесу прийому, обробки та виконання замовлень від клієнтів.

1.2 Мета створення додатку

Мета створення додатку для ресторану включає ряд ключових аспектів, спрямованих на покращення діяльності закладу та задоволення потреб клієнтів. Перш за все, це збільшення ефективності роботи персоналу, що дозволить оптимізувати процеси прийняття та обробки замовлень, що є критично важливим у динамічному середовищі ресторанного бізнесу. Додаток також спрямований на покращення якості обслуговування клієнтів через швидке та точне оброблення їхніх замовлень, що сприятиме позитивному враженню та лояльності клієнтів.

Однією з ключових переваг є зниження ризику помилок при обробці замовлень, що допоможе уникнути непорозумінь та покращить загальний досвід клієнтів. Крім того, додаток забезпечить ресторан цінною інформацією про динаміку продажів та вподобання клієнтів, що дозволить керувати бізнесом на більш обґрунтованому рівні та адаптувати пропозиції до потреб аудиторії.

1.3 Цільова аудиторія

Цільовою аудиторією підсистеми обліку замовлень є персонал ресторану, який займається прийомом, обробкою та виконанням замовлень від клієнтів.

2 Вимоги до додатку

2.1 Вимоги до додатку в цілому

2.1.1 Вимоги до структури й функціонування додатку

Застосунок має бути доступним в основних магазинах на телефоні або планшеті. Додаток повинен складатися із взаємозалежних розділів із чітко розділеними функціями.

2.1.2 Вимоги до персоналу

- Від персоналу не вимагається значних технічних навичок.
- Персонал повинен володіти загальними навичками роботи з мобільними пристроями та стандартними мобільними операційними системами (Android, iOS).
- Персонал повинен бути ознайомлений з інтерфейсом мобільного додатку та його основними функціональними можливостями.

2.1.3 Вимоги до збереження інформації

Уся інформація надана у додатку буде зберігатися у базі даних реалізованій засобами системи управління базами даних MongoDB.

2.1.4 Вимоги до розмежування доступу

Мобільний додаток має бути загальнодоступним для всіх користувачів. Користувачі зможуть завантажити його з магазинів мобільних додатків (App Store, Google Play).

Для доступу до адміністративної панелі мобільного додатку буде потрібна авторизація та аутентифікація. Користувачі-адміністратори зможуть отримати доступ до адміністративної панелі за допомогою унікального логіна та пароля.

Всі дані користувачів, що зберігаються в мобільному додатку, будуть захищені за допомогою сучасних методів криптографії. Доступ до цих даних буде мати лише уповноважений персонал.

2.2 Структура додатку

2.2.1 Загальна інформація про структуру додатку

Структура додатку являє собою набір сторінок, які також є пунктами головного меню.

Такими розділами є:

Авторизація – на сторінці зображене поле для авторизації користувачів.

Головна – сторінка для вибору потрібних функції.

Замовлення – інформація про замовлення ресторану.

Історія замовлень – інформація про історію минулих замовлень.

Промокоди – інформація про існуючі промокоди для замовлень.

Аналітика – сторінка з аналітикою вашого бізнесу.

Налаштування аккаунту – налаштування аккаунту клієнта.

2.2.2 Навігація

Відповідно до бажаного замовником дизайну додатку, для навігації, у шапці буде створена система контент меню. Меню необхідне для швидкого переміщення користувача по усім доступним сторінкам. Меню буде відображатися на всіх сторінках, щоб відвідувач міг в будь-який момент часу перейти на будь-яку сторінку додатку.

2.2.3 Наповнення додатку (контент)

Для управління контентом додатку буде використана внутрішня система управління.

Заповнення та редагування контенту додатку має бути зроблено через панель керування, використовуючи інформацію з бази даних.

Всю інформацію для наповнення додатку має надавати ресторан який використовує додаток.

2.2.4 Дизайн та структура додатку

Мобільний додаток буде мати сучасний та елегантний дизайн, який буде приємним для сприйняття користувачів. Основними кольорами мобільного додатку будуть синій та білий, які відповідають кольоровій гамі бренду ресторану.

Інтерфейс мобільного додатку буде простим у використанні та інтуїтивно зрозумілим. Користувачі зможуть легко знайти те, що шукають, і виконувати необхідні дії без будь-яких проблем.

Розташування елементів на головній сторінці додатку схематично показано на рисунку А.1.

Рисунок А.1 – Схема головної сторінки

2.2.5 Система навігації (карта додатку)

Карта додатку зображена на рисунку А.2.

Рисунок А.2 – Карта додатку

2.3 Вимоги до функціонування системи

2.3.1 Потреби користувача

Потреби користувача, визначені на основі рішення замовника, представлені у таблиці А.1.

Таблиця А.1 – Потреби користувача

ID	**Потреби користувача**	**Джерело**
UN-01	Можливість зробити замовлення	Працівник
UN-02	Можливість переглянути історію замовлень	Працівник
UN-03	Перегляд можливих промокодів	Працівник
UN-04	Перегляд аналітики продажів	Працівник, власник
UN-05	Прийняття та обробка замовлень	Працівник
UN-06	Налаштування параметрів мобільного додатку	Власник

2.3.2 Функціональні вимоги

На основі потреб користувача були визначені такі функціональні вимоги:

-реєстрація та авторизація користувачів;

-перегляд замовлень;

-перегляд історії замовлень;

-аналітика;

-адміністрування інформації, видалення, зміну користувацької групи, надання користувацьких прав;

2.3.3 Системні вимоги

Даний розділ визначає, розподіляє та вказує на системні вимоги, визначені розробником. Їх перелік наведений в таблиці А.2.

Таблиця А.2 – Системні вимоги

ID	**Системні вимоги**	**Пріоритет**	**Опис**
SR-01	Наявність модуля замовлень	M	Надає можливість створити замовлення
SR-02	База даних із замовленнями	M	Зберігає інформацію про всі замовлення, включаючи дані про клієнтів, послуги, дату та час запису, статус замовлення тощо
SR-03	Модуль відстеження статусу замовлення	S	Надає можливість клієнтам відстежувати статус своїх замовлень
SR-04	Система нагадувань	S	Нагадує клієнтам про їхні майбутні записи та про необхідність підтвердження запису
SR-05	База даних з історією замовлень	M	Зберігає інформацію про минулі замовлення
SR-06	Модуль с історією замовлень	M	Надає можливість переглянути минулі замовлення
SR-07	Модуль с промокодами	M	Відповідає за відображення та створення промокодів

Умовні позначення в таблиці А.2:

Must have (M) – вимоги, які повинні бути реалізовані в системі;

Should have (S) – вимоги, які мають бути виконані, але вони можуть почекати своєї черги;

Could have (C) – вимоги, які можуть бути реалізовані, але вони не є центральною ціллю проекту.

2.4 Вимоги до видів забезпечення

2.4.1 Вимоги до інформаційного забезпечення

Реалізація додатку відбувається з використанням:

- React Native
- Node JS
- MongoDB

2.4.2 Вимоги до лінгвістичного забезпечення

Додаток має бути виконаний українською мовою.

2.4.3 Вимоги до програмного забезпечення

Програмне забезпечення клієнтської частини повинне задовольняти наступним вимогам:

Телефон або планшет:

- **Android:** 4.4 і вище.
- **iOS:** 8.0 і вище.

3 Склад і зміст робіт зі створення додатку

Докладний опис етапів роботи зі створення додатку наведено в таблиці А.3.

Таблиця А.3 – Етапи створення додатку

№	Склад і зміст робіт	Строк розробки (у робочих днях)
1	Постановка цілей необхідних для досягнення певного результату	1 день
2	Складання технічного завдання	3 дні
3	Підготовка прототипу	2 дні
4	Створення макету дизайну додатку	2 дні
5	Верстка	3 дні
6	Робота над модулями для додатку	2 дні
7	Робота з контентом	1 день
8	Перевірка працездатності додатку	1 день
9	Завершення роботи	1 день
	Загальна тривалість робіт	17 днів

4 Вимоги до складу й змісту робіт із введення додатку в експлуатацію

Для того, щоб wдодатком могли користуватися клієнти необхідно розмістити його у мережі Інтернет, тому необхідно залити його на в AppStore та PlayMarket. В магазини переноситься додаток і наповнення бази даних з подальшою їх доробкою.

ДОДАТОК Б

Планування робіт

Деталізація мети проекту методом SMART. Продуктом дипломного проекту є додаток доповненої реальності системи дизайну інтер'єру.

Результати деталізації методом SMART розміщені у табл. Б.1.

Таблиця Б.1 − Деталізація мети методом SMART

Specific (конкретна)	Створити додаток доповненої реальності системи дизайну інтер'єру, для візуалізації предметів дизайну в реальному просторі.
Measurable (вимірювана)	Результатом роботи проекту є оцінка замовника.
Achievable (досяжна)	Реалізації системи здійснюється за допомогою середовища розробки React Native, з використанням бекенд Node JS, та бібліотекою MongoDB.
Relevant (реалістична)	У наявності є всі необхідні технічні та програмні засоби. Розробники достатньо кваліфіковані для виконання поставлених задач.
Time-framed (обмежена у часі)	Ціль має часове обмеження. Робота повинна бути виконана у терміни, що були оговорені замовником проекту. Проект повинен бути виконаний згідно з календарним планом.

Планування змісту структури робіт. Основним інструментом для планування змісту структури робіт служить WBS діаграма − графічне подання згрупованих елементів проекту у вигляді пакета робіт, які ієрархічно пов'язані з продуктом проекту. Побудуємо структуру WBS, у якій детально опишемо роботи, які потрібно виконати на кожному етапі створення проекту. Виконаємо декомпозицію робіт для даного проекту. Діаграма WBS зображена на рис. Б.1.

Планування структури організації, для впровадження готового проекту (OBS). Після побудови WBS розробимо організаційну структуру виконавців OBS. Організаційна структура проекту стосується тільки внутрішньої організаційної структури проекту і не стосується відносин проектних груп чи учасників з батьківськими організаціями. Діаграма OBS зображена на рис. Б.2. Список виконавців, що функціонують в проекті знаходиться в табл. Б.2.

Таблиця Б.2 − Виконавці проекту

Роль	Ім’я	Проектна роль
Розробник	Федорченко О.В	Виконує розробку основного функціоналу проекту та інтерфейс користувача
Тестувальник	Федорченко О.В	Відповідає за тестування функціоналу та дизайну додатку, перевірку моделі на адекватність.
Косультант проекту	Парфененко Ю.В	Формує завдання на розробку проекту.
Менеджер проекту	Федорченко О.В	Відповідає за виконання термінів, розподіл ресурсів та завдань між учасниками. Виконує збір та аналіз даних.

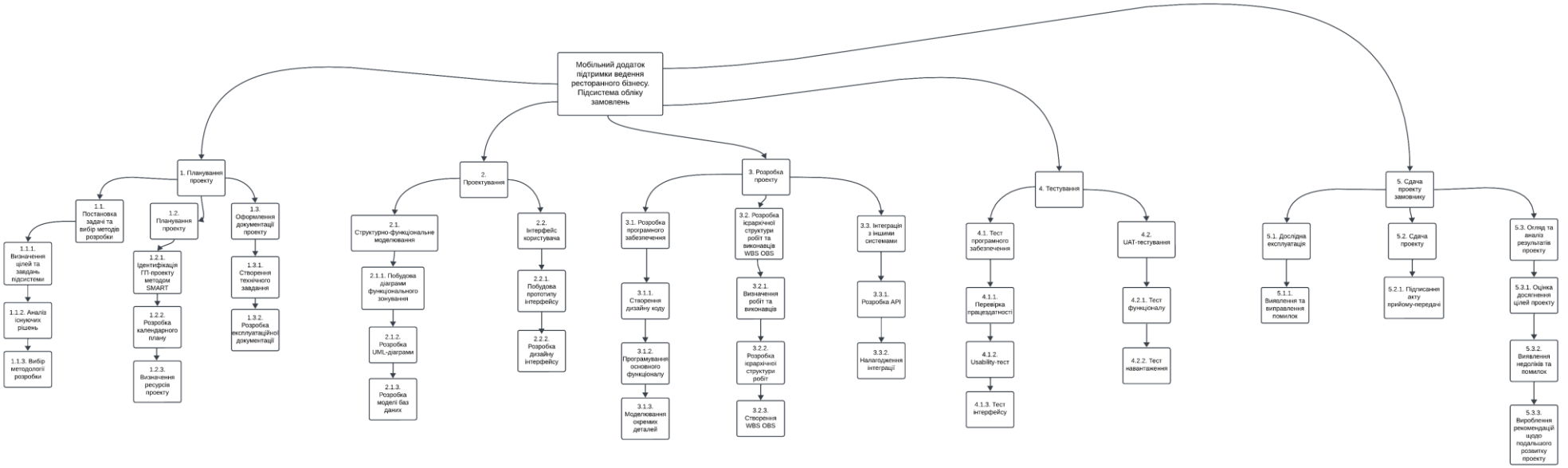

Рисунок Б.1 − WBS. Структура робіт проекту

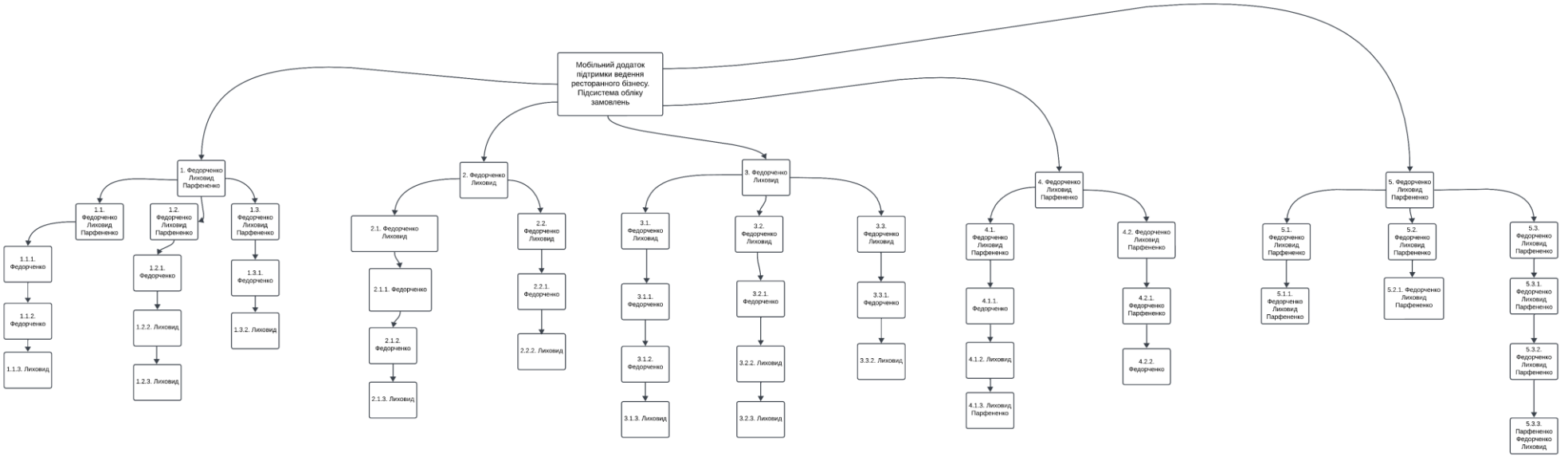

Рисунок Б.2 − Організаційна структура проекту (OBS)

Діаграма Ганта. Далі побудуємо календарний план виконання дипломного проекту. Найпоширеніший формат графіка в будь-якій галузі — діаграма Ганта. Цей графік дозволяє менеджерам проекту і всій команді розробників візуалізувати графіки часу і взаємозв'язок між окремими завданнями та етапами роботи над проектом. Тривалість виконання робіт зазначена в днях, але фактична тривалість виконання робіт приблизно дорівнює 2-3 години на день. Для того щоб мати реальне уявлення про тривалість виконання робіт з урахуванням обмеженості у використанні ресурсів, з урахуванням вихідних та святкових днів, побудовано календарний графік. Діаграма Ганта та список робіт діаграми Ганта зображені на рис. Б.3-Б.6.

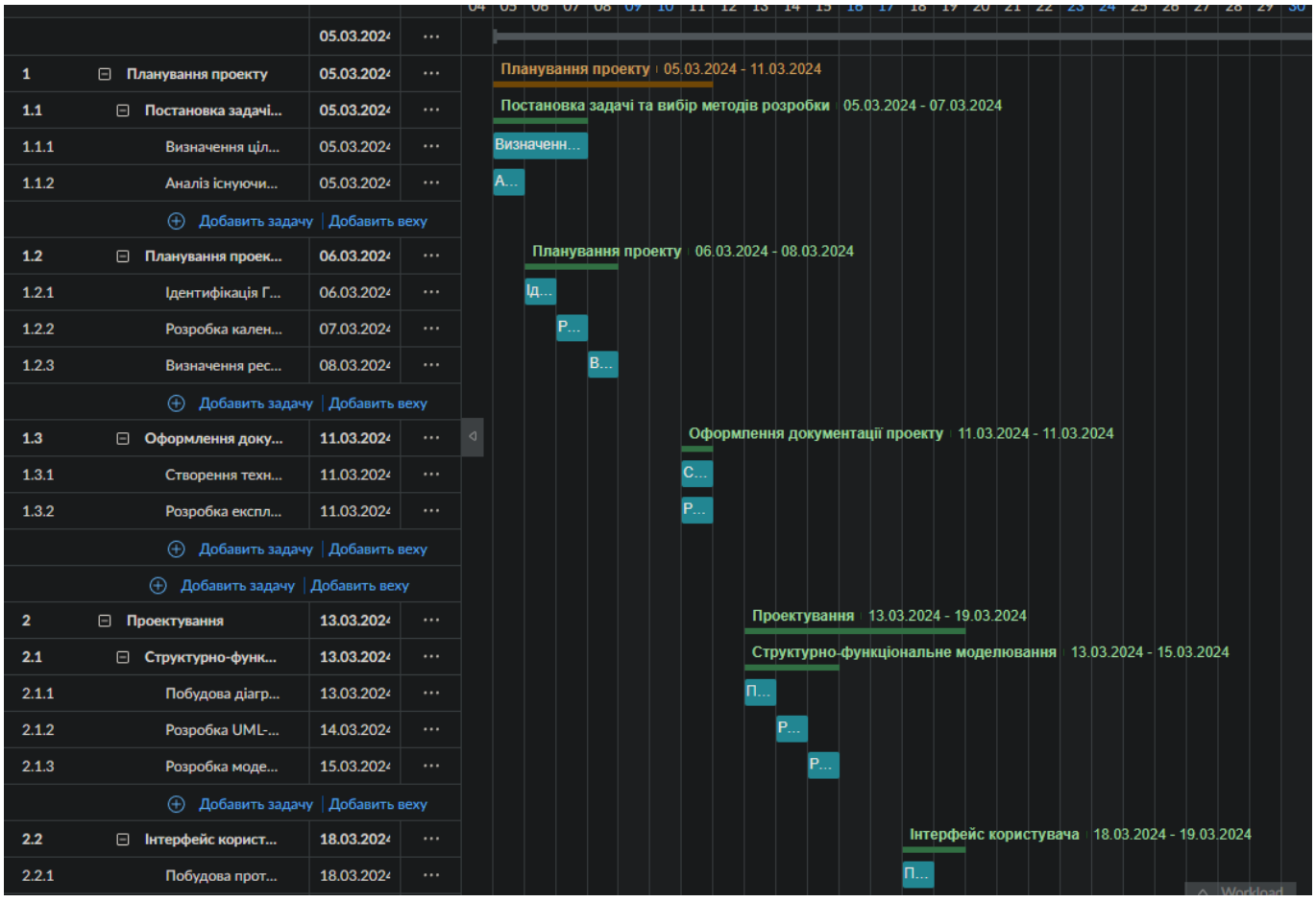

Рисунок Б.3 − Діаграма Ганта

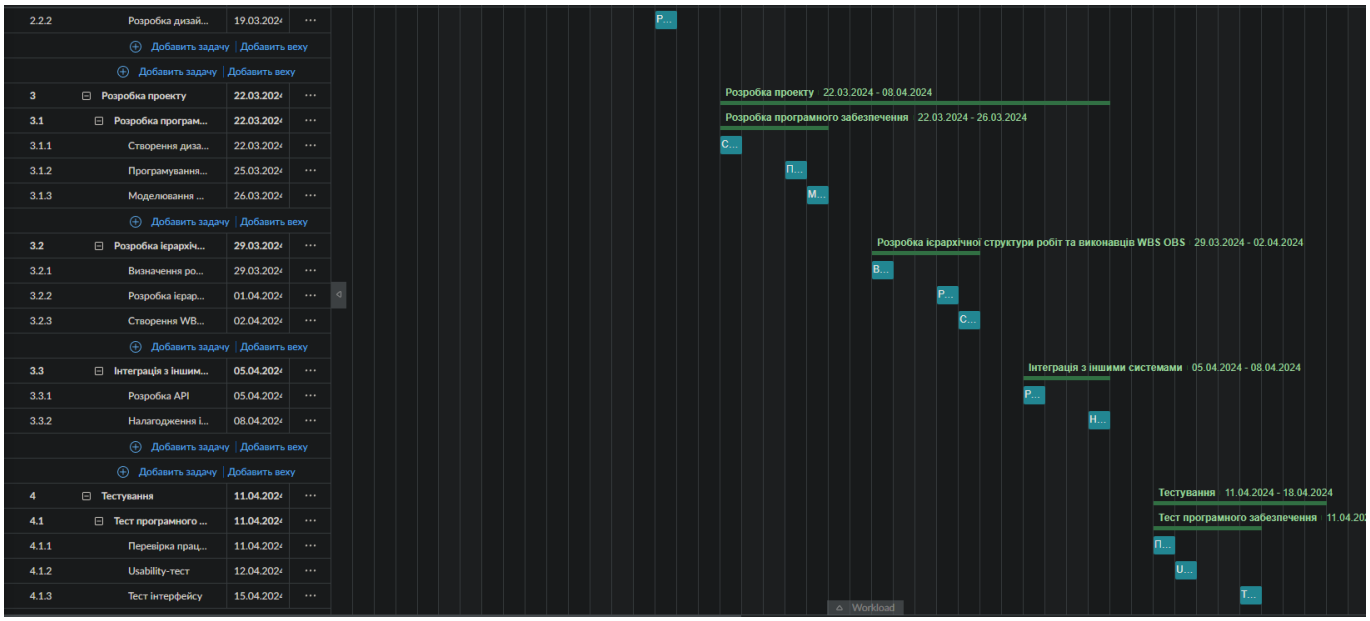

Рисунок Б.4 – Продовження діаграми Ганта

Рисунок Б.5 – Продовження діаграми Ганта

Аналіз ризиків. Виконаємо якісну і кількісну оцінку ризиків роботи. При якісній оцінці визначимо ризики, що потребують швидкого реагування. Така оцінка визначить ступінь важливості ризику і дозволить вибрати спосіб реагування. Кількісна оцінка ризиків буде виконана для більш повної ідентифікації ризиків та ступеня їхнього впливу на виконання проекту. Кількісна і якісна оцінка ризиків можуть використовуватися окремо або разом, залежно від наявного часу і бюджету, необхідності в кількісній або якісній оцінці ризиків. У табл. Б.5 знаходиться

класифікація ризиків за показниками ймовірності виникнення ризику та величині втрат.

Далі виконаємо планування реагування на ризики — це розробка методів і технологій зниження негативного впливу ризиків на проект. Визначимо ефективність розробки реагування на проект, визначимо чи будуть наслідки впливу ризику на проект позитивними або негативним. Оцінюємо ризики за показниками, що знаходяться в табл. Б.3. На основі оцінки будуємо матрицю ймовірності виникнення ризиків та впливу ризику, що зображена на рис. Б.7.

Таблиця Б.3 − Шкала оцінювання ймовірності виникнення та впливу ризику на виконання проекту

Оцінка	Ймовірність виникнення	Вплив ризику
1	Низька	Низький
2	Середня	Середній
3	Висока	Високий

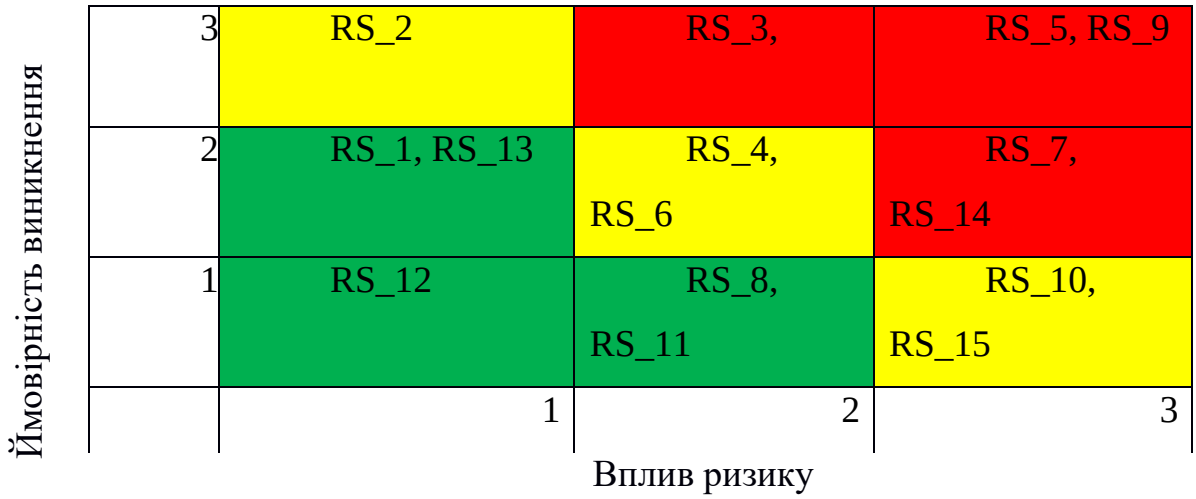

Рисунок Б.7 − Матриця ймовірності виникнення ризиків та впливу ризику

- зелений колір – прийнятні ризики;
- жовтий колір – виправданні ризики;
- червоний колір – недопустимі ризики.

На підставі отриманого значення індексу ризики класифікують: за рівнем ризику, що знаходиться в табл. А.4.

Таблиця Б.4 − Шкала оцінювання за рівнем ризику

№	Назва	Межі	Ризики, які входять(номера)
1	Прийнятні	1≤R≤2	1,8,11,12,13
2	Виправдані	3≤R≤4	2,4,6,10,15
3	Недопустимі	6≤R≤9	3,5,7,9,14

Таблиця Б.5 − Оцінка ймовірності виникнення, величини витрат та індексу ризику

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	План А	Тип стратегії реагування	План Б
RS_1	Відкритий	Недооцінка складності розробки	Середня	Високий	6	1. Провести ретельний аналіз вимог до підсистеми 2. Розбити розробку на етапи з чітко визначеними завданнями 3. Залучити до розробки досвідчених фахівців.	Попередження	Переоцінити терміни розробки проекту, якщо це буде необхідно.
RS_2	Відкритий	Недоліки в роботі підсистеми	Висока	Високий	9	1. Ретельно тестувати підсистему на всіх етапах розробки 2. Залучити користувачів до тестування підсистеми 3. Використовувати методи виявлення та виправлення помилок.	Попередження	Використовувати резервні методи обліку замовлень на випадок збоїв у роботі підсистеми.

Продовження таблиці Б.5

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	План А	Тип стратегії реагування	План Б
RS_3	Відкритий	Несумісність з існуючими системами	Середня	Середній	4	1. Провести аналіз існуючих систем 2. Розробити інтерфейси для взаємодії з існуючими системами 3. Провести тестування сумісності підсистеми з існуючими системами.	Попередження	Розробити план ручного перенесення даних з існуючих систем, якщо це буде необхідно.
RS_4	Відкритий	Недостатня кібербезпека	Низька	Високий	3	1. Використовувати надійні методи шифрування даних 2. Застосовувати протоколи безпечного доступу до даних 3. Регулярно оновлювати програмне забезпечення підсистеми.	Попередження	Відновити дані з резервних копій у разі кібератаки.
RS_5	Відкритий	Невідповідність	Середня	Середній	4	1. Провести дослідження	Попередження	Внести зміни до

		очікуванням користувачів				потреб користувачів 2. Залучити користувачів до розробки підсистеми 3. Збирати та аналізувати відгуки користувачів.		підсистеми на основі відгуків користувачів.

Продовження таблиці Б.5

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	План А	Тип стратегії реагування	План Б
RS_6	Відкритий	Недотримання законодавства	Низька	Високий	3	1. Ознайомитися з вимогами законодавства, що стосуються обробки даних. 2. Впровадити заходи для дотримання законодавства 3. Регулярно перевіряти відповідність підсистеми законодавству.	Попередження	Зупинити роботу підсистеми, якщо буде виявлено невідповідність законодавству.
RS_7	Відкритий	Недостатня масштабованість	Низька	Низька	2	1. Провести аналіз потенційного зростання користувачів 2. Використовувати масштабовані технології. 3. Провести навантажувальні	Попередження	Внести зміни до інфраструктури для підтримки зростання користувачів, якщо це буде необхідно.

						тестування підсистеми.		
RS_8	Теоретичний	Зміна ринкових умов	Низька	Високий	3	1. Регулярно моніторити ринкові умови 2. Зберігати гнучкість у плануванні розвитку підсистеми 3. Бути готовим до внесення змін до підсистеми	Попередження	-

Додаток В

КОД РЕАЛІЗАЗІЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

```
```import React, { useEffect, useState } from 'react';
import {
  FlatList,
  Keyboard,
  Platform,
  SafeAreaView,
  StyleSheet,
  Text,
  TouchableOpacity,
  TouchableWithoutFeedback,
  View,
} from 'react-native';
import {
  ActionButton,
  BottomSheet,
  Button,
  Divider,
  Header,
  Icon,
  Input,
} from '../../components';
import { useNavigation } from '@react-navigation/native';
import { ProductsProps } from './types';
import { ProductCard } from './components/ProductCard';
import { useDispatch, useSelector } from 'react-redux';
import { deleteProduct, getProductsList } from '../../store/slices/products';
import { useTranslation } from 'react-i18next';
import { Toast } from 'react-native-toast-message/lib/src/Toast';
import { TOASTS } from '../../i18n/toasts';
import i18n from '../../i18n';

export const Products: React.FC<ProductsProps> = () => {
  const navigation = useNavigation<any>();
  const dispatch = useDispatch();

  const { products } = useSelector((store: any) => store.products);
```
```

```
const { profile } = useSelector((store: any) => store.auth);

const [productList, setProductList] = useState<any>(null);
const { t } = useTranslation();

useEffect(() => {
  setProductList(products);
}, [products]);

return (
  <SafeAreaView style={styles.area}>
    <View style={styles.container}>
      <Header />
      <Divider height={20} />
      <View style={styles.headerWrapper}>
        <Text style={styles.titleText}>{t('products')}</Text>
        <ActionButton
          iconName="plus"
          onPress={() => {
            if (true) {
              navigation.navigate('CreateProduct');
            } else {
              //TOAST
            }
          }}
          size="large"
        />
      </View>

      <Divider height={20} />

      <FlatList
        showsVerticalScrollIndicator={false}
        data={productList ?? []}
        renderItem={({ item }) => (
          <ProductCard
            title={item?.name}
            image={item?.image ?? ''}
            price={item?.price}
            onEdit={() => {
              navigation.navigate('CreateProduct', { ...item, edit: true });
            }}
```

```
              onDelete={() => {
                dispatch(
                  deleteProduct(
                    item?.id,
                    () => {
                      Toast.show({
                        text1: TOASTS[i18n.language].SUCCESS_DELETE_PRODUCT,
                      });
                    },
                    (error: string) => {
                      Toast.show({
                        text1: TOASTS[i18n.language].ERROR,
                        text2:
                          TOASTS[i18n.language][error] ?? 'Unexpected error',
                        type: 'error',
                      });
                    },
                  ) as any,
                );
              }}
            />
          )}
          style={styles.list}
        />
      </View>
    </SafeAreaView>
  );
};

const styles = StyleSheet.create({
  headerWrapper: {
    flexDirection: 'row',
    justifyContent: 'space-between',
    alignItems: 'center',
  },
  createbuttonWrapper: { alignSelf: 'center' },
  area: { flex: 1 },
  container: { paddingHorizontal: 15 },
  buttonWrapper: {
    justifyContent: 'center',
    alignItems: 'center',
    width: '100%',
```

```
    marginTop: 10,
  },

  list: {
    marginTop: 20,
    height: '87%',
  },

  titleText: {
    fontFamily: Platform.OS === 'ios' ? 'Montserrat' : 'Montserrat-SemiBold',

    fontSize: 28,
    fontWeight: '600',
    color: '#000000',
  },
});

import React, { Fragment, useEffect, useState } from 'react';
import {
  FlatList,
  Keyboard,
  Platform,
  SafeAreaView,
  ScrollView,
  StyleSheet,
  Text,
  TouchableOpacity,
  TouchableWithoutFeedback,
  View,
} from 'react-native';
import {
  BottomSheet,
  Button,
  Divider,
  Header,
  Icon,
  ImageInput,
  Input,
  KeyboardAware,
} from '../../components';
import { useNavigation, useRoute } from '@react-navigation/native';
import { CreateProductProps } from './types';
```

```
import { useDispatch, useSelector } from 'react-redux';
import {
  createProduct,
  getProductsList,
  updateProducts,
} from '../../store/slices/products';
import { useTranslation } from 'react-i18next';
import { Toast } from 'react-native-toast-message/lib/src/Toast';
import { TOASTS } from '../../i18n/toasts';
import i18n from '../../i18n';

export const CreateProduct: React.FC<CreateProductProps> = () => {
  const navigation = useNavigation<any>();
  const dispatch = useDispatch();
  const { t } = useTranslation();

  const onPressDismiss = () => {
    Keyboard.dismiss();
  };

  const { params } = useRoute<any>();
  const { currentBusiness } = useSelector((store: any) => store.business);
  const { categories } = useSelector((store: any) => store.products);

  const [name, setName] = useState(params?.name ?? '');
  const [price, setPrice] = useState(params?.price ? params?.price + '' : '');
  const [self, setSelf] = useState(
    params?.selfPrice ? params?.selfPrice + '' : '',
  );
  const [category, setCategory] = useState<any>({
    id: params?.categoryId ?? '',
  });
  const [photo, setPhoto] = useState<any>('');
  const [imageUrl, setImageUrl] = useState<any>(params?.image ?? '');
  const [inventory, setInventory] = useState<any>([]);
  const [isValidForm, setIsValidForm] = useState({
    name: true,
    price: true,
    selfPrice: true,
    photo: true,
  });
```

```
const validateForm = (onSuccess: any) => {
  let isValid = true;

  if (name.length > 2 && name.length < 20) {
    setIsValidForm(prev => ({ ...prev, name: true }));
  } else {
    isValid = false;
    setIsValidForm(prev => ({ ...prev, name: false }));
  }
  if (Number(price) > 0) {
    setIsValidForm(prev => ({ ...prev, price: true }));
  } else {
    isValid = false;
    setIsValidForm(prev => ({ ...prev, price: false }));
  }
  if (Number(self) > 0) {
    setIsValidForm(prev => ({ ...prev, selfPrice: true }));
  } else {
    isValid = false;
    setIsValidForm(prev => ({ ...prev, selfPrice: false }));
  }
  if (photo?.filename) {
    setIsValidForm(prev => ({ ...prev, photo: true }));
  } else {
    isValid = false;
    setIsValidForm(prev => ({ ...prev, photo: false }));
  }

  if (isValid) {
    onSuccess();
  }
};

return (
  <SafeAreaView style={styles.area}>
    <KeyboardAware>
      <TouchableWithoutFeedback onPress={onPressDismiss}>
        <View style={styles.container}>
          <Header withGoBack />
          <Divider height={20} />
          <ImageInput
            onSelect={setPhoto}
```

```
  imageUrl={imageUrl}
  isValid={isValidForm.photo}
/>
<Divider height={20} />



<Input
  placeholder={t('name')}
  value={name}
  onChange={setName}
  isValid={isValidForm.name}
/>
<Divider height={20} />
<Input
  placeholder={t('price')}
  value={price}
  onChange={setPrice}
  isValid={isValidForm.price}
/>
<Divider height={20} />
<Input
  placeholder={t('selfPrice')}
  value={self}
  onChange={setSelf}
  isValid={isValidForm.selfPrice}
/>
<Divider height={20} />
<Text style={styles.labelText}>
  {t('itemsInventory') + ` (${inventory.length})`}
</Text>
<Divider height={15} />
<Button
  text={t('editInventory')}
  onPress={() => {
    navigation.navigate('PickInventory', {
      inventory,
      setInventory,
    });
  }}
/>
<Divider height={40} />
```

```
<Text style={styles.labelText}>{t('category')}</Text>
<Divider height={15} />
<View style={{ height: 35 }}>
  <ScrollView
    horizontal
    contentContainerStyle={styles.categoryScroll}>
    {categories.map((el: any) => (
      <Fragment key={el.id}>
        <TouchableOpacity
          onPress={() => {
            setCategory(el);
          }}
          style={[
            styles.categoryItem,
            {
              backgroundColor:
                el.id === category?.id ? '#384494' : '#D9F0FF',
            },
          ]}>
          <Text
            style={[
              styles.categoryText,
              {
                color: el.id === category?.id ? 'white' : 'black',
              },
            ]}>
            {el?.name}
          </Text>
        </TouchableOpacity>
        <Divider width={5} />
      </Fragment>
    ))}
  </ScrollView>
</View>
<Divider height={15} />
<Button
  text={t('editCategory')}
  onPress={() => {
    navigation.navigate('Categories');
  }}
/>
```

```
                <Divider height={45} />

                {/* <View style={styles.buttonWrapper}> */}
                <Button
                  text={params?.edit ? t('update') : t('submit')}
                  onPress={() => {
                    const formData = new FormData();
                    formData.append('name', name);
                    formData.append('price', price);
                    formData.append('selfPrice', self);
                    formData.append('businessId', currentBusiness?.id);
                    if (inventory.length) {
                      const resultArray = inventory.flatMap(({ id, total }: any) =>
                        Array.from({ length: total }, () => String(id)),
                      );
                      resultArray.forEach((el: any, index: number) => {
                        formData.append(`inventories[${index}]`, el);
                      });
                    }

                    category?.id && formData.append('categoryId', category?.id);

                    params?.id && formData.append('productId', params?.id);



                    photo.path &&
                      formData.append('image', {
                        name: photo.filename,
                        type: photo.mime ?? 'image/jpeg',
                        uri: photo.path,
                      } as any);
                    if (params?.edit) {
                      validateForm(() =>
                        dispatch(
                          updateProducts(
                            formData,
                            () => {
                              Toast.show({
                                text1:
TOASTS[i18n.language].SUCCESS_UPDATE_PRODUCT,
                              });
```

```
                            navigation.navigate('Products');
                          },
                          (error: string) => {
                            Toast.show({
                              text1: TOASTS[i18n.language].ERROR,
                              text2:
                                TOASTS[i18n.language][error] ??
                                'Unexpected error',
                              type: 'error',
                            });
                          },
                        ) as any,
                      ),
                    );
                  } else {
                    validateForm(() =>
                      dispatch(
                        createProduct(
                          formData,
                          () => {
                            Toast.show({
                              text1:
TOASTS[i18n.language].SUCCESS_UPDATE_PRODUCT,
                            });
                            navigation.navigate('Products');
                          },
                          (error: string) => {
                            Toast.show({
                              text1: TOASTS[i18n.language].ERROR,
                              text2:
                                TOASTS[i18n.language][error] ??
                                'Unexpected error',
                              type: 'error',
                            });
                          },
                        ) as any,
                      ),
                    );
                  }
                }}
              />
              <Divider height={20} />
```

```
            {/* </View> */}
          </View>
        </TouchableWithoutFeedback>
      </KeyboardAware>
    </SafeAreaView>
  );
};

const styles = StyleSheet.create({
  categoryText: {
    fontFamily: Platform.OS === 'ios' ? 'Montserrat' : 'Montserrat-Regular',
  },
  categoryItem: {
    paddingHorizontal: 15,
    paddingVertical: 5,
    borderRadius: 50,

    justifyContent: 'center',
    alignItems: 'center',
  },
  categoryScroll: {
    height: 35,
    width: '100%',
  },
  labelText: {
    fontFamily: Platform.OS === 'ios' ? 'Montserrat' : 'Montserrat-Regular',

    fontSize: 16,
  },
  createbuttonWrapper: { alignSelf: 'center' },
  area: { flex: 1 },
  container: { paddingHorizontal: 15, height: '100%' },
  buttonWrapper: {
    alignSelf: 'center',
    marginTop: 20,
    // position: 'absolute',
    // bottom: 10,
  },

  list: {
    marginTop: 20,
```

```
    height: '87%',
  },

  titleText: {
    fontFamily: Platform.OS === 'ios' ? 'Montserrat' : 'Montserrat-SemiBold',

    fontSize: 28,
    fontWeight: '600',
    color: '#000000',
  },
});

import React, { useEffect, useState } from 'react';
import {
  FlatList,
  Keyboard,
  Platform,
  SafeAreaView,
  StyleSheet,
  Text,
  TouchableOpacity,
  TouchableWithoutFeedback,
  View,
} from 'react-native';
import {
  ActionButton,
  BottomSheet,
  Button,
  Divider,
  Header,
  Icon,
  Input,
} from '../../components';
import { useNavigation } from '@react-navigation/native';
import { PromocodesProps } from './types';
import { PromocodeCard } from './components/SupplierCard';
import { useDispatch, useSelector } from 'react-redux';

import {
  createPromocode,
  deletePromocode,
  getPromocodesList,
```

```
  updatePromocode,
} from '../../store/slices/promocodes';
import { useTranslation } from 'react-i18next';
import { Toast } from 'react-native-toast-message/lib/src/Toast';
import { TOASTS } from '../../i18n/toasts';
import i18n from '../../i18n';

export const Promocodes: React.FC<PromocodesProps> = () => {
  const navigation = useNavigation<any>();
  const dispatch = useDispatch();
  const { t } = useTranslation();

  const [open, setOpen] = useState(false);
  const [edit, setEdit] = useState(false);
  const [item, setItem] = useState<any>(null);

  const [code, setCode] = useState('');
  const [percent, setPercent] = useState('');
  const [amount, setAmount] = useState('');

  const [isValidForm, setIsValidForm] = useState({
    code: true,
    percent: true,
    amount: true,
  });

  const validateForm = (onSuccess: any) => {
    let isValid = true;

    if (code.length > 2) {
      setIsValidForm(prev => ({ ...prev, code: true }));
    } else {
      isValid = false;
      setIsValidForm(prev => ({ ...prev, code: false }));
    }

    if (Number(percent) > 0) {
      setIsValidForm(prev => ({ ...prev, percent: true }));
    } else {
      isValid = false;
      setIsValidForm(prev => ({ ...prev, percent: false }));
    }
```

```
  if (Number(amount) > 0) {
    setIsValidForm(prev => ({ ...prev, amount: true }));
  } else {
    isValid = false;
    setIsValidForm(prev => ({ ...prev, amount: false }));
  }

  if (isValid) {
    onSuccess();
  }
};

const { promocodes } = useSelector((store: any) => store.promocode);
const { currentBusiness } = useSelector((store: any) => store.business);

const [promocodesList, setPromocodesList] = useState<any>(null);

useEffect(() => {
  setPromocodesList(promocodes);
}, [promocodes]);

return (
  <SafeAreaView style={styles.area}>
    <View style={styles.container}>
      <Header />
      <Divider height={20} />
      <View style={styles.headerWrapper}>
        <Text style={styles.titleText}>{t('promocodes')}</Text>
        <ActionButton
          iconName="plus"
          onPress={() => {
            setOpen(true);
            setEdit(false);
            setCode('');
            setPercent('');
            setAmount('');
          }}
          size="large"
        />
      </View>
      <Divider height={20} />
```

```
<FlatList
  showsVerticalScrollIndicator={false}
  data={promocodesList ?? []}
  renderItem={({ item }) => (
    <PromocodeCard
      name={item.promocode}
      phone={item.sale_percent + '%'}
      onDelete={() => {
        dispatch(
          deletePromocode(
            item?.id,
            () => {
              Toast.show({
                text1: TOASTS[i18n.language].SUCCESS_DELETE_PROMOCODE,
              });
            },
            (error: string) => {
              Toast.show({
                text1: TOASTS[i18n.language].ERROR,
                text2:
                  TOASTS[i18n.language][error] ?? 'Unexpected error',
                type: 'error',
              });
            },
          ) as any,
        );
      }}
      onEdit={() => {
        setItem(item);
        setEdit(true);
        setCode(item.promocode);
        setPercent(item.sale_percent + '');
        setAmount(item.use_amount + '');
        setOpen(true);
      }}
    />
  )}
  style={styles.list}
/>
```

```
</View>
<BottomSheet
  open={open}
  snapPoints={Platform.OS === 'ios' ? ['50%'] : ['60%']}
  onDismiss={() => {
    setOpen(false);
  }}>
  <Input
    placeholder={t('promocode')}
    value={code}
    onChange={setCode}
    inBottomSheet
    isValid={isValidForm.code}
  />
  <Divider height={20} />

  <Input
    placeholder={t('salePercent')}
    value={percent}
    onChange={setPercent}
    inBottomSheet
    isValid={isValidForm.percent}
  />
  <Divider height={20} />

  <Input
    placeholder={t('useAmount')}
    value={amount}
    onChange={setAmount}
    inBottomSheet
    isValid={isValidForm.amount}
  />
  <Divider height={30} />
  <Button
    text={edit ? t('update') : t('submit')}
    mode="large"
    onPress={() => {
      if (edit) {
        validateForm(() =>
          dispatch(
            updatePromocode(
              {
```

```
            promocode: code,
            useAmount: amount,
            salePercent: percent,
            businessId: currentBusiness?.id,
            promocodeId: item?.id,
          },
          () => {
            Toast.show({
              text1: TOASTS[i18n.language].SUCCESS_UPDATE_PROMOCODE,
            });
            setOpen(false);
          },
          (error: string) => {
            Toast.show({
              text1: TOASTS[i18n.language].ERROR,
              text2:
                TOASTS[i18n.language][error] ?? 'Unexpected error',
              type: 'error',
            });
          },
        ) as any,
      ),
    );
  } else {
    validateForm(() =>
      dispatch(
        createPromocode(
          {
            promocode: code,
            useAmount: amount,
            salePercent: percent,
            businessId: currentBusiness?.id,
          },
          () => {
            Toast.show({
              text1: TOASTS[i18n.language].SUCCESS_CREATE_PROMOCODE,
            });
            setOpen(false);
          },
          (error: string) => {
            Toast.show({
              text1: TOASTS[i18n.language].ERROR,
```

```
                        text2:
                          TOASTS[i18n.language][error] ?? 'Unexpected error',


                        type: 'error',
                      });
                    },
                  ) as any,
                ),
              );
            }
            setItem(null);
          }}
        />
      </BottomSheet>
    </SafeAreaView>
  );
};

const styles = StyleSheet.create({
  headerWrapper: {
    flexDirection: 'row',
    justifyContent: 'space-between',
    alignItems: 'center',
  },
  area: { flex: 1 },
  container: { paddingHorizontal: 15 },

  list: {
    marginTop: 20,
    height: '77%',
  },

  titleText: {
    fontFamily: Platform.OS === 'ios' ? 'Montserrat' : 'Montserrat-SemiBold',

    fontSize: 28,
    fontWeight: '600',
    color: '#000000',
  },
});
```

```
import React, { Fragment, useEffect, useState } from 'react';
import {
  FlatList,
  Keyboard,
  Platform,
  SafeAreaView,
  ScrollView,
  StyleSheet,
  Text,
  TouchableOpacity,
  TouchableWithoutFeedback,
  View,
} from 'react-native';
import {
  BottomSheet,
  Button,
  Divider,
  Header,
  Icon,
  Input,
} from '../../components';
import { useNavigation } from '@react-navigation/native';
import { NewOrderProps } from './types';
import { ProductCard } from './components/ProductCard';
import { useDispatch, useSelector } from 'react-redux';
import { getInventoryList } from '../../store/slices/inventory';
import {
  getCategoriesList,
  getProductsList,
} from '../../store/slices/products';
import { getSuppliersList } from '../../store/slices/suppliers';
import { addToBasket, getOrdersList } from '../../store/slices/orders';
import { getPromocodesList } from '../../store/slices/promocodes';
import { useTranslation } from 'react-i18next';

export const NewOrder: React.FC<NewOrderProps> = () => {
  const navigation = useNavigation<any>();

  const dispatch = useDispatch();
  const { t } = useTranslation();

  const { currentBusiness } = useSelector((store: any) => store.business);
```

```
const { currentBasket } = useSelector((store: any) => store.orders);
const { products } = useSelector((store: any) => store.products);
const { categories } = useSelector((store: any) => store.products);

const [productList, setProductList] = useState<any>(null);
const [category, setCategory] = useState<any>({ id: '' });

useEffect(() => {
  dispatch(getInventoryList(currentBusiness?.id) as any);
  dispatch(getProductsList(currentBusiness?.id) as any);
  dispatch(getSuppliersList(currentBusiness?.id) as any);
  dispatch(getPromocodesList(currentBusiness?.id) as any);
  dispatch(getOrdersList(currentBusiness?.id) as any);
  dispatch(getCategoriesList(currentBusiness?.id) as any);
}, []);

useEffect(() => {
  setProductList(products);
}, [products]);
// console.log(productList[1]);

useEffect(() => {
  if (category?.id) {
    setProductList(
      products.filter((item: any) => item?.category_id === category?.id),
    );
  } else {
    setProductList(products);
  }
}, [category]);

return (
  <SafeAreaView style={styles.area}>
    <View style={styles.container}>
      <Header />
      <Divider height={20} />



      <Text style={styles.titleText}>{t('newOrder')}</Text>
      <Divider height={20} />
      <View style={{ height: 35 }}>
```

```
  <ScrollView horizontal contentContainerStyle={styles.categoryScroll}>
    {!!categories?.length &&
      categories.map((el: any) => (
        <Fragment key={el.id}>
          <TouchableOpacity
            onPress={() => {
              el.id === category?.id
                ? setCategory({ id: '' })
                : setCategory(el);
            }}
            style={[
              styles.categoryItem,
              {
                backgroundColor:
                  el.id === category?.id ? '#384494' : '#D9F0FF',
              },
            ]}>
            <Text
              style={[
                styles.categoryText,
                {
                  color: el.id === category?.id ? 'white' : 'black',
                },
              ]}>
              {el?.name}
            </Text>
          </TouchableOpacity>
          <Divider width={5} />
        </Fragment>
      ))}
  </ScrollView>
</View>
<FlatList
  showsVerticalScrollIndicator={false}
  data={productList ?? []}
  numColumns={1}
  renderItem={({ item }) => (
    <ProductCard
      title={item?.name}
      image={item?.image ?? ''}
      price={item?.price}
      onAdd={() => {
```

```
                dispatch(addToBasket({ ...item, total: 1 }) as any);
              }}
            />
          )}
          style={styles.list}
          contentContainerStyle={styles.listContent}
        />
        <View style={styles.buttonWrapper}>
          <Button
            text={t('checkout') + ` (${currentBasket.length})`}
            onPress={() => {
              navigation.navigate('Basket');
            }}
          />
        </View>
      </View>
    </SafeAreaView>
  );
};

const styles = StyleSheet.create({
  area: { flex: 1 },
  container: { paddingHorizontal: 15 },
  buttonWrapper: {
    justifyContent: 'center',
    alignItems: 'center',
    width: '100%',
    marginTop: 10,
  },

  list: {
    marginTop: 20,
    height: Platform.OS === 'ios' ? '70%' : '60%',
  },
  listContent: {
    justifyContent: 'center',
  },

  categoryText: {
    fontFamily: Platform.OS === 'ios' ? 'Montserrat' : 'Montserrat-Regular',
  },
  categoryItem: {
```

```
    paddingHorizontal: 15,
    paddingVertical: 5,
    borderRadius: 50,

    justifyContent: 'center',
    alignItems: 'center',
  },
  categoryScroll: {
    height: 35,
    width: '100%',
  },
  titleText: {
    fontFamily: Platform.OS === 'ios' ? 'Montserrat' : 'Montserrat-SemiBold',
    fontSize: 28,
    fontWeight: '600',
    color: '#000000',
  },
});

import React, { useEffect, useState } from 'react';
import {
  FlatList,
  Keyboard,
  Platform,
  SafeAreaView,
  StyleSheet,
  Text,
  TouchableOpacity,
  TouchableWithoutFeedback,
  View,
} from 'react-native';
import {
  BottomSheet,
  Button,
  Divider,
  Header,
  Icon,
  Input,
} from '../../components';
import { useNavigation } from '@react-navigation/native';
import { BasketProps } from './types';
import { ProductCard } from './components/ProductCard';
```

```
import { useDispatch, useSelector } from 'react-redux';
import {
  addToBasket,
  createOrder,
  getOrdersList,
  removeFromBasket,
} from '../../store/slices/orders';
import { useTranslation } from 'react-i18next';
import { checkValidPromocode } from '../../store/slices/business';
import { Toast } from 'react-native-toast-message/lib/src/Toast';
import { TOASTS } from '../../i18n/toasts';
import i18n from '../../i18n';

export const Basket: React.FC<BasketProps> = () => {
  const navigation = useNavigation<any>();
  const dispatch = useDispatch();

  const { t } = useTranslation();

  const { currentBasket, orders } = useSelector((store: any) => store.orders);
  const { currentBusiness } = useSelector((store: any) => store.business);

  const [basket, setBasket] = useState<any>([]);
  const [payBy, setPayBy] = useState<any>('cash');

  const [open, setOpen] = useState<any>(false);
  const [promo, setPromo] = useState<any>('');
  const [promoResponse, setPromoResponse] = useState<any>(null);

  useEffect(() => {
    setBasket(currentBasket);
  }, [currentBasket]);

  const countTotalPrice = (total: Array<any>) => {
    let totalPrice = 0;
    total.forEach((el: any) => (totalPrice += el.total * el.price));
    return totalPrice;
  };

  function calculateDiscountedPrice(
    originalPrice: number,
    discountPercentage: number,
```

```
) {
  const discountAmount = (originalPrice * discountPercentage) / 100;

  const finalPrice = originalPrice - discountAmount;

  return finalPrice;
}
return (
  <SafeAreaView style={styles.area}>
    <View style={styles.container}>
      <Header withGoBack />
      <Divider height={20} />
      <Text style={styles.titleText}>{t('yourItems')}</Text>
      <FlatList
        showsVerticalScrollIndicator={false}
        data={basket}
        renderItem={({ item }) => (
          <ProductCard
            title={item?.name}
            total={item?.total}
            price={item?.price}
            image={item?.image ?? ''}
            onAdd={() => {
              dispatch(addToBasket(item) as any);
            }}
            onRemove={() => {
              dispatch(removeFromBasket(item) as any);
            }}
          />
        )}
        style={styles.list}
      />
      <View style={styles.buttonWrapper}>
        <View style={styles.textWrapper}>
          <Text style={styles.text}>{t('totalItems')}</Text>
          <Text style={styles.text}>{basket?.length ?? 0}</Text>
        </View>
        <View style={styles.textWrapper}>
          <Text style={styles.text}>{t('price')}</Text>
          <Text style={styles.text}>
            {`${
              promoResponse?.salePercent
```

```
          ? calculateDiscountedPrice(
              countTotalPrice(basket),
              promoResponse?.salePercent,
            )
          : countTotalPrice(basket)
      }${currentBusiness?.currency ?? ''}`}
    </Text>
  </View>
  <View style={styles.textWrapper}>
    <Text style={styles.text}>{t('payWith') + ':'}</Text>
    <View style={styles.paymentWrapper}>
      <TouchableOpacity
        onPress={() => {
          setPayBy('cash');
        }}>
        <Text
          style={[
            styles.text,
            {
              opacity: payBy === 'cash' ? 1 : 0.4,
              fontWeight: payBy === 'cash' ? '600' : '400',
            },
          ]}>
          {t('cash')}


        </Text>
      </TouchableOpacity>
      <Divider width={20} />
      <TouchableOpacity
        onPress={() => {
          setPayBy('card');
        }}>
        <Text
          style={[
            styles.text,
            {
              opacity: payBy === 'card' ? 1 : 0.4,
              fontWeight: payBy === 'card' ? '600' : '400',
            },
          ]}>
          {t('card')}
```

```
      </Text>
    </TouchableOpacity>
  </View>
</View>
<Divider height={20} />
<Button
  text={`${t('checkPromo')}${
    promoResponse?.salePercent
      ? ' (' + promoResponse?.salePercent + '%)'
      : ''
  }
    `}
  onPress={() => {
    setOpen(true);
  }}
  mode="lite"
/>
<Divider height={10} />

<Button
  text={t('confirm')}
  onPress={() => {
    dispatch(
      createOrder(
        {
          businessId: currentBusiness?.id,
          payType: payBy,
          products: basket.flatMap(({ id, total }: any) =>
            Array.from({ length: total }, () => String(id)),
          ),
          promocodeId: promoResponse?.id ?? '',
        },
        () => {
          Toast.show({
            text1: TOASTS[i18n.language].SUCCESS_CREATE_ORDER,
          });

          dispatch(getOrdersList(currentBusiness?.id) as any);
          navigation.goBack();
        },
        (error: string) => {
          Toast.show({
```

```
                            text1: TOASTS[i18n.language].ERROR,
                            text2:   TOASTS[i18n.language][error]   ??   'Unexpected
error',
                            type: 'error',
                          });
                        },
                      ) as any,
                    );
                  }}
                />
              </View>
            </View>
            <BottomSheet
              open={open}
              snapPoints={['40%']}
              onDismiss={() => {
                setOpen(false);
              }}>
              <Input  placeholder={t('promocode')}  onChange={setPromo}  inBottomSheet
/>

              <Divider height={40} />
              <Button
                text={t('submit')}
                mode="large"
                onPress={() => {
                  dispatch(
                    checkValidPromocode(
                      {
                        businessId: currentBusiness.id,
                        promocodeName: promo,
                      },
                      (res: any) => {
                        Toast.show({
                          text1: TOASTS[i18n.language].VALID_PROMOCODE,
                        });
                        setOpen(false);
                        setPromoResponse(res);
                      },
                      (error: string) => {
                        Toast.show({
                          text1: TOASTS[i18n.language].ERROR,
```

```
                    text2: TOASTS[i18n.language][error] ?? 'Unexpected error',
                    type: 'error',
                  });
                },
              ) as any,
            );
          }}
        />
      </BottomSheet>
    </SafeAreaView>
  );
};

const styles = StyleSheet.create({
  area: { flex: 1 },
  container: { paddingHorizontal: 15 },
  paymentWrapper: {
    flexDirection: 'row',
    width: '50%',
    justifyContent: 'flex-end',
  },
  textWrapper: {
    flexDirection: 'row',
    width: '100%',
    justifyContent: 'space-between',
    marginBottom: 10,
  },
  text: {
    fontFamily: Platform.OS === 'ios' ? 'Montserrat' : 'Montserrat-SemiBold',

    fontSize: 16,
    fontWeight: '500',
    color: '#000000',
    opacity: 0.4,
  },

  buttonWrapper: {
    justifyContent: 'center',
    alignItems: 'center',
    width: '100%',
    marginTop: 10,
  },
```

```
  list: {
    marginTop: 20,
    height: Platform.OS === 'ios' ? '58%' : '45%',
  },



  titleText: {
    fontFamily: Platform.OS === 'ios' ? 'Montserrat' : 'Montserrat-SemiBold',

    fontSize: 28,
    fontWeight: '600',
    color: '#000000',
  },
});
```