Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: "Розробка крос-платформного мобільного застосунку для пасіки "Божий Дар" на основі фреймворка Flutter"

Виконав: студент IV курсу, групи СНс-42
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)

_____ Михальський Є.В.
(підпис) (прізвище та ініціали)

Керівник _____ Дмитроца Л.П.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Марценко С.В.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ Оробчук О.Р.
(підпис) (прізвище та ініціали)

Тернопіль
2024

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«__» _червня_ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп’ютерні науки
(шифр і назва спеціальності)

Студенту Михальському Євгенію Васильовичу
(прізвище, ім’я, по батькові)

1. Тема роботи "Розробка крос-платформного мобільного застосунку для пасіки "Божий Дар" на основі фреймворка Flutter"

Керівник роботи Дмитроца Леся Павлівна, к.т.н., доцент кафедри КН
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*29*» *квітня* 2024 року № *4/7-472*

2. Термін подання студентом завершеної роботи 24 червня 2024р.

3. Вихідні дані до роботи Технічне завдання, вимоги замовника, документація, результати попередніх досліджень та аналізів.

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1 Аналіз предметної області та постановка завдання. 1.1 Аналітичний огляд існуючих рішень. 1.2 Технічне завдання. 1.2.1 Найменування та область застосування. 1.2.2 Призначення розробки. 1.2.3 Вимоги до функціоналу застосунку. 1.2.4 Стадії та етапи розробки. 1.2.5 Порядок тестування та прийому. 1.3 Висновок до першого розділу. 2 Проєктна частина. Проєктування мобільного застосунку. 2.1 Архітектура застосунку. 2.2 Архітектура представлення. 2.3 Дизайн застосунку. 2.4 Розробка бази даних та моделей. 2.5 Розробка інтерфейсу користувача. 2.6 Висновок до другого розділу. 3 Практична частина. Реалізація мобільного застосунку. 3.1 Реалізація аутентифікації. 3.2 Програмування основного функціоналу. 3.2.1 Функціонал адміністратора. 3.2.2 Функціонал клієнта. 3.3 Тестування застосунку 3.4 Висновок до третього розділу. 4 Безпека життєдіяльності, основи охорони праці. 4.1 Безпека життєдіяльності: Комплексний аналіз життєдіяльності людини 4.2 Основи охорони праці: Психологічні чинники небезпеки. 4.3 Висновок до четвертого розділу. Висновки. Перелік джерел. Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень, слайдів)
1 Титульний слайд. 2 Технічне завдання. 3 Обґрунтування вибору технології Flutter. 4 Можливості Flutter. 5 Архітектура застосунку. 6 Дизайн застосунку. 7 Розробка моделей та бази даних. 8 Firebase та аутентифікація. 9 Функціонал застосунку. 10 Висновки та Подальший розвиток. 11 Заключний слайд.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Сенчишин В. С., доцент кафедри МТ		

7. Дата видачі завдання 29 січня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	30.01.2024	
2.	Підбір джерел про крос-платформну розробку мобільних застосунків	31.01.2024-03.02.2024	
3.	Опрацювання джерел щодо розробки крос-платформного мобільного застосунку для пасіки «Божий Дар»	04.02.2024-06.02.2024	
4.	Виконання дослідження щодо розробки крос-платформних мобільних застосунків. Розроблення застосунку для пасіки «Божий Дар»	07.02.2024-11.02.2024	
5.	Оформлення розділу «Аналіз предметної області та постановка завдання»	03.06.2024-05.06.2024	
6.	Оформлення розділу «Проєктна частина. Проєктування мобільного застосунку»	06.06.2024-08.06.2024	
6.	Оформлення розділу «Практична частина. Реалізація мобільного застосунку»	09.06.2024-11.06.2024	
7.	Виконання завдання до підрозділу «Безпека життєдіяльності»		
8.	Виконання завдання до підрозділу «Основи охорони праці»		
9.	Оформлення кваліфікаційної роботи	16.06.2024-17.06.2024	
10.	Нормоконтроль	18.06.2024-19.06.2024	
11.	Перевірка на плагіат	20.06.2024	
12.	Попередній захист кваліфікаційної роботи	21.06.2024	
13.	Захист кваліфікаційної роботи	29.06.2024	

Студент _____ Михальський Є.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Дмитроца Л.П.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

"Розробка крос-платформного мобільного застосунку для пасіки Дар" на основі фреймворка Flutter" // Кваліфікаційна робота освітнього рівня «Бакалавр» // Михальський Євгеній Васильович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СНс-42 // Тернопіль, 2024 // С. , рис. – 15, додат. – 5, бібліогр. – 35.

Ключові слова: крос-платформний застосунок, пасіка, Flutter, мобільна розробка, Dart, UI/UX дизайн, база даних.

Кваліфікаційна робота присвячена розробці крос-платформного мобільного застосунку для пасіки "Божий Дар" на основі фреймворка Flutter.

В першому розділі описано аналітичний огляд існуючих рішень, технічне завдання, вимоги до функціоналу, стадії розробки та порядок прийому.

У другому розділі досліджено загальну архітектуру, архітектуру представлення, дизайн застосунку, розробку бази даних та користувацький інтерфейс.

У третьому розділі описано реалізацію основного функціоналу, аутентифікацію, функціонал адміністратора та клієнта,

Об'єкт дослідження: мобільний застосунок для управління пасікою. Предмет дослідження: процеси розробки, проектування та впровадження мобільного застосунку для адміністратора і клієнта.

ANNOTATION

"Development of a Cross-Platform Mobile Application of the Apiary "God's Gift" Based on the Flutter Framework" // Qualification work of the educational level "Bachelor" // Mykhalskyi Yevhenii// Ternopil Ivan Pulyu National Technical University, Computer and Information Systems and Software Engineering Faculty, Computer Sciences Department, group SNs-42 // Ternopil, 2024 // P. fig. – 15, annexes. – 5, references – 35.

Keywords: cross-platform application, apiary, Flutter, mobile development, Dart, UI/UX design, database.

The qualification work is dedicated to the development of a cross-platform mobile application for the " God's Gift " apiary based on the Flutter framework.

The first section describes the analytical review of existing solutions, the technical task, functional requirements, development stages, and acceptance procedure.

The second section investigates the overall architecture, presentation architecture, application design, database development, and user interface.

The third section describes the implementation of the main functionality, authentication, administrator and client functionality.

Object of research: mobile application for apiary management. Subject of research: development, design, and implementation processes of the mobile application for the administrator and client.

ПЕРЕЛІК СКОРОЧЕНЬ

AOT-компіляція – Ahead-Of-Time-компіляція

APK – Android Package Kit

ERD – Entity-Relationship Diagram

JIT-компіляція – Just-In-Time-компіляція

UI – User Interface

OTP – One-Time Password

БД – База даних

ООП – Об'є́ктно-орієнто́ване програмува́ння

ЗМІСТ

ВСТУП

Внаслідок глобалізації та зростаючого попиту на натуральні продукти, зокрема бджолині продукти, ефективне управління пасіками та оптимізація процесів продажу меду набувають особливого значення. Актуальність теми визначається не лише зростаючим попитом на натуральні продукти, але й потребою у зручних та ефективних інструментах для управління та взаємодії з клієнтами. Компанії повинні включати нові цифрові рішення на постійній основі з метою максимізації зростання доходів [1]. В умовах сучасного ринку, де користувачі використовують різноманітні пристрої та платформи, крос-платформні мобільні застосунки стають надзвичайно важливими.

Мобільні застосунки дозволяють адміністраторам пасік ефективно керувати своєю діяльністю та спілкуватися з клієнтами на будь-яких пристроях, незалежно від їх операційної системи. Приблизно третина цих компаній використовують технології, що підтримують цифровізацію операцій, такі як управління взаємовідносинами з клієнтами або системи планування ресурсів підприємства [2].

Розробка крос-платформних мобільних застосунків на основі фреймворка Flutter забезпечує одночасну підтримку різних мобільних операційних систем. Такий підхід дозволяє максимально зростити аудиторію та ефективність взаємодії з нею, що робить його найбільш оптимальним рішенням для розвитку сучасного бізнесу у сфері бджільництва. Першим кроком для підтримки бізнесу є перевірка їх цифрової готовності, можливостей і розробка чітких планів покращення [3].

Метою даної кваліфікаційної роботи є підвищення якості послуг у сфері бджільництва шляхом розробки крос-платформного мобільного застосунку для пасіки «Божий Дар» на основі фреймворка Flutter. Для досягнення поставленої мети необхідно вирішити низку завдань, зокрема:

– проаналізувати стан досліджень у галузі крос-платформних мобільних застосунків;

– вивчити вимоги замовника та технічне завдання;

– розробити архітектуру програми;

– створити дизайн інтерфейсу користувача (UI/UX);

– реалізувати основні функціональні можливості застосунків, включаючи контроль кількості літрів меду, керування замовленнями та оптимізацію продажів для адміністратора, а також зручний інтернет-магазин для клієнтів;

– протестувати та налагодити застосунок.

Розроблений мобільний застосунок дозволить пасіці «Божий Дар» значно підвищити ефективність управління замовленнями та продажами меду, забезпечуючи зручність і прозорість цих процесів. Адміністратор зможе контролювати кількість літрів меду, керувати замовленнями та оптимізувати продажі, а клієнти отримають можливість зручно замовляти мед через інтернет-магазин. Це сприятиме підвищенню продуктивності, якості обслуговування та задоволеності клієнтів.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Аналітичний огляд існуючих рішень

Мобільний застосунок – це програмне забезпечення, призначене для роботи на мобільних пристроях, таких як смартфони та планшети [4]. Він дає користувачам можливість виконувати певні функції, взаємодіяти з сервісами, робити онлайн замовлення та отримувати інформацію прямо на свої пристрої незалежно від часу та місця. Мобільні застосунки стають все більш популярними завдяки своїй зручності, доступності та функціональності, що дозволяє користувачам отримувати доступ до потрібних сервісів у будь-який момент.

Крос-платформна мобільна розробка – це процес створення мобільних застосунків, які можуть працювати на різних платформах, таких як Android та iOS [5]. Це означає, що одна кодова база може бути використана для створення застосунків для різних операційних систем.

Переваги крос-платформної мобільної розробки:

– економія часу та ресурсів: за допомогою крос-платформної розробки можна зменшити час розробки та витрати, оскільки один код може бути використаний для декількох платформ;

– більша аудиторія: крос-платформні застосунки можуть бути запущені на різних пристроях, що розширює аудиторію користувачів;

– спільний код: зменшення кількості коду допомагає знизити ризик виникнення помилок та спрощує процес обслуговування.

У сучасному світі швидко розвивається технологічна сфера, що ставить перед розробниками мобільних застосунків великі виклики [6]. Одним із найважливіших питань є вибір платформи для розробки, яка забезпечить якість, швидкість та ефективність програмного продукту на різних пристроях. У цьому контексті виникає необхідність у порівнянні різних крос-платформних рішень.

Найпопулярніші фреймворки крос-платформної мобільної розробки:

1. React Native: бібліотека, яка дозволяє розробляти мобільні застосунки за допомогою JavaScript та React.

2. Kotlin Multiplatform: дозволяє розробляти крос-платформні застосунки, використовуючи мову програмування Kotlin та спільний код для різних платформ.

3. Flutter: SDK, розроблений Google, який дозволяє створювати красиві та швидкі крос-платформні застосунки за допомогою мови програмування Dart [7].

Для створення застосунку було обрано фреймворк Flutter через його численні переваги. Flutter забезпечує швидку розробку завдяки великій кількості вбудованих компонентів та інструментів, що дозволяє зосередитися на функціоналі застосунку. Крім того, він має велику швидкодію та продуктивність завдяки використанню власного механізму візуалізації Impeller. Також Flutter забезпечує якісний та адаптивний вигляд та поведінку застосунків на різних платформах, що робить його вигідним вибором для крос-платформної розробки. До всіх вищезазначених переваг слід додати, що Flutter користується значною та активною спільнотою розробників, що забезпечує підтримку, допомогу та подальший розвиток фреймворку.

Flutter використовує мову програмування Dart як свій основний інструмент для розробки мобільних застосунків. Dart – це мова програмування, розроблена компанією Google, яка використовується для створення різних типів застосунків, включаючи веб-додатки, серверні додатки та мобільні застосунки [8]. Вона є сучасною, об'єктно-орієнтованою мовою програмування зі строгою типізацією, але яка підтримує і динамічну типізацію.

Dart має простий та зрозумілий синтаксис, що дозволяє швидко навчатися та ефективно писати код. Вона підтримує багато сучасних концепцій програмування, таких як асинхронність, потоки, функціональне програмування та ООП, надає підтримку інтерфейсів та міксин, що сприяють кращому

розподіленні архітектурних рівнів застосунку, полегшують роботу з об'єктами, підвищують читабельність та зручність кодової бази.

Однією з важливих особливостей Dart є його велика екосистема бібліотек та пакетів, які допомагають розробникам розвивати різноманітні застосунки [9]. Це включає в себе бібліотеки для роботи з HTTP-запитами, роботи з базами даних, реалізації UI та багато іншого. Dart також підтримує крос-платформну розробку, що дозволяє розробникам писати код один раз та компілювати його для різних платформ, таких як веб, мобільні платформи та сервери.

Dart має два типи компіляції: JIT та AOT.

JIT-компіляція використовується під час розробки для швидкого перекладу джерелних кодів Dart у виконуваний код на льоту. Це дозволяє розробникам бачити зміни в реальному часі під час розробки та швидко вносити зміни в код. Використання JIT-компіляції забезпечує швидку ітерацію у розробці та полегшує відлагодження коду. AOT-компіляція використовується для компіляції коду Dart в машинний код перед виконанням програми [10]. Це дозволяє збільшити швидкодію та продуктивність програми, оскільки вона виконується безпосередньо на пристрої користувача. AOT-компіляція особливо корисна для мобільних застосунків, де швидкість та ефективність мають велике значення. Обидва типи компіляції мають свої переваги у розробці. JIT-компіляція зручна для швидкої ітерації та відлагодження коду, тоді як AOT-компіляція забезпечує високу швидкодію та ефективність виконання застосунків. Ці два підходи доповнюють один одного і дозволяють створювати високоякісні та ефективні програми на Dart.

Загалом, Dart – це потужна та різнобічна мова програмування, яка дозволяє розробникам створювати різноманітні та ефективні застосунки для різних платформ [11].

Враховуючи всі переваги Flutter як фреймворку та мови програмування Dart, найкращим вибором для розробки мобільного застосунку для пасіки "Божий Дар" виявляється використання цих інструментів. Цей фреймворк

дозволяє створити високоякісний, продуктивний та естетично привабливий застосунок для користувачів на платформах Android та iOS.

1.2 Технічне завдання

1.2.1 Найменування та область застосування

Найменування – застосунок для пасіки "Божий Дар". Це програмне забезпечення призначене для оптимізації управління пасікою та продажів продуктів бджільництва. Застосунок буде доступний на двох основних мобільних платформах: Android та iOS.

Область застосування застосунку для адміністратора:

1. Контроль кількості меду: адміністратор може відслідковувати запаси меду в реальному часі. Це дозволяє вчасно планувати виробництво та запобігати нестачі товару.

2. Керування замовленнями: застосунок надає можливість обробляти замовлення клієнтів, відслідковувати їх статус. Це допомагає зменшити час обробки замовлень та покращити обслуговування клієнтів.

3. Оптимізація продажів: застосунок надає інструменти для аналізу продажів, що дозволяє виявляти найпопулярніші продукти, планувати маркетингові кампанії та розробляти стратегії для збільшення обсягів продажів.

4. Статистика та аналітика: застосунок включає функції для збору та аналізу даних щодо продажів, замовлень та запасів. Це допомагає адміністратору приймати обґрунтовані рішення на основі реальних даних.

5. Налаштування знижок та акцій: адміністратор може встановлювати знижки на різні види продукції, включаючи мед, а також контролювати систему кешбеку для постійних клієнтів. Це сприяє залученню нових клієнтів та стимулює повторні покупки.

6. Розширення переліку товарів: застосунок дозволяє адміністратору додавати нові товари до каталогу, не обмежуючись лише медом. Це можуть

бути інші продукти бджільництва, такі як прополіс, маточне молочко, віск тощо. Це розширює асортимент і збільшує потенційний обсяг продажів.

Область застосування застосунку для клієнта:

1. Зручний інтернет-магазин: клієнти можуть переглядати каталог продукції, включаючи мед та інші продукти бджільництва, робити замовлення та відслідковувати їх статус. Це забезпечує комфортний та швидкий процес покупки.

2. Особистий кабінет: клієнти мають доступ до свого особистого кабінету, де можуть переглядати історію замовлень, змінювати особисті дані та переключати колірну тему. Це підвищує зручність використання застосунку.

3. Підтримка та зворотний зв’язок: через застосунок клієнти можуть звертатися до служби підтримки, залишати відгуки та пропозиції. Це сприяє покращенню якості обслуговування та підвищенню задоволеності клієнтів.

4. Клієнти можуть з зручністю відслідковувати статус свого замовлення.

5. Клієнти можуть частково або повністю оплачувати покупки бонусами, отриманими за попередні успішні покупки в застосунку.

Застосунок покликаний значно підвищити ефективність управління пасікою та процесами продажу меду, забезпечуючи зручність та прозорість як для адміністратора, так і для клієнтів. Завдяки широким можливостям застосунку, пасіка "Божий Дар" зможе досягти нових висот у своїй діяльності, забезпечуючи високу якість продукції та сервісу.

1.2.2 Призначення розробки

Метою розробки мобільного застосунку для пасіки "Божий Дар" є забезпечення комплексного рішення для оптимізації управління та підвищення ефективності процесів продажу продуктів бджільництва. Застосунок призначений для виконання кількох основних функцій, які сприяють автоматизації та полегшенню роботи адміністратора, а також забезпечують зручний доступ до продуктів для клієнтів.

Для адміністратора застосунок служить інструментом для відстеження запасів меду, обробки замовлень, аналізу продажів і проведення маркетингових кампаній. Він також дозволяє налаштовувати знижки та систему кешбеку, розширювати асортимент продукції.

Для клієнтів застосунок пропонує зручний інтерфейс для замовлення меду та інших продуктів бджільництва, доступ до спеціальних пропозицій, особистий кабінет для керування замовленнями та можливість зв'язатися зі службою підтримки.

Таким чином, розробка мобільного застосунку спрямована на створення високоякісного програмного забезпечення, яке забезпечить ефективне управління пасікою та покращить взаємодію з клієнтами, що сприятиме зростанню бізнесу та підвищенню задоволеності клієнтів.

1.2.3 Вимоги до функціоналу застосунку

Функціонал мобільного застосунку повинен відповідати наступним вимогам:

1. Адміністраторський інтерфейс:

– управління запасами: застосунок повинен забезпечувати можливість редагування та контролю кількості меду та інших продуктів бджільництва пасіки "Божий Дар";

– обробка замовлень: адміністратор повинен мати інструменти для прийому та обробки та замовлень;

– аналітика продажів: застосунок має слугувати аналітичним інструментом для порівняльної характеристики продуктів та відстеження лідерів продажу;

– маркетинг і знижки: інтегровані функції для створення знижок та керування системою кешбеку;

– розширення асортименту: можливість додавання нових товарів, оновлення інформації про наявні продукти та управління каталогом продукції.

2. Клієнтський інтерфейс:

– зручний каталог товарів: інтуїтивно зрозумілий інтерфейс для перегляду та вибору продуктів, зокрема меду та інших бджолиних продуктів;

– особистий кабінет: користувачі повинні мати можливість створювати та управляти своїми обліковими записами, переглядати історію замовлень та нараховані бонуси за попередні замовлення;

– процес замовлення: спрощена та швидка процедура оформлення замовлень з можливістю вибору способу доставки;

– зворотній зв'язок: інструменти для зв'язку зі службою підтримки та отримання допомоги у разі виникнення питань або проблем.

3. Технічні вимоги:

– крос-платформність: застосунок повинен однаково добре функціонувати на операційних системах Android та iOS;

– безпека: забезпечення конфіденційності даних користувачів та захист від несанкціонованого доступу [12];

– швидкодія та стабільність: висока продуктивність та стабільність роботи застосунку, швидке завантаження та обробка даних [13];

– масштабованість: можливість розширення функціоналу та інтеграції нових модулів у майбутньому.

Дотримання цих вимог гарантуватиме створення ефективного, надійного та зручного для використання мобільного застосунку, який задовольнить потреби як адміністратора, так і клієнтів пасіки "Божий Дар". Це дозволить оптимізувати внутрішні процеси управління запасами та замовленнями, підвищити ефективність продажів, а також забезпечити клієнтам зручний доступ до продуктів пасіки через інтуїтивно зрозумілий інтерфейс. Крім того, інтеграція функцій аналітики та маркетингу сприятиме стратегічному розвитку бізнесу, допомагаючи вчасно реагувати на зміни попиту та впроваджувати успішні рекламні кампанії. Завдяки високій продуктивності, стабільності та безпеці, застосунок забезпечить безперебійну роботу та надійний захист даних

користувачів, що є ключовими аспектами для досягнення довгострокового успіху та задоволення клієнтів.

1.2.4 Стадії та етапи розробки

Для розробки мобільного застосунку пасіки "Божий Дар" використовується систематичний підхід, що включає кілька ключових етапів. Починаючи з аналізу вимог, відбувається збір та детальне вивчення потреби адміністратора та клієнтів, визначаючи необхідний функціонал. Далі відбувається проєктування інтерфейсу, створюється візуальна концепція та розміщення елементів для забезпечення зручної навігації.

Розробка застосунку включає дві основні частини: бекенд та фронтенд. На етапі розробки бекенду створюються серверні компоненти для обробки бізнес-логіки, роботи з базою даних та інших адміністративних функцій . Паралельно з цим розробляється фронтенд для мобільних пристроїв, забезпечуючи його адаптивність та естетичний вигляд.

Останній етап включає налагодження, формування apk та bundle перед фінальним продакшином.

1.2.5 Порядок тестування та прийому

Для успішного прийому розробленого мобільного застосунку для пасіки "Божий Дар" необхідно дотримуватися встановлених термінів, що зазначені в індивідуальному завданні. Виконавець зобов'язаний забезпечити повну відповідність технічному завданню, яке передбачає створення мобільного застосунку для розширення функціональності та зручності використання продукції. Прийом повинен відбуватися перед комісією у такій послідовності:

– підготовка доповіді, в якій будуть висвітлені ключові моменти розробки, особливості архітектури та технічні вирішення, що використовувалися під час створення застосунку;

– демонстрація роботи мобільного застосунку, що включатиме показ інтерфейсу, основних функцій, а також можливостей інтеракції з користувачем;

– проведення контрольних тестів для впевненості в правильному функціонуванні застосунку;

– відповіді на запитання та зауваження комісії Замовника щодо розробленого мобільного застосунку.

Крім того, для належного представлення розробленого програмного забезпечення необхідно підготувати презентацію, яка ілюструватиме ключові особливості, переваги та потенціал застосунку для користувачів застосунку.

1.3 Висновок до першого розділу

У першому розділі здійснено ґрунтовний аналіз предметної області та постановка завдання для розробки мобільного застосунку для пасіки "Божий Дар". Проведено аналітичний огляд існуючих рішень, що дозволило оцінити конкурентне середовище та визначити найкращі практики, які можна застосувати у розробці власного продукту. Вивчено область застосування, що надало можливість чітко окреслити призначення розробки, що у свою чергу визначило основні напрями функціоналу.

На основі проведеного аналізу були сформульовані вимоги до функціоналу застосунку, що включають публікацію нових товарів, надання інформації щодо замовлень і знижок, ознайомлення клієнтів з продукцією, управління замовленнями та надання контактних даних. Це забезпечить зручність у використанні як для адміністратора, так і для клієнтів, та сприятиме ефективному виконанню бізнес-процесів.

Було також розглянуто стадії та етапи розробки, що включають проєктування, дизайн та реалізацію. Це дозволить структуровано підходити до процесу створення застосунку, забезпечуючи високу якість кінцевого продукту. Окремо було визначено порядок тестування та прийому розробки, що гарантує

відповідність готового застосунку встановленим вимогам та його стабільну роботу.

Отже, у першому розділі були визначені всі необхідні передумови для успішного започаткування розробки мобільного застосунку для пасіки "Божий Дар", що включають чітке розуміння цілей, вимог та етапів реалізації проєкту.

РОЗДІЛ 2. ПРОЄКТНА ЧАСТИНА. ПРОЄКТУВАННЯ МОБІЛЬНОГО ЗАСТОСУНКУ

2.1 Архітектура застосунку

Архітектура мобільного застосунку – це структура, що визначає логічне розташування його функціональних модулів, компонентів та інтерфейсів [15]. Вона являє собою план, у якому простежується логічний зв'язок між різними частинами застосунку. Створення архітектури застосунку є важливою частиною розробки проєкту, без неї неможливо почати саму розробку.

Для мобільного застосунку пасіки "Божий Дар" було обрано архітектуру MVVM (Model-View-ViewModel). Ця архітектура підтримує двонаправлене зв'язування даних, що полегшує розробку.

Для розробки структурних схем застосунку можна використовувати багато онлайн-сервісів, таких як Draw.io, Creately, SmartDraw, Microsoft Visio, Google Docs, Wireflow та Canva. Ознайомившись з цими прикладами, було прийнято рішення використовувати сервіс Draw.io за його зручність та ефективність. Інтерфейс даного сервісу зображено на рисунку 2.1.

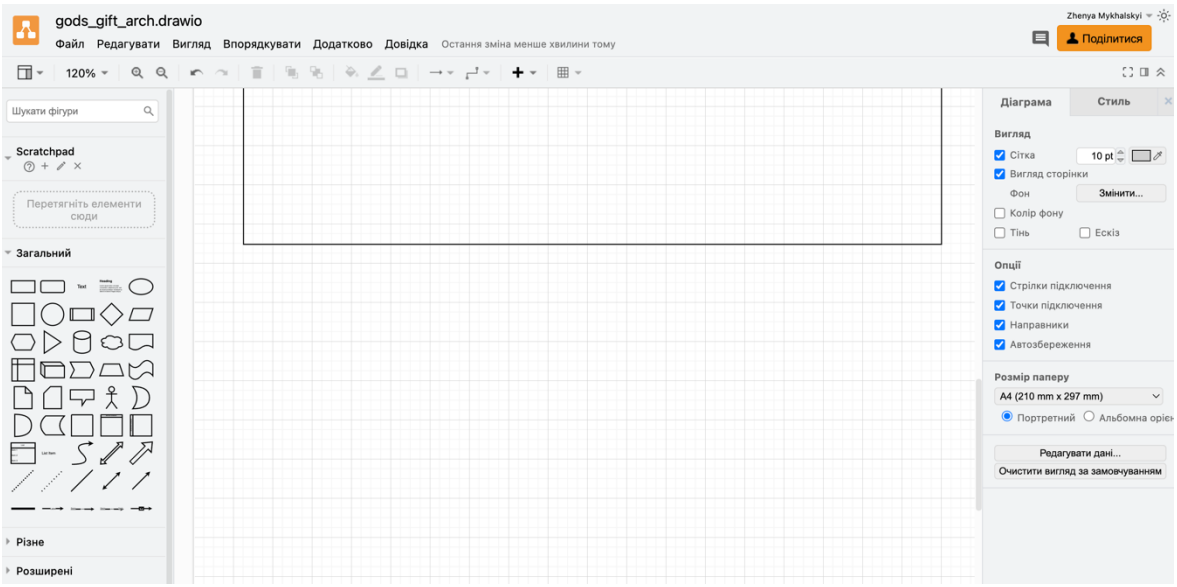

Рисунок 2.1 – Інтерфейс сервісу Draw.io

Компоненти структури застосунку включають два різних види інтерфейсів – для адміністратора (див. рисунок 2.2) та для клієнта (див. рисунок 2.3), залежно від того, хто входить у застосунок. При авторизації програма перевіряє номер телефону користувача: якщо це номер адміністратора, то завантажується адмін-версія застосунку, інакше – клієнтська версія.

Адмін-версія має три основні екрани. Головний екран, де адміністратор може переглядати товари. Екран редагування, де адміністратор може редагувати інформацію про товари та систему кешбеку, робити знижку на конкретний товар. Екран замовлень, де показуються всі замовлення, які роблять юзери.

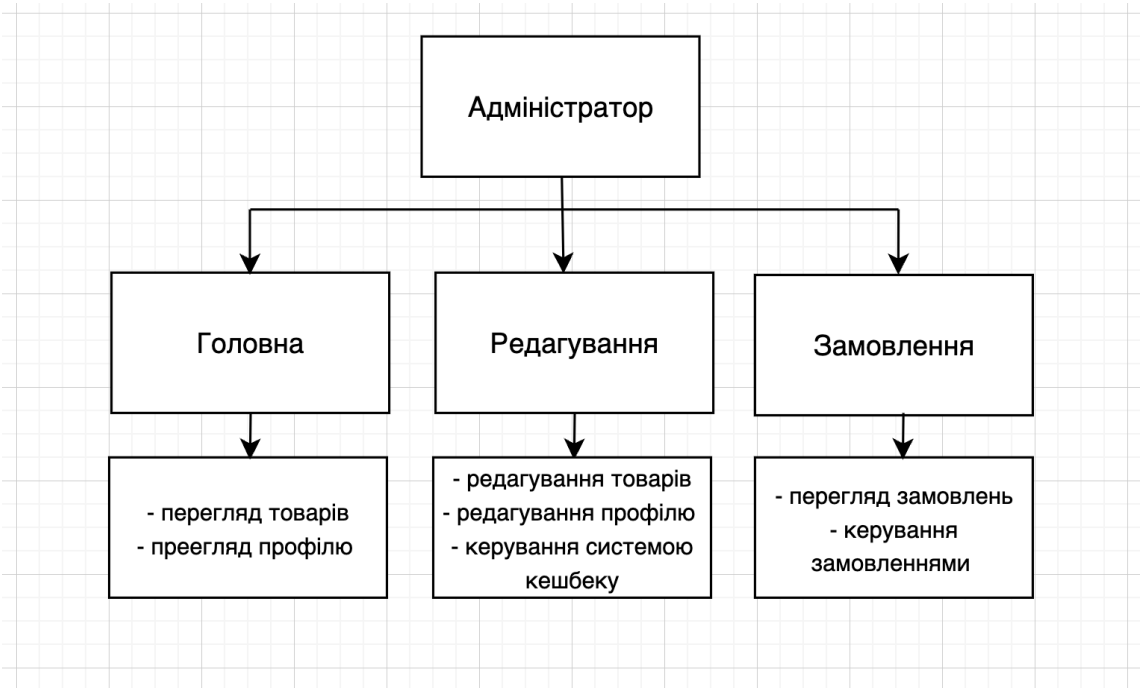

Рисунок 2.2 – Схема застосунку адміністратора

Клієнтська версія також має три основні екрани. Головний екран, де користувач може переглядати товари, додавати їх до корзини і переходити до оформлення замовлення. Екран профілю користувача, де він може переглядати свої бонуси, редагувати інформацію для доставки та переглядати свої замовлення. Екран контактної інформації, де міститься інформація про магазин та адміністратора.

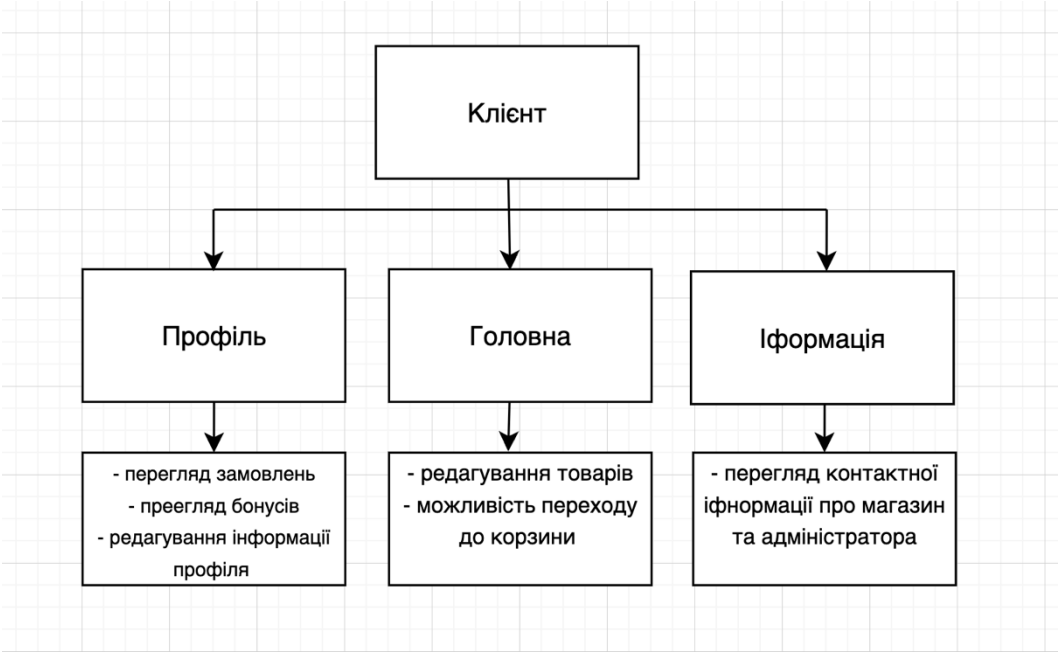

Рисунок 2.3 – Схема застосунку клієнта

Розробка архітектурної схеми почалась з ознайомлення вимог до функціоналу застосунку. Для візуалізації функціоналу було розроблено діаграму прецедентів, що зображена на рисунку 2.4.

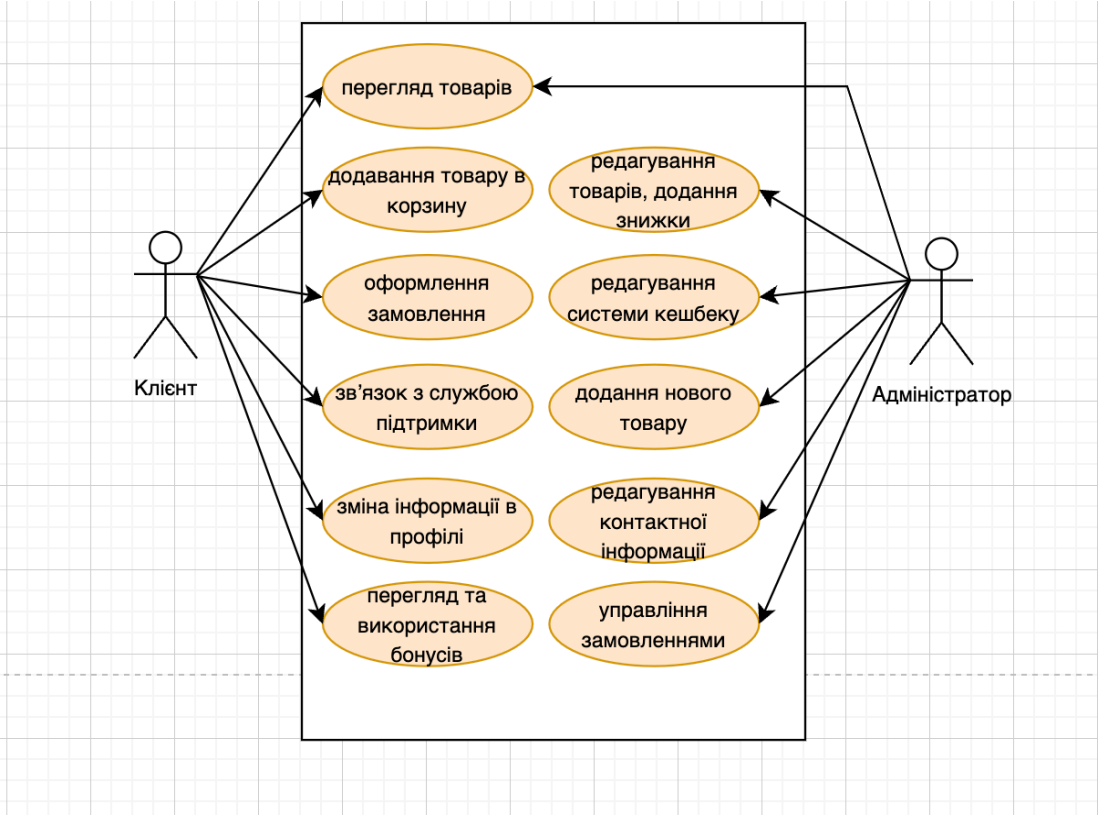

Рисунок 2.4 – Діаграма прецедентів застосунку

Функціонал застосунку включає публікацію нових товарів в каталог, надання інформації щодо замовлень і знижок, ознайомлення клієнтів з продукцією, доступною до замовлення, управління замовленнями, включаючи статуси та історію замовлень, надання контактних даних для зв’язку з адміністрацією пасіки.

Окрім цього, архітектура забезпечує гнучкість та масштабованість, що дозволяє легко додавати нові функції в майбутньому. Усі ці елементи разом утворюють цілісну систему, спрямовану на забезпечення ефективної роботи як для користувачів, так і для адміністрації.

Неможливо недооцінити важливість головного екрану. Саме ознайомлення з ним формує перше враження про застосунок, а отже і в цілому про пасіку "Божий Дар". При створенні головного екрану потрібно бути найбільш ретельним, він повинен одразу зацікавити користувача своїм наповненням. Оскільки основною задачею застосунку є забезпечення зручності та прозорості процесів замовлення та продажу меду, то і наповнення головного екрану буде виходити з даної цілі.

2.2 Архітектура представлення

Архітектура представлення мобільного застосунку визначає, як дані відображаються на екрані користувача і як користувачі взаємодіють із застосунком. Це критично важливий аспект, оскільки від нього залежить зручність та інтуїтивність інтерфейсу користувача.

Основні концепції архітектури представлення включають компонентний підхід, двонаправлене зв'язування даних та менеджмент стану [16]. Компонентний підхід означає, що всі елементи інтерфейсу представлені як незалежні компоненти або віджети. Це дозволяє легко керувати ними, повторно використовувати та тестувати. Двонаправлене зв'язування даних забезпечує автоматичне оновлення інтерфейсу при зміні стану даних і навпаки. Менеджмент стану стосується процесу контролю і керування станом

застосунку, що гарантує збереження консистентності даних та ефективності взаємодії з інтерфейсом [17].

Для керування станом у Flutter існує кілька популярних стейт-менеджерів:

1. Provider: Популярний менеджер стану, що забезпечує простий та зручний спосіб керування станом за допомогою контексту.

2. Riverpod: Покращена версія Provider, що надає більш гнучкий і потужний спосіб управління станом.

3. Bloc/Cubit: Архітектурний патерн для керування станом, що базується на потоках та подіях. Він добре підходить для великих та складних проєктів.

4. GetX: Простий у використанні менеджер стану, який пропонує високий рівень продуктивності та зручність у реалізації.

5. MobX: Реактивний менеджер стану, що дозволяє створювати ефективні та гнучкі структури стану застосунку [18].

Для розробки мобільного застосунку пасіки "Божий Дар" було обрано Provider як основний стейт-менеджер. Це рішення випливало з таких міркувань: Provider є відносно простим у налаштуванні та використанні, що знижує поріг входження для розробників. Він дозволяє легко інтегруватися з іншими бібліотеками та інструментами, забезпечуючи гнучкість у реалізації різних функціональних вимог. Крім того, Provider забезпечує високу продуктивність завдяки ефективному управлінню станом та мінімальному використанню ресурсів [19]. Його популярність також пояснюється великою та активною спільнотою розробників, що забезпечує швидке вирішення проблем та підтримку.

У архітектурі представлення застосунку використовуються три провайдера для ефективного управління даними та функціоналом. Cart Provider відповідає за зберігання і маніпулювання даними корзини покупок, забезпечуючи користувачам зручний інтерфейс для додавання, видалення та зміни товарів у корзині. Products Provider керує інформацією про товари, дозволяючи адміністраторам додавати нові товари, оновлювати наявні,

встановлювати ціни та знижки. Також, цей провайдер забезпечує швидкий доступ до даних для відображення продуктів на головному екрані клієнтського застосунку. Theme Provider є не менш важливим елементом, забезпечуючи можливість зміни теми застосунку. Цей провайдер дозволяє користувачам вибирати між темною та світлою кольоровими темами та оформленням інтерфейсу, що підвищує користувацький комфорт та персоналізацію застосунку.

Кожен з цих провайдерів використовується для зберігання стану застосунку і керування його поведінкою відповідно до функціональних вимог. Інкапсуляція логіки в провайдерах дозволяє зменшити зв'язність між компонентами застосунку, що покращує його масштабованість та зручність підтримки. Такий підхід до організації архітектури дозволяє ефективно працювати над розширенням функціональності застосунку, забезпечуючи високу якість інтерфейсу та надійність його роботи.

Загалом, використання Provider як стейт-менеджера забезпечує стабільність та ефективність управління станом, сприяючи створенню інтуїтивного та привабливого інтерфейсу для користувачів і адміністратора.

2.3 Дизайн застосунку

Дизайн мобільного застосунку є одним із ключових елементів, який визначає зручність користування та загальне враження від продукту [20]. Гарний дизайн не лише робить застосунок привабливим для користувачів, але й забезпечує інтуїтивно зрозумілий інтерфейс, який полегшує взаємодію з функціоналом застосунку. При розробці дизайну важливо враховувати цільову аудиторію, потреби користувачів, а також сучасні тенденції у сфері UI/UX.

Колірна палітра є важливою частиною дизайну мобільного застосунку. Вибір кольорів може впливати на настрій користувача, його сприйняття інформації та загальну естетику застосунку. Використання відповідних кольорів допомагає створити гармонійний та привабливий інтерфейс. Зазвичай,

дизайнери використовують обмежену кількість основних кольорів, доповнюючи їх відтінками та акцентними кольорами для виділення важливих елементів [21]. Палітри можуть включати основні кольори, що визначають загальний вигляд і відчуття від застосунку, акцентні кольори для виділення важливих елементів, таких як кнопки чи посилання, та нейтральні кольори, що забезпечують баланс та допомагають основним і акцентним кольорам виділятися. Є безліч генераторів готових колірних палітр, які використовують алгоритми для формування збалансованої колірної палітри, що буд візуально приємна користувачу. Приклад готових колірних палітр зображено на рисунку 2.5.

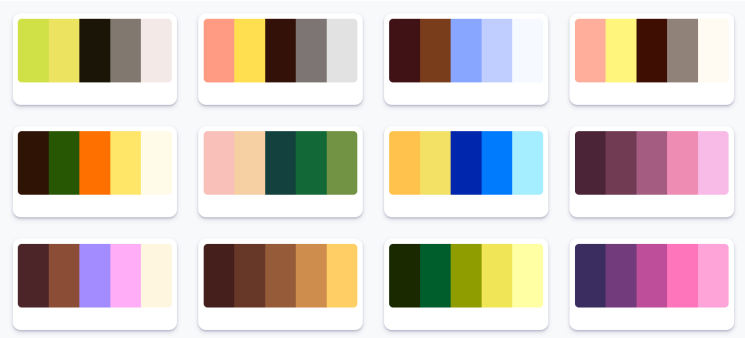

Рисунок 2.5 – Приклади готових колірних палітр

Для створення дизайну мобільних застосунків використовуються різноманітні інструменти, які забезпечують зручність роботи та багатий функціонал для дизайнерів. Серед найпопулярніших інструментів:

– Figma;
– Adobe XD;
– Sketch;
– InVision;
– Marvel;
– Zeplin.

В якості інструменту для створення дизайну мобільного застосунку пасіки "Божий Дар" було обрано Figma. Figma забезпечує ефективну співпрацю

між дизайнерами та розробниками завдяки можливості роботи в реальному часі [22], наявності потужних інструментів для прототипування та підтримки версійності. Вона також дозволяє легко експортувати елементи дизайну та інтегрувати їх у код застосунку. Інтерфейс даного сервісу зображено на рисунку 2.6.

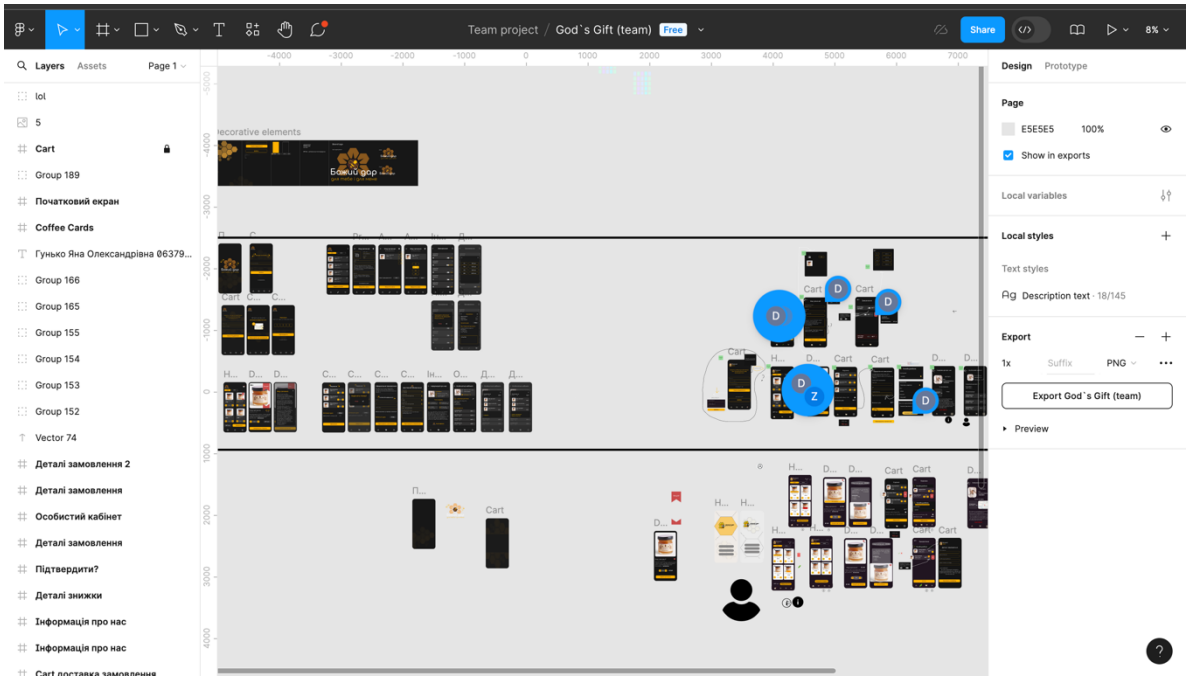

Рисунок 2.6 – Інтерфейс сервісу Figma

Шрифти також є дуже важливим елементом дизайну, який впливає на читабельність та естетику інтерфейсу. Вибір відповідного шрифту допомагає створити професійний та привабливий вигляд застосунку [23]. Популярними шрифтами для мобільних застосунків є Roboto, Open Sans, Lato та Montserrat. Провівши аналіз макетів різних застосунків для розробки дизайну було вирішено використовувати шрифт Montserrat Alternates (див. рисунок 2.7). Цей шрифт чудово доповнює загальний стиль застосунку та його графічні елементи, забезпечуючи легкість читання та ненав'язливість. Важливим критерієм при виборі цього шрифту стала також підтримка кирилиці.

При створенні дизайну мобільного застосунку було враховано всі вищезгадані аспекти. Використання Figma дозволило ефективно прототипувати

інтерфейси та створювати інтерактивні макети, які відображають функціонал застосунку. Колірна палітра була підібрана з урахуванням естетики та зручності для користувачів, забезпечуючи приємний та гармонійний вигляд інтерфейсу. Шрифти були обрані з метою забезпечити високу читабельність та елегантність текстових елементів.

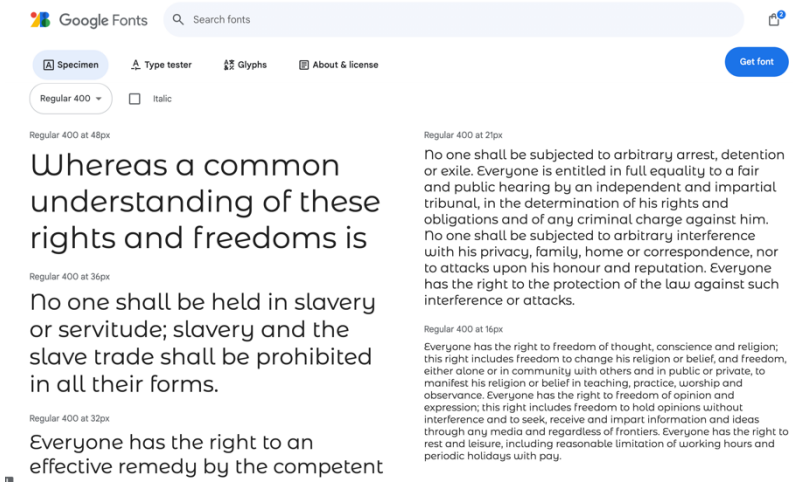

Рисунок 2.7 – Шрифт Montserrat Alternates

Дизайн мобільного застосунку є невід'ємною частиною його успіху, тому важливо підходити до цього процесу з увагою до деталей та зосереджуючись на потребах користувачів. Це дозволяє створити продукт, який не лише виглядає привабливо, але й забезпечує зручність та інтуїтивність у використанні.

Важливим аспектом дизайну також є розробка логотипу [24], який буде асоціюватись з брендом пасіки "Божий Дар". Логотип має бути простим, легко запам'ятовуваним та відображати основну ідею бренду. Логотип для застосунку зображено на рисунку 2.8.

Рисунок 2.8 – Логотип застосунку

Для застосунку було обрано логотип, що включає зображення бджоли та стилізоване зображення медових сот, що символізує продукцію пасіки. Логотип неодноразово використовується в застосунку для створення впізнаваності та єдності візуального стилю. Його присутність на різних екранах та елементах інтерфейсу допомагає користувачам легко асоціювати застосунок із пасікою, забезпечуючи єдиний бренд. Зображення бджоли було обрано не випадково – воно символізує працьовитість, природність та якість продукції, що важливо для нашого бренду. Стилізовані медові соти відображають основний продукт пасіки – мед. Логотип використовується на головному екрані, в шапці застосунку, що сприяє підвищенню впізнаваності бренду та забезпечує єдність візуального стилю.

2.4 Розробка бази даних та моделей

Для розробки моделей та бази даних в мобільному застосунку пасіки "Божий Дар" було обрано Firestore Database, що є частиною платформи Firebase. Firestore Database надає можливість зберігати дані у вигляді колекцій та документів, що дозволяє організовувати інформацію у структурований та зручний для доступу формат [25]. Крім того, для зберігання зображень товарів та аватарів користувачів було використано Firebase Storage. Це рішення дозволяє зберігати великі об'єми файлів безпосередньо у хмарі, забезпечуючи їхню доступність та безпеку. Інтерфейс сервісу Firebase зображено на рисунку 2.9.

Колекції на Firestore були структуровані таким чином, щоб відповідати функціональним вимогам застосунку. Зокрема, було створено наступні колекції: admin, users, products та orders.

Колекція "admin" містить два важливі документи. Перший документ, "cashback", зберігає сітку кешбек нарахувань. Ця сітка включає інформацію про відсотки нарахувань, що залежать від загальної суми, витраченої клієнтом на замовлення. Така структура дозволяє адміністратору гнучко керувати

політикою кешбеків, надаючи різні відсотки нарахувань залежно від витрат клієнтів. Другий документ у колекції "admin" містить дані про адміністратора та магазин, забезпечуючи централізоване зберігання критичної інформації про керівництво та контактні дані.

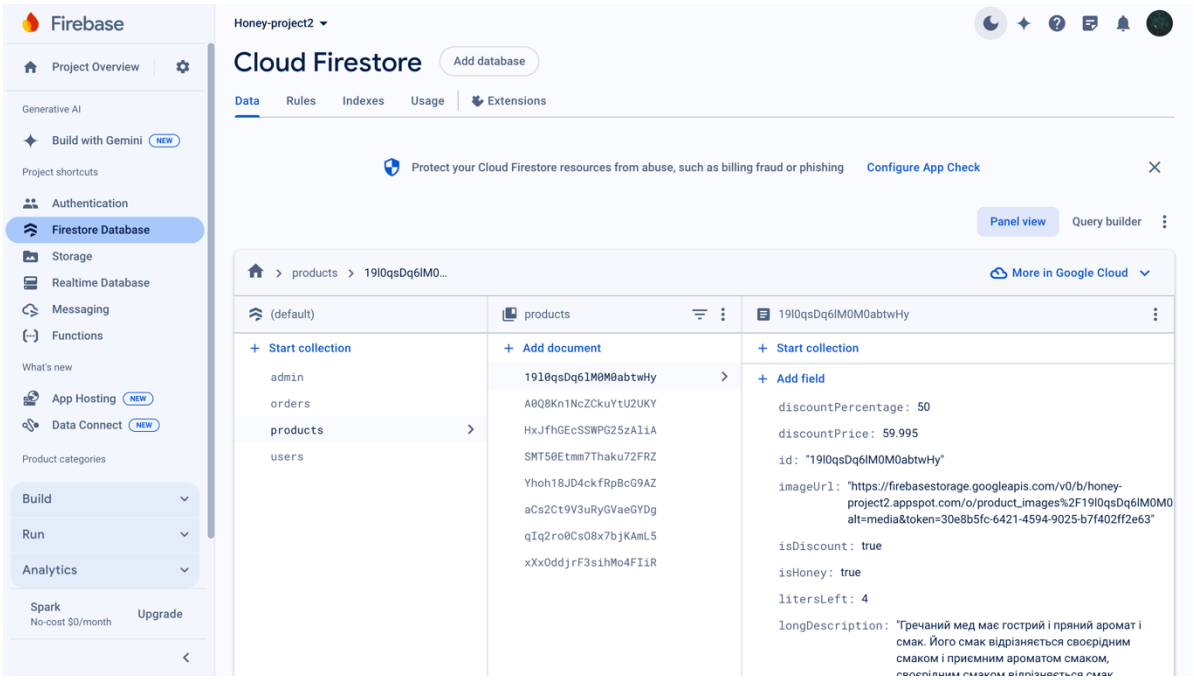

Рисунок 2.9 – Інтерфейс сервісу Firebase

Колекція "users" включає документи, кожен з яких відповідає окремому користувачеві застосунку. Ці документи містять інформацію про користувача, включаючи його особисті дані, історію замовлень, нараховані бонуси та налаштування профілю. Завдяки такій структурі адміністратор може легко відслідковувати активність користувачів, а також надавати їм персоналізовані пропозиції та підтримку.

Колекція "products" містить документи, що описують кожен продукт, доступний у каталозі магазину. Кожен документ включає такі атрибути, як назва продукту, опис, ціна, наявність, а також посилання на зображення, збережені у Firebase Storage. Це дозволяє адміністратору додавати нові товари, оновлювати інформацію про існуючі продукти та видаляти товари з каталогу у разі необхідності. Така організація даних забезпечує зручний доступ до

інформації про продукцію для користувачів застосунку, дозволяючи їм легко переглядати та замовляти товари.

Колекція "orders" включає документи, що відповідають кожному замовленню, зробленому через застосунок. Кожен документ містить інформацію про замовлення, включаючи деталі продуктів, що були замовлені, статус замовлення, а також контактні дані клієнта. Така структура дозволяє адміністраторам легко відслідковувати всі замовлення, керувати їхніми статусами та забезпечувати ефективну обробку кожного замовлення.

Розробка моделей та бази даних почалася з визначення вимог до функціоналу застосунку. Було важливо забезпечити ефективне зберігання та доступ до даних, враховуючи різні сценарії використання застосунку. Графічно діаграма моделей (ERD) зображена на рисунку 2.10. Лістинг коду моделей наведено у додатках А, Б, В та Д.

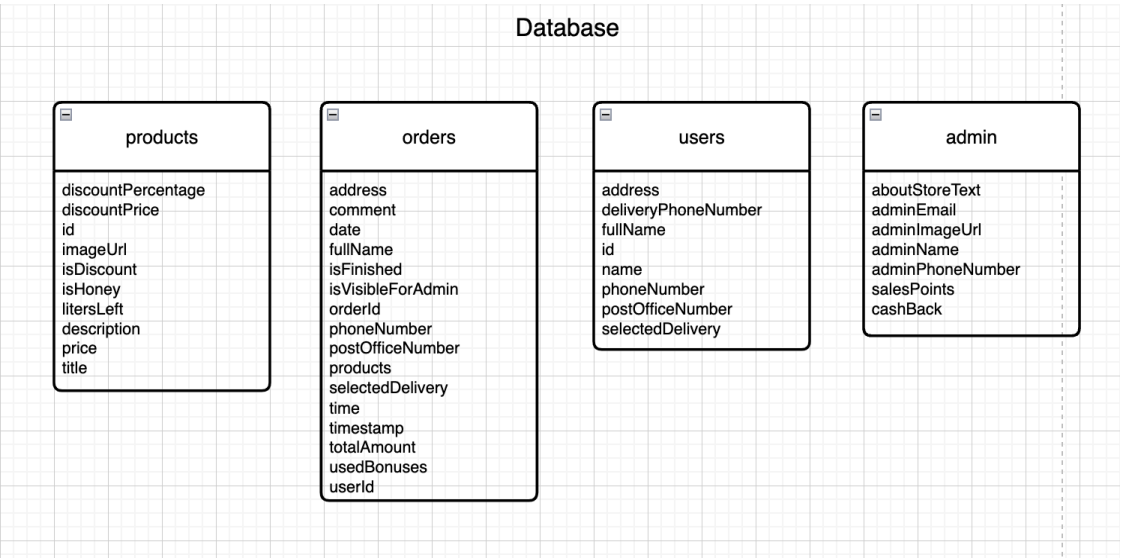

Рисунок 2.10 – Діаграма моделей бази даних

Вибір Firestore Database обумовлювався його гнучкістю та можливістю масштабування, що є критичним для застосунків з динамічно змінюваними даними. Firestore Database надає ряд переваг для розробників, включаючи автоматичну синхронізацію даних між клієнтами у реальному часі, масштабованість та високий рівень безпеки. Використання Firebase Storage для

зберігання зображень дозволяє зменшити навантаження на базу даних та забезпечити швидкий доступ до файлів незалежно від їхнього розміру [26].

Розробка моделей даних також включала визначення взаємозв'язків між різними типами даних. Наприклад, кожне замовлення у колекції "orders" містить посилання на користувача з колекції "users" та продукти з колекції "products". Це дозволяє забезпечити цілісність даних та спростити їхню обробку у застосунку. Завдяки використанню моделей даних, можна легко маніпулювати даними у коді, забезпечуючи високу продуктивність та надійність застосунку.

Загалом, розробка моделей та бази даних для мобільного застосунку пасіки "Божий Дар" є критично важливим етапом, що забезпечує ефективне зберігання та обробку даних. Вибір Firestore Database та Firebase Storage дозволив створити гнучку, масштабовану та надійну структуру, що відповідає всім вимогам застосунку.

2.5 Розробка інтерфейсу користувача

Розробка інтерфейсу користувача (UI) є однією з найважливіших частин створення мобільного застосунку. Вона охоплює всі елементи, з якими взаємодіє користувач, і має на меті забезпечити зручність, естетичну привабливість та інтуїтивну зрозумілість. При розробці застосунку особлива увага приділялась забезпеченню простоти використання, логічності розташування елементів та відповідності сучасним тенденціям дизайну мобільних інтерфейсів [27].

Основний інтерфейс мобільного застосунку було розроблено з урахуванням потреб як адміністратора, так і звичайних користувачів. Після авторизації за номером телефону програма визначає, чи є користувач адміністратором, і в залежності від цього показує відповідний інтерфейс. Якщо вхід здійснює адміністратор, він потрапляє на головний екран адміністраторського інтерфейсу, де може переглядати пости, редагувати товари

та керувати замовленнями. Інтерфейс застосунку адміністратора зображено на рисунку 2.11. Для звичайних користувачів передбачено клієнтський інтерфейс з можливістю перегляду та замовлення товарів, управління профілем і доступу до контактної інформації.

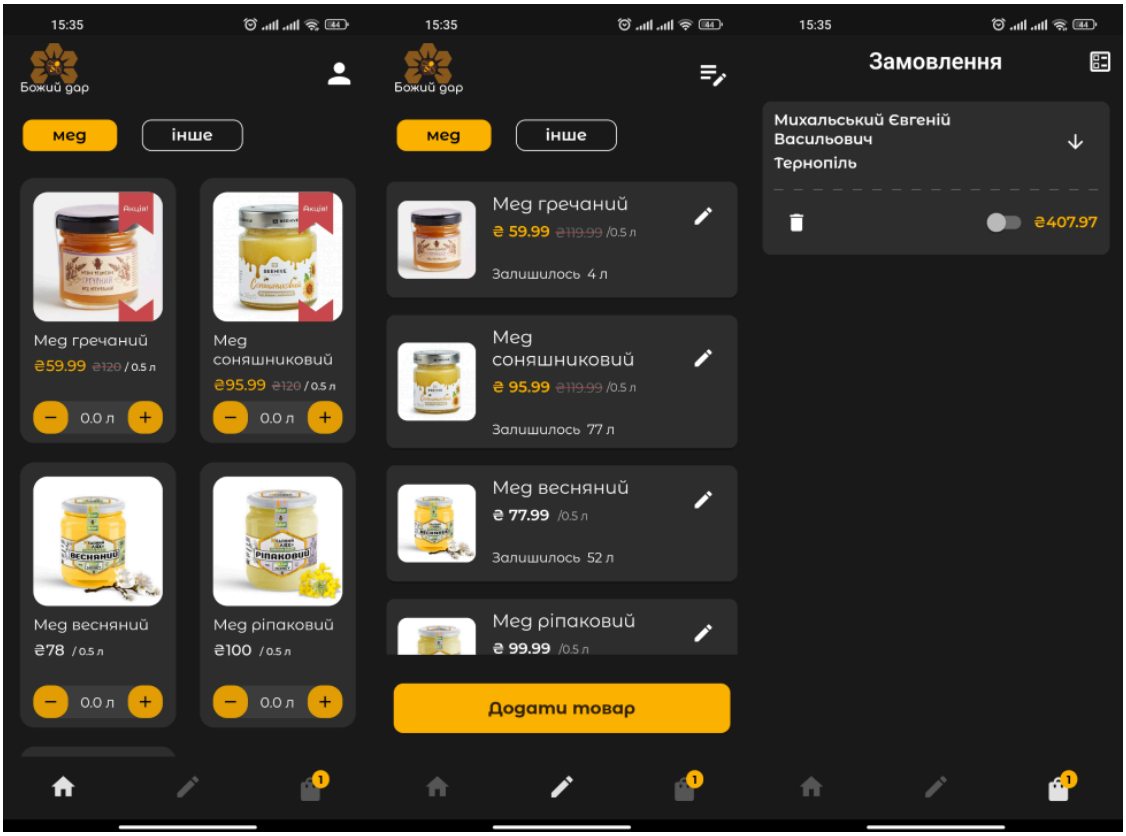

Рисунок 2.11 – Інтерфейс застосунку адміністратора

Головний екран адміністратора включає три основні секції: перегляд товарів та інформації про магазин, редагування товарів та управління системою кешбеку, управління замовленнями. У секції перегляду товарів адміністратор може бачити всі добавлені товари, що дозволяє швидко орієнтуватись та бачити як виглядає сітка товарів для покупців. Секція редагування дозволяє змінювати інформацію про товари, додавати нові товари, налаштовувати знижки та кешбек. Важливо, щоб ці функції були легко доступні та зрозумілі для адміністратора, тому акцент був зроблений на простоті та логічності розташування елементів. Секція управління замовленнями надає

адміністраторам можливість переглядати всі замовлення та маніпулювати ними.

Для звичайних користувачів інтерфейс також має три основні екрани: головний екран, екран профілю та екран контактної інформації. Інтерфейс застосунку клієнта зображено на рисунку 2.12.

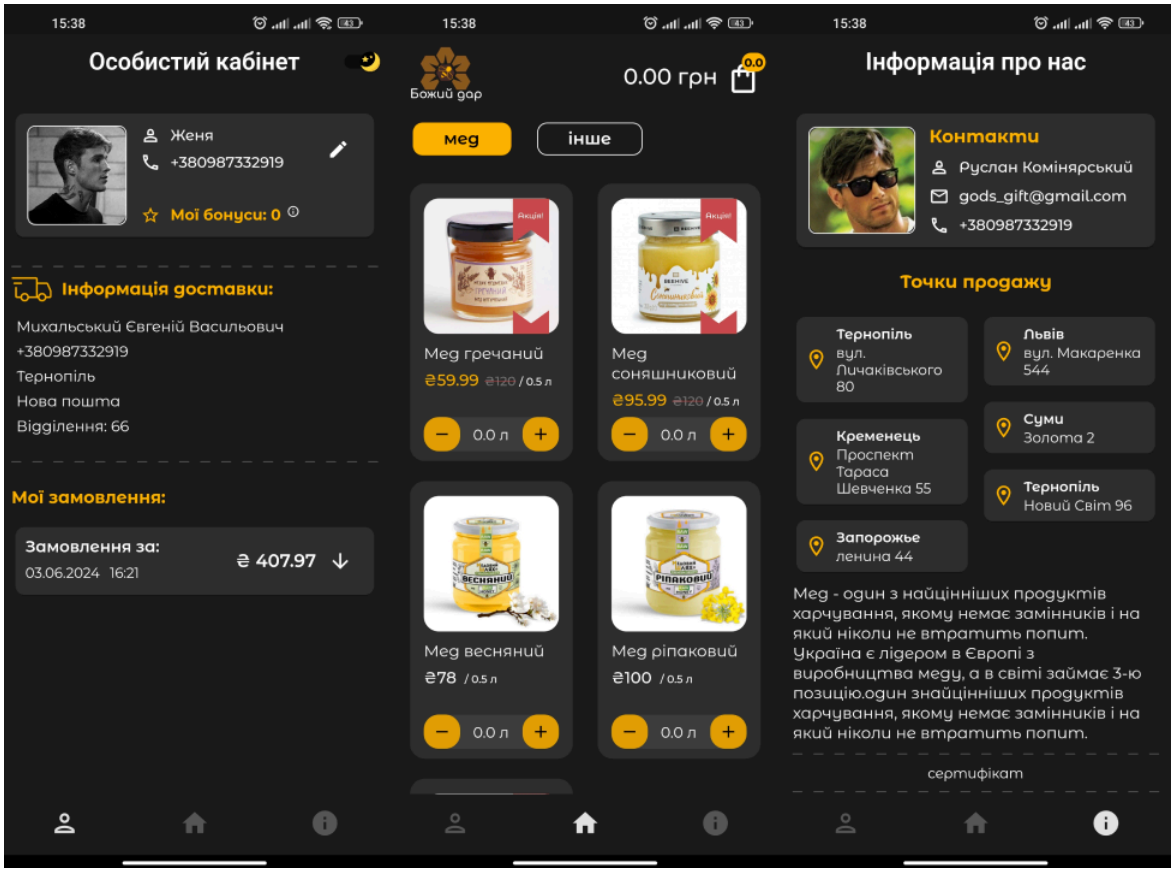

Рисунок 2.12 – Інтерфейс застосунку клієнта

Головний екран призначений для перегляду асортименту товарів, додавання їх до корзини та переходу до оформлення замовлення. Інтерфейс головного екрану розроблено з урахуванням максимального спрощення навігації та забезпечення зручного доступу до всіх функцій. Користувач може легко знайти потрібний товар завдяки чіткій організації каталогу, добавити потрібний об’єм у корзину та одразу перейти до оформлення замовлення. Важливим аспектом є також відображення акційних пропозицій, що допомагає залучати увагу користувачів до актуальних товарів.

Екран профілю користувача включає можливість перегляду бонусів, редагування інформації для доставки та перегляду історії замовлень. Цей екран має бути інтуїтивно зрозумілим та легко доступним, щоб користувач міг швидко знайти необхідну інформацію та внести зміни.

Екран контактної інформації надає користувачам можливість швидко знайти контактні дані магазину та адміністрації, а також переглянути місце розташування точок продажу. Цей екран має забезпечити легкий доступ до всієї необхідної інформації для зв'язку з магазином, тому особлива увага приділялася зручності його використання та чіткому розташуванню елементів.

Для розробки інтерфейсу користувача було використано інструмент Figma, який дозволяє створювати високоякісні дизайни та забезпечує зручність командної роботи. Завдяки цьому інструменту вдалося забезпечити високу узгодженість між різними екранами та елементами інтерфейсу, що сприяє покращенню користувацького досвіду.

Для кожного екрану інтерфейсу було розроблено окремі макети, що включають всі необхідні елементи та функції. Це дозволяє забезпечити високу якість дизайну та зручність його реалізації в коді.

Загалом, розробка інтерфейсу користувача є складним та багатоетапним процесом, що вимагає уваги до деталей та злагодженої роботи команди дизайнерів та розробників. Використання сучасних інструментів та технологій дозволяє створювати високоякісні та зручні інтерфейси, що забезпечують позитивний користувацький досвід та сприяють успіху мобільного застосунку [28].

2.6 Висновок до другого розділу

У другому розділі було детально розглянуто архітектурні та дизайнерські аспекти розробки мобільного застосунку пасіки "Божий Дар". Архітектура застосунку базується на MVVM (Model-View-ViewModel), що забезпечує ефективне двонаправлене зв'язування даних із розділенням на інтерфейси для

адміністратора і клієнта. Використання сервісу Draw.io дозволило створити структурні схеми зручно та ефективно.

Крім того, архітектура представлення обрана з урахуванням компонентного підходу та управління станом за допомогою Provider для забезпечення ефективності та масштабованості застосунку. Вибір цього стейт-менеджера обгрунтовується його простотою та гнучкістю у реалізації різноманітних функціональних вимог.

Нарешті, дизайн застосунку був створений з використанням Figma для забезпечення співпраці між дизайнерами та розробниками, а також для створення гармонійного інтерфейсу з використанням відповідних колірних палітр та шрифтів. Цей підхід дозволяє підвищити зручність взаємодії з користувачем та створити інтуїтивно зрозумілий і естетично приємний дизайн.

Таким чином, розглянуті аспекти архітектури та дизайну дозволяють забезпечити якісний та зручний мобільний застосунок для пасіки "Божий Дар", відповідно до вимог користувачів та сучасних стандартів розробки програмного забезпечення.

РОЗДІЛ 3. ПРАКТИЧНА ЧАСТИНА. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ МОБІЛЬНОГО ЗАСТОСУНКУ

3.1 Реалізація аутентифікації

Для реалізації системи аутентифікації в застосунку пасіки "Божий Дар" було обрано Firebase Authentication, що є однією з найпопулярніших та надійних систем авторизації для мобільних застосунків, надаючи простий і ефективний спосіб управління автентифікацією користувачів. Цей сервіс розроблений компанією Google і інтегрований з іншими сервісами Firebase, що робить його чудовим вибором для сучасних мобільних застосунків. Інтерфейс Firebase Authentication зображено на рисунку 3.1.

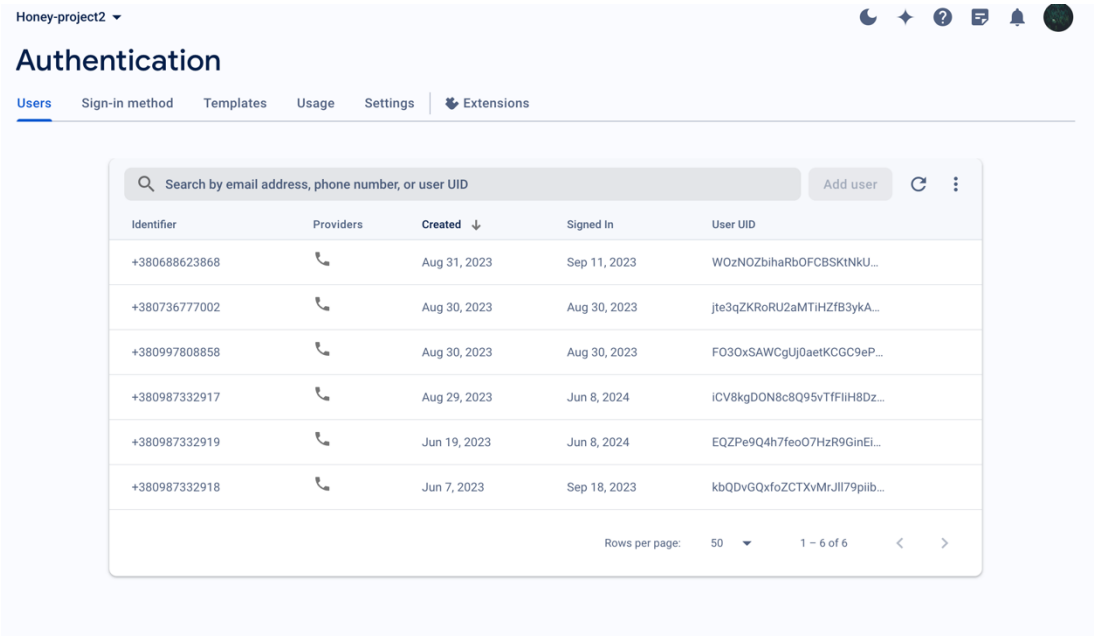

Рисунок 3.1 – Інтерфейс Firebase Authentication

Однією з ключових переваг Firebase аутентифікації є її швидкість і простота в налаштуванні. Сервіс дає можливість легко інтегрувати авторизацію через Firebase в застосунок всього лише за кілька кроків, використовуючи зручний SDK та документацію, яка чітко описує всі можливості і функціонал.

Це значно спрощує роботу з автентифікацією, звільняючи час для інших аспектів розробки застосунку.

Також Firebase забезпечує високий рівень безпеки та захисту персональних даних користувачів. Всі дані, включаючи паролі та іншу конфіденційну інформацію, передаються зашифровано за допомогою сучасних протоколів шифрування, що відповідає найвищим стандартам безпеки. Це особливо важливо в сучасному цифровому середовищі, де забезпечення конфіденційності даних користувачів є пріоритетом для багатьох компаній і розробників.

Крім того, Firebase Authentication підтримує різні методи авторизації, такі як електронна пошта та пароль, соціальні мережі (Google, Facebook, Twitter, GitHub) [29], OTP через номер телефону. Це розширює можливості застосунку, надаючи користувачам можливість вибору найзручнішого способу авторизації, відповідно до їхніх особистих уподобань і потреб.

Обираючи Firebase Authentication для застосунку пасіки "Божий Дар", було враховано його широкі можливості, високу надійність і підтримку від Google, що гарантує стабільну роботу і підтримку в майбутньому. Цей вибір дозволяє зосередитися на інших аспектах розробки застосунку, знаючи, що авторизація належним чином захищена і дотримується відповідних стандартів безпеки.

Наступним кроком був вибір типу авторизації через OTP за номером телефону. Це обумовлено тим, що цільова аудиторія застосунку складається зі старших людей, яким важко орієнтуватися в складних технологічних процесах, таких як робота з електронною поштою або запам'ятовування складних паролів. Використання OTP для авторизації через номер телефону є логічним вибором для забезпечення комфорту та зручності при користуванні застосунком.

Коли користувач відкриває застосунок вперше або виходить з облікового запису, він потрапляє на екран авторизації. Тут він має можливість ввести свій номер телефону. Після введення номеру телефону користувачу автоматично надсилається OTP код на введений номер. Отримавши код, користувач вводить

його у відповідне поле і натискає кнопку для авторизації. Якщо OTP код введений правильно, то відбувається вхід в застосунок, а користувач отримує доступ до основного функціоналу. У випадку, якщо користувач невірно введе OTP код, система відобразить відповідне повідомлення про помилку. Це повідомлення інформує користувача про те, що введений код неправильний, і пропонує спробувати ще раз. Така перевірка гарантує, що доступ до застосунку отримають лише ті користувачі, які ввели коректний код, що підвищує рівень безпеки. Код з повідомленням про невірно введений OTP зображено на рисунку 3.2.

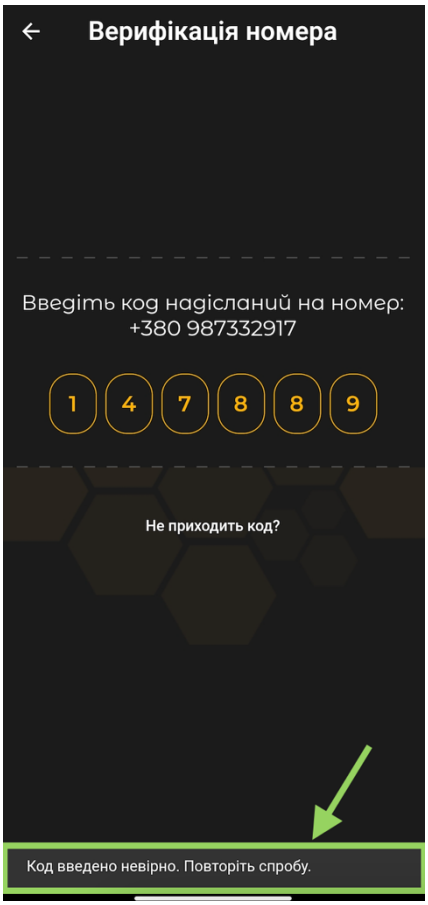

Рисунок 3.2 – Повідомленням про невірно введений OTP код

Якщо користувач ще не зареєстрований в системі, то при вході в застосунок йому відкривається екран реєстрації. Тут він заповнює додаткове поле для завершення реєстрації, а саме ім'я. Після введення цих даних

користувач стає повноцінним учасником системи з усіма правами та можливостями.

Якщо ж користувач вже зареєстрований і спробує знову ввести свій номер телефону для реєстрації, система автоматично виявить, що такий користувач вже існує. В такому випадку він буде повідомлений про це із відповідним повідомленням (див. рисунок 3.3), яке рекомендує використовувати функцію входу замість повторної реєстрації. Код методу, що перевіряє наявність користувача в БД наведений в лістингу 3.1. Повний лістинг модуля аутентифікації наведено в додатку Е.

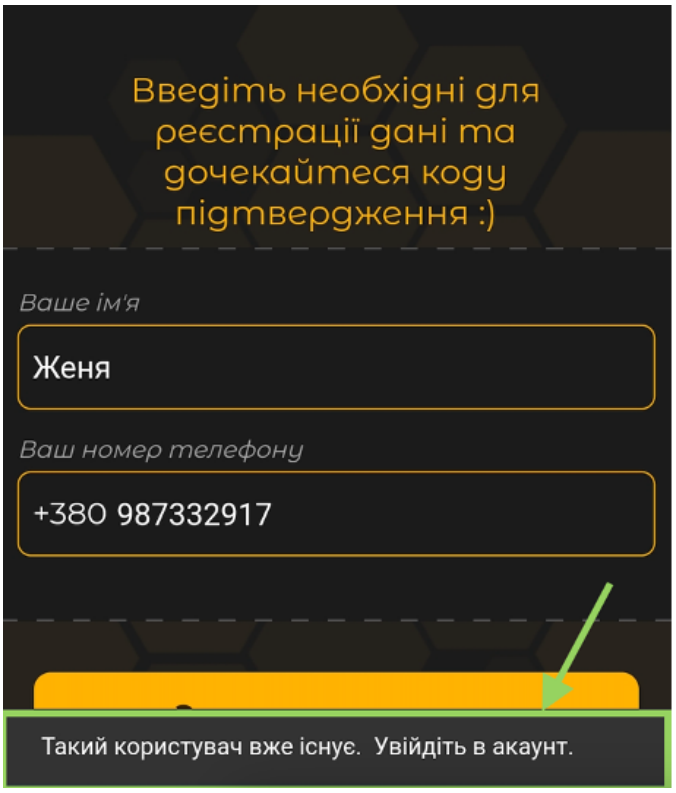

Рисунок 3.3 – Повідомлення про спробу повторної реєстрації

Лістинг 3.1 – Метод перевірки наявності користувача в БД

```
Future<bool> checkPhoneNumberExists(String phone) async {
  final CollectionReference userColection =
    FirebaseFirestore.instance.collection('users');
  QuerySnapshot querySnapshot =
    await userColection.where('phoneNumber',isEqualTo:
      phone).get();
  return querySnapshot.docs.isNotEmpty;
}
```

Для форми аутентифікації в застосунку використовуються валідатори, які перевіряють коректність введених користувачем даних. Це забезпечує правильність і безпеку інформації, що вводиться в поля форми. Валідатори перевіряють відповідність введених значень певним критеріям і, у разі виявлення помилок, виводять відповідні повідомлення.

Перший валідатор перевіряє поле з ім'ям користувача. Він переконується, що ім'я не є порожнім і містить принаймні два символи (див. лістинг 3.2).

Лістинг 3.2 – Валідатор імені користувача

```
validator: (value) {
  if (value!.isEmpty || value.length < 2) {
    return 'Невірно введене ім’я';
  }
  return null;
},
```

Другий валідатор перевіряє номер телефону. Він використовує регулярний вираз для перевірки, чи відповідає введений номер формату українського номера телефону. Якщо номер не відповідає цьому формату, виводиться повідомлення про невірно введений номер (див. лістинг 3.3).

Лістинг 3.3 – Валідатор номеру телефону користувача

```
validator: (value) {
  final regExp = RegExp(r'^\+380[0-9]{9}$');
  if (value == null||
      value.isEmpty ||
      !regExp.hasMatch('+380$value')) {
    return 'Невірний номер телефону!';
   }
   return null;
},
```

Ця система дозволяє забезпечити високий рівень зручності для користувачів, зокрема для старших людей, спрощуючи процес входу в застосунок і запобігаючи можливості помилкових дублікатів облікових записів. Використання номера телефону для OTP авторизації відкриває широкі

можливості для швидкого і безпечного доступу до застосунку, що є ключовим аспектом в контексті забезпечення відповідних стандартів безпеки даних та зручності для кінцевих користувачів.

3.2 Програмування основного функціоналу

Основний функціонал мобільного застосунку пасіки "Божий Дар" охоплює різноманітні можливості, які забезпечують ефективну взаємодію як для адміністратора, так і для клієнтів. Завдяки зручному та інтуїтивно зрозумілому інтерфейсу, адміністратор може легко керувати товарами, знижками та обробляти замовлення, тоді як клієнти можуть без зайвих зусиль додавати товари до корзини, оформлювати замовлення та редагувати свій профіль.

3.2.1 Функціонал адміністратора

Основний функціонал адміністратора включає декілька важливих аспектів, таких як додання товару, редагування товару, встановлення знижок на товар, управління сіткою кешбеку та обробка замовлень. Додавання нового товару до каталогу здійснюється через спеціальну форму, де адміністратор вводить усі необхідні дані, включаючи назву, опис, ціну та інші характеристики. Після заповнення форми ці дані зберігаються в базі даних (див. лістинг 3.4). Редагування існуючого товару здійснюється через аналогічну форму, де адміністратор може змінити будь-які необхідні параметри продукту.

Лістинг 3.4 – Метод додання товару до каталогу

```
Future<void> addProduct(Product product, File pickedImage) async {
    try {
      DocumentReference docRef =
          FirebaseFirestore.instance.collection('products').doc();
      Reference ref = FirebaseStorage.instance
          .ref()
          .child('product_images')
```

```
          .child('${docRef.id.toString()}.jpg');
      await docRef.set({
        'id': docRef.id,
        'title': product.title,
        'price': product.price,
        'imageUrl': imageUrl,
        'litersLeft': product.litersLeft,
        'shortDescription': product.productDescription,
        'isHoney': product.isHoney,
        'isDiscount': product.isDiscount ?? false,
        'discountPrice': product.discountPrice ?? 0,
        'discountPercentage': product.discountPercentage ?? 0,
      });
      Product newProduct = Product(
        id: docRef.id,
        title: product.title,
        price: product.price,
        imageUrl: imageUrl,
        litersLeft: product.litersLeft,
        productDescription: product.productDescription,
        isHoney: product.isHoney,
        isDiscount: product.isDiscount,
        discountPrice: product.discountPrice,
        discountPercentage: product.discountPercentage,
      );
      _items.add(newProduct);
      notifyListeners();
    } catch (e) {
      log(e);
    }
  }
```

Крім того, адміністратор має можливість встановлювати знижки на товари. Цей процес реалізований через окремий інтерфейс, де можна вказати розмір знижки та термін її дії. Управління сіткою кешбеків також є важливою функцією, що дозволяє налаштовувати відсоток кешбеку в залежності від загальної суми покупок клієнта. Метод _saveCashbackValues (див. лістинг 3.5) відповідає за збереження сітки кешбеку в базі даних. Цей метод здійснює декілька важливих кроків, щоб переконатися, що дані про кешбек правильно зберігаються в колекції 'admin' документа 'cashback' у Firestore.

Лістинг 3.5 – Метод оновлення сітки кешбеку

```
Future<void> _saveCashbackValues() async {
   if (!hasInternetConnection) {
```

```
      return;
    }
    try {
      setState(() {
        _isLoading = true;
      });
      if (_formKey.currentState!.validate()) {
        List<String> percentages = _percentageControllers
            .map((controller) => controller.text)
            .toList();
        List<String> amounts =
            _amountControllers.map((controller) =>
controller.text).toList();
        await FirebaseFirestore.instance
            .collection('admin')
            .doc('cashback')
            .set({
          'percentages': percentages,
          'amounts': amounts,
        });

        scaffoldContext.showSnackBar(
          const SnackBar(content: Text('Сітка кешбеку успішно
збережена')),
        );
        navContext.pop();
      }
    } catch (e) {
      log('Error fetching cashback values: $e');
    }
}
```

Після перевірки інтернет-з'єднання, метод змінює стан компонента, встановлюючи _isLoading в true, щоб вказати на початок процесу завантаження. Далі, якщо дані форми валідні, метод збирає введені адміністратором значення кешбеку з контролерів _percentageControllers та _amountControllers. Ці значення формуються у списки, які зберігаються в колекції Firestore. Метод виконує асинхронний запит до Firestore, зберігаючи сітку кешбеку у вигляді двох полів: 'percentages' та 'amounts'. Після успішного збереження даних показується повідомлення "Сітка кешбеку успішно збережена", а також виконується перехід до попереднього екрану.

Обробка замовлень адміністратором включає декілька етапів. Після того, як клієнт забирає замовлення з поштового відділення, адміністратор може

завершити це замовлення, запустивши метод confirmOrder (див. лістинг 3.6), що автоматично нараховує бонуси клієнтові відповідно до його програми лояльності. У випадку, якщо замовлення стає неактуальним, адміністратор має можливість видалити його з системи через спеціальний інтерфейс управління замовленнями.

Лістинг 3.6 – Метод підтвердження замовлення та нарахування кешбеку

```
void confirmOrder() async {
      try {
        if (!hasInternetConnection) {
          return;
        }
        final cashbackData = await getCashbackDataFromFirestore();
        isOrderFinished = true;
        final userId = order['userId'];
        final double orderAmount = order['totalAmount'];
        final userDoc = await FirebaseFirestore.instance
            .collection('users')
            .doc(userId)
            .get();
        final userOrdersTotalAmount =
            userDoc.data()?['ordersTotalAmount'] ?? 0.0;
        int maxAmountIndex = -1;
        for (int i = 0; i < cashbackData['amounts'].length; i++) {
               if (cashbackData['amounts'][i] <=
     userOrdersTotalAmount) {
            maxAmountIndex = i;
          } else {
            break;
          }
        }
        int percentage = maxAmountIndex >= 0
            ? cashbackData['percentages'][maxAmountIndex]
            : 0;
        double bonusAmount = (percentage / 100) * orderAmount;
        updateUserTotalAmountAndBonuses(userId, orderAmount,
bonusAmount);
        updateIsFinishedStatus(true);
      } catch (e) {
        log(e);
      }
}
```

Метод отримує значення сітки кешбеку з Firestore колекції. Далі відбувається пошук документа клієнта з колекції 'users' у Firestore. З цього

документа отримується загальна сума замовлень клієнта. Після цього визначається максимальний індекс суми, яка підходить під поточну загальну суму замовлень користувача, і відповідний відсоток кешбеку.

Метод розраховує бонусну суму bonusAmount як відсоток від суми замовлення orderAmount і викликає функції updateUserTotalAmountAndBonuses для оновлення загальної суми замовлень та бонусів клієнта, а також updateIsFinishedStatus для оновлення статусу замовлення. Такий підхід забезпечує високу ефективність роботи та зручність керування замовленнями, що є ключовим фактором успішного функціонування застосунку.

3.2.2 Функціонал клієнта

Основний функціонал клієнта також включає кілька важливих операцій, які роблять використання застосунку простим та зручним. Клієнт може додавати товари до корзини, що дозволяє зібрати всі необхідні продукти перед оформленням замовлення. Цей процес здійснюється через інтуїтивно зрозумілий інтерфейс, де кожен доданий товар автоматично зберігається в кошику користувача (див. лістинг 3.7).

Лістинг 3.7 – Метод додання товару в корзину

```
void addHalfLiterToCart({required Product product}) {
    if (_items.containsKey(product.id)) {
      _items.update(
          product.id,
          (existingCartItem) => CartItemModel(
              id: existingCartItem.id,
              title: existingCartItem.title,
              price: existingCartItem.price,
              liters: existingCartItem.liters + 0.5,
              imageUrl: existingCartItem.imageUrl));
    } else {
      _items.putIfAbsent(
          product.id,
          () => CartItemModel(
              id: product.id,
              title: product.title,
              price: product.discountPrice == 0
```

```
                    ? product.price
                    : product.discountPrice!,
                liters: 0.5,
                imageUrl: product.imageUrl));
    }
  notifyListeners();
}
```

Після додавання товарів до корзини, клієнт може перейти до оформлення замовлення, де вводить необхідні дані для доставки. Підтвердивши замовлення, ці дані передаються на сервер для подальшої обробки (див. лістинг 3.8).

Лістинг 3.7 – Метод оформлення замовлення клієнтом

```
Future<void> _submitForm(BuildContext context) async {
   if (_formkey.currentState!.validate()) {
      setState(() {
       _isLoading = true;
      });
      if (!hasInternetConnection || user == null) {
        return;
      }
      await _saveUserDataToFirestore();
      if (!widget.isEditProfile) {
        final orderData = _buildOrderData(cartProvider, user);
        await _saveOrderToFirestore(orderData);
        await _saveUserDataToFirestore();
        _resetProductsdata();
        _resetUserBonuses();
      }

      navigatorContext.pushAndRemoveUntil(
        MaterialPageRoute(
          builder: (context) =>
              const UserMainScreen(selectedBottomNavBarIndex: 0),
        ),
        (route) => false,
      );
      setState(() {
        _isLoading = false;
      });
   }
}
```

Метод _submitForm відповідає за збереження інформації про користувача та підтвердження замовлення. Спочатку він перевіряє валідність форми; якщо форма невалідна, інтернет-з'єднання відсутнє або користувач не

аутентифікований, функція припиняє виконання. Далі метод зберігає дані користувача. Якщо клієнт не перебуває на екрані редагування профілю, створюються дані замовлення, які потім зберігаються у Firestore.

Таким чином після першого замовлення введені клієнтом дані для оформлення замовлення автоматично зберігаються в його профілі, що дозволяє уникнути повторного введення інформації у майбутньому. Клієнт може редагувати ці дані у будь-який момент через свій профіль, що забезпечує високу гнучкість і зручність використання.

Для реалізації зазначеного функціоналу використовуються сучасні технології та бібліотеки. Наприклад, Flutter забезпечує створення зручного та швидкого користувацького інтерфейсу, а Firebase використовується для управління даними. Firebase Firestore застосовується для зберігання та отримання інформації про товари, замовлення та профілі користувачів, що дозволяє забезпечити високу швидкість та надійність роботи застосунку.

Отже, застосунок пасіки "Божий Дар" пропонує широкий спектр функціональних можливостей для адміністраторів та клієнтів, використовуючи сучасні технології для забезпечення високої якості сервісу. Всі процеси автоматизовані та інтегровані в єдину систему, що дозволяє ефективно керувати пасікою, обробляти замовлення та надавати клієнтам зручний інструмент для замовлень.

3.3 Тестування застосунку

Тестування є важливим етапом розробки будь-якого програмного продукту, особливо мобільного застосунку, який буде використовуватись широким колом користувачів. Основною метою тестування є виявлення помилок та забезпечення того, що всі функціональні можливості працюють відповідно до вимог. Тестування мобільного застосунку для пасіки “Божий Дар” включало кілька етапів мануального тестування, спрямованих на перевірку різних аспектів функціоналу.

Спершу було проведено тестування основного функціоналу застосунку, такого як авторизація та реєстрація користувачів. Було важливо переконатися, що користувачі можуть успішно увійти до системи, використовуючи свої облікові записи. Адміністраторська частина програми також була ретельно перевірена, щоб впевнитись, що вона доступна тільки тим користувачам, які мають відповідні права доступу. Для цього використовувались тестові облікові записи адміністратора та звичайного користувача, щоб перевірити правильність розгалуження доступу до різних частин застосунку.

Після успішного тестування авторизації, увагу було зосереджено на функціоналі управління товарами. Тестування включало додавання нових товарів до каталогу, редагування існуючих товарів, видалення товарів та перевірку наявності товарів у каталозі. Цей процес вимагав ретельної перевірки, щоб переконатися, що всі зміни правильно відображаються у користувацькому інтерфейсі та базі даних Firestore. Було також перевірено, що зображення товарів успішно завантажуються та зберігаються у Firebase Storage.

Тестування секції замовлень включало перевірку можливості користувачів створювати нові замовлення, переглядати статуси існуючих замовлень та їх історію. Для цього використовувались різні сценарії, включаючи додавання товарів до корзини, оформлення замовлення та перевірку оновлення статусів замовлень у реальному часі. Особлива увага приділялась перевірці коректності роботи функції нарахування кешбеку, яка базується на загальній витраченій сумі.

Функціонал адміністратора був протестований для забезпечення можливості керування знижками та перегляду користувацьких облікових записів. Тестування включало створення та редагування знижок, перевірку правильності їх відображення у користувацькому інтерфейсі та базі даних. Крім того, було перевірено можливість адміністратора переглядати та редагувати інформацію про користувачів, відстежувати їхню активність та забезпечувати підтримку.

Тестування користувацького інтерфейсу включало перевірку всіх екранних форм на предмет коректного відображення інформації та зручності використання. Особлива увага приділялась головному екрану, який є ключовим елементом взаємодії з користувачем. Було перевірено, що всі секції головного екрану (навігація, каталог товарів та перехід до корзини) працюють належним чином та у користувачів є швидкий доступ до основних функцій застосунку.

Також було проведено тестування секції профілю користувача, яка включає перегляд бонусів, редагування інформації для доставки та перегляд історії замовлень. Було перевірено, що користувач може успішно оновлювати свої дані, і що зміни правильно відображаються у базі даних. Було також протестовано, що історія замовлень відображається коректно, з усіма необхідними деталями.

Фінальний етап тестування включав перевірку контактної інформації про магазин та адміністратора. Було важливо переконатися, що користувачі мають доступ до актуальної та правильної контактної інформації, яка дозволяє їм зв’язатися з адміністрацією пасіки у разі необхідності.

Загалом, мануальне тестування застосунку для пасіки “Божий Дар” охоплювало всі основні функціональні компоненти, включаючи авторизацію, управління товарами, замовленнями та знижками, а також перевірку користувацького інтерфейсу. Кожен етап тестування був спрямований на забезпечення стабільності роботи застосунку, відповідності його функціональних можливостей вимогам, а також надання користувачам зручного та інтуїтивно зрозумілого інтерфейсу.

3.4 Висновок до третього розділу

У третьому розділі було реалізовано систему аутентифікації для мобільного застосунку пасіки "Божий Дар" за допомогою Firebase Authentication, що забезпечила простий, надійний та безпечний спосіб управління автентифікацією користувачів. Вибір Firebase Authentication

обґрунтовано його широкими можливостями, високою надійністю, підтримкою різних методів авторизації та інтеграцією з іншими сервісами Firebase. Особливу увагу було приділено використанню OTP-авторизації за номером телефону, що значно спрощує процес входу в застосунок для цільової аудиторії, яка включає старших людей.

Крім аутентифікації було реалізовано функціонал застосунку для адміністратора і клієнтів. Адміністратори можуть додавати нові товари до каталогу, редагувати існуючі товари, встановлювати знижки та керувати сіткою кешбеку. Обробка замовлень включає можливість завершення замовлення з автоматичним нарахуванням бонусів відповідно до програми лояльності. Для клієнтів основний функціонал включає можливість додавання товарів до корзини, оформлення замовлення та редагування профілю. Дані, введені під час першого замовлення, автоматично зберігаються у профілі користувача, що дозволяє уникнути повторного введення інформації у майбутньому. Клієнти можуть редагувати ці дані у будь-який момент через свій профіль.

Тестування застосунку включало мануальну перевірку основних функцій, таких як авторизація, управління товарами та замовленнями, нарахування кешбеку та адміністраторські функції. Особливу увагу приділено перевірці зручності інтерфейсу користувача та коректності роботи з Firebase для зберігання даних і зображень. Тестування забезпечило високу якість роботи застосунку та відповідність його функціональних можливостей вимогам.

Використання сучасних технологій, зокрема Flutter для створення інтерфейсу та Firebase для управління даними, забезпечило зручність, швидкість та надійність роботи застосунку.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Безпека життєдіяльності: Комплексний аналіз життєдіяльності людини

Комплексний аналіз життєдіяльності людини в умовах розробки та використання крос-платформних мобільних застосунків має важливе значення для забезпечення її ефективності та безпеки. В умовах сучасного світу, де цифровізація і автоматизація відіграють все більшу роль, необхідно враховувати всі аспекти взаємодії людини з технологіями та навколишнім середовищем [30]. Під час роботи з мобільними застосунками користувачі стикаються з різними викликами, включаючи фізичне, психологічне та соціальне навантаження. Комплексний аналіз дозволяє виявити фактори, які можуть впливати на здоров'я та безпеку користувачів, а також розробити ефективні стратегії для їхнього подолання.

Важливою складовою комплексного аналізу є вивчення фізичного навантаження, яке зазнають користувачі під час роботи з мобільними застосунками. Тривале використання смартфонів або планшетів може призводити до м'язового напруження, болю в шиї та спині, а також до проблем із зором [31]. Для мінімізації цих негативних впливів необхідно враховувати ергономічні принципи під час розробки інтерфейсу та функціоналу застосунку. Ергономічні принципи включають зручність використання, легкодоступність елементів управління, адаптацію під різні типи пристроїв та зменшення навантаження на зір за допомогою оптимальних колірних схем та розмірів шрифту.

Психологічний аспект також є важливою складовою комплексного аналізу. Використання нових технологій може викликати стрес, особливо у користувачів, які не звикли до цифрових інновацій. Для подолання цього необхідно забезпечити зручний та інтуїтивно зрозумілий інтерфейс, а також надати користувачам достатню підтримку та навчання.

Соціальний аспект життєдіяльності також не можна залишати без уваги. Використання мобільних застосунків для управління пасікою вимагає взаємодії з різними стейкхолдерами, включаючи клієнтів, постачальників та співробітників. Ефективна комунікація та співпраця є ключовими для забезпечення безпеки та успіху діяльності. Комплексний аналіз життєдіяльності людини дозволяє врахувати всі ці аспекти і розробити стратегії для покращення здоров'я, безпеки та продуктивності користувачів. Впровадження таких заходів сприятиме більш ефективному використанню технологій та зменшенню ризиків, пов'язаних з їхнім застосуванням.

Також слід враховувати вплив навколишнього середовища на життєдіяльність людини. Зміни клімату, екологічні проблеми та інші фактори можуть мати значний вплив на здоров'я та безпеку працівників. Тому важливо проводити регулярний моніторинг умов праці та вживати заходів для їх поліпшення. Це може включати забезпечення належної вентиляції, контроль рівня шуму, освітлення та інших параметрів, що впливають на комфорт і безпеку роботи [32].

Загалом, комплексний аналіз життєдіяльності людини в умовах використання крос-платформних мобільних застосунків дозволяє виявити потенційні ризики та розробити ефективні заходи для їхнього зменшення. Це включає врахування фізичних, психологічних, соціальних та екологічних факторів, які можуть впливати на здоров'я та безпеку користувачів. Впровадження таких заходів сприятиме підвищенню ефективності роботи, зниженню рівня стресу та забезпеченню високого рівня безпеки на робочому місці.

4.2 Основи охорони праці: Психологічні чинники небезпеки

Психологічні чинники небезпеки відіграють важливу роль в забезпеченні охорони праці. Робота в умовах високих інформаційних навантажень, багатозадачності та необхідності швидко адаптуватися до нових технологій

може викликати стрес у працівників. Важливо розуміти, як психологічні чинники можуть впливати на безпеку та ефективність праці, щоб розробити стратегії для їхнього подолання [33].

Стрес, викликаний роботою, може мати різні причини, включаючи перевантаження інформацією, недостатню підтримку з боку колег або керівництва, а також невпевненість у власних силах. Ці фактори можуть призводити до зниження концентрації, підвищеної втоми та зростання кількості помилок, що, в свою чергу, підвищує ризик нещасних випадків на робочому місці. Для зменшення впливу стресу на працівників необхідно впроваджувати програми підтримки, такі як навчання методам управління стресом, організація регулярних перерв та забезпечення комфортних умов праці.

Також важливо враховувати індивідуальні психологічні особливості працівників. Різні люди мають різні рівні стійкості до стресу, здатності до багатозадачності та адаптивності. Розуміння цих відмінностей дозволяє більш ефективно розподіляти завдання та ресурси, а також надавати індивідуальну підтримку працівникам, які цього потребують. Крім того, робота в колективі, де підтримується атмосфера довіри та взаємодопомоги, сприяє зменшенню рівня стресу та підвищенню загального рівня безпеки [34].

Крос-платформні мобільні застосунки можуть бути корисними інструментами для покращення психологічного клімату на робочому місці. Вони можуть надавати можливість для швидкої та зручної комунікації, допомагати в організації робочого процесу та забезпечувати доступ до необхідної інформації в будь-який час і в будь-якому місці. Це сприяє зниженню рівня стресу та підвищенню продуктивності праці. Використання таких застосунків також дозволяє керівникам ефективно контролювати робочі процеси та швидко реагувати на будь-які проблеми, що виникають.

Психологічні чинники небезпеки також можуть впливати на взаємодію між працівниками. Недостатнє спілкування або конфлікти в колективі можуть створювати напружену атмосферу, що негативно впливає на безпеку праці. Для подолання цих проблем необхідно розвивати навички ефективної комунікації,

вирішення конфліктів та командної роботи. Психологічна підтримка та турбота про психічне здоров'я працівників повинні бути невід'ємною частиною системи охорони праці. Це включає регулярні оцінки психологічного стану працівників, організацію тренінгів з управління стресом, а також створення умов для відпочинку. Інтеграція цих заходів у щоденну практику сприятиме збереженню здоров'я працівників, підвищенню їхньої продуктивності та забезпеченню високого рівня безпеки на робочому місці.

Крім того, важливо враховувати вплив зовнішніх чинників, таких як сімейні проблеми, фінансові труднощі або особисті конфлікти, які можуть впливати на психологічний стан працівників. Роботодавці повинні бути готові надавати необхідну підтримку та допомогу, щоб мінімізувати негативний вплив таких факторів на роботу працівників. Це може включати консультації з психологом, організацію програм підтримки працівників та створення умов для балансу між роботою та особистим життям [35].

У цілому, психологічні чинники небезпеки мають значний вплив на безпеку та ефективність праці. Врахування цих факторів і впровадження відповідних заходів дозволяє забезпечити комфортні умови роботи, знизити рівень стресу та підвищити загальний рівень безпеки. Це, в свою чергу, сприяє збереженню здоров'я працівників, підвищенню їхньої продуктивності та забезпеченню високого рівня охорони праці.

4.3 Висновок до четвертого розділу

У четвертому розділі описано важливі аспекти безпеки життєдіяльності та охорони праці, які є ключовими для успішного впровадження крос-платформного мобільного застосунку для пасіки “Божий Дар” на основі Flutter. Дослідження зосереджено на забезпеченні фізичної, психологічної та соціальної безпеки користувачів у процесі роботи з технологіями.

Аналіз життєдіяльності в умовах використання мобільних застосунків виявляє фізичні навантаження, пов’язані з тривалим використанням пристроїв,

що можуть призводити до м’язового напруження та проблем із зором. Врахування ергономічних принципів під час розробки інтерфейсу зменшує навантаження на зір та підвищує зручність використання.

Психологічний аспект також критично важливий. Використання нових технологій може викликати стрес, що впливає на продуктивність та безпеку праці. Необхідно забезпечити інтуїтивно зрозумілий інтерфейс та підтримку користувачів через навчальні матеріали та служби підтримки.

Соціальний аспект охоплює взаємодію з різними стейкхолдерами, що вимагає ефективної комунікації та співпраці для забезпечення безпеки та успіху діяльності. Це включає моніторинг умов праці та впровадження заходів для їхнього поліпшення, таких як належна вентиляція та контроль рівня шуму.

Загалом, цей розділ висвітлює важливість врахування фізичних, психологічних та соціальних факторів для забезпечення безпеки та ефективності праці при використанні крос-платформних мобільних застосунків.

ВИСНОВКИ

В першому розділі кваліфікаційної роботи освітнього рівня «Бакалавр»:

- подано аналітичний огляд існуючих рішень у сфері мобільних застосунків;
- розглянуто технічне завдання, яке включає найменування, область застосування та призначення розробки;
- висвітлено вимоги до функціоналу застосунку, етапи розробки та порядок тестування і прийому;
- проаналізовано основні аспекти, які необхідно врахувати під час розробки крос-платформного застосунку.

В другому розділі кваліфікаційної роботи:

- досліджено архітектуру застосунку, включаючи визначення загальної структури;
- обґрунтовано архітектуру представлення, яка забезпечує зручність користування та підтримку різних платформ;
- сформовано дизайн застосунку, включаючи візуальні елементи та користувацький інтерфейс, що сприяє зручній взаємодії користувачів із застосунком;
- розроблено базу даних та моделі, що забезпечують зберігання, обробку та управління даними користувачів.

В третьому розділі кваліфікаційної роботи:

- розроблено систему аутентифікації користувачів;
- розроблено функціонал для адміністратора пасіки та клієнтів, що використовують застосунок;
- протестовано та налагоджено застосунок.

У розділі «Безпека життєдіяльності, основи охорони праці»:

- висвітлено комплексний аналіз життєдіяльності людини у контексті використання мобільних застосунків;

- розглянуто психологічні чинники небезпеки, що можуть виникати під час взаємодії користувачів з мобільними пристроями.

Дипломна робота демонструє високий рівень аналізу, проектування та реалізації мобільного застосунку для пасіки "Божий Дар", що включає застосунок для адміністратора з управління замовленнями та зручний інтернет-магазин для клієнтів. Результати досліджень підтверджують його важливість та практичну цінність для покращення управління пасікою та забезпечення зручності покупцям у виборі та покупці меду через мобільний застосунок.

ПЕРЕЛІК ДЖЕРЕЛ

1 Strutynska, I., Dmytrotsa, L., Kozbur, H., and Dmytruk, U. (2020) 'Key enablers contributing to the development of the digital economy', in Proceedings of the International Scientific and Practical Conference 'Digital Economy as a Factor of Innovative Development of Society', 11 November 2020, Ternopil: TNTU, pp. 43-45. Available at: http://elartu.tntu.edu.ua/handle/lib/34051.

2 Lesiv, V. and Dmytrotsa, L. (2020) 'Profile of small and medium-sized enterprises in Europe', in Proceedings of the International Scientific and Practical Conference 'Digital Economy as a Factor of Innovative Development of Society', 11 November 2020, Ternopil: TNTU, pp. 131-132. Available at: http://elartu.tntu.edu.ua/handle/lib/34095.

3 Stsibailo, Y. and Dmytrotsa, L. (2022) 'Unreadiness factors of companies for industry 4.0', in Proceedings of the II International Scientific Conference 'Business Transformation for Sustainable Future: Research, Digitalization and Innovation', 23-24 November 2022, Ternopil: FOP Palianytsia V.A., pp. 70-72. Available at: http://elartu.tntu.edu.ua/handle/lib/39780.

4 Techopedia Mobile Application (Mobile App). Available at: https://www.techopedia.com/definition/2953/mobile-application-mobile-app (Accessed: 3 June 2024).

5 ELEKS The Advantages of Cross-Platform Mobile App Development. Available at: https://eleks.com/types-of-software-development/the-advantages-of-cross-platform-mobile-app-development/ (Accessed: 3 June 2024).

6 NIX Solutions 'Top 4 mobile app development challenges and their solutions', NIX Solutions. Available at: https://www.nixsolutions.com/blog/top-4-mobile-app-development-challenges-solutions/ (Accessed: 3 June 2024).

7 Kasera, Y. (2021) 'How to use Flutter to build cross-platform mobile apps', Medium. Available at: https://yash-kasera.medium.com/how-to-use-flutter-to-build-cross-platform-mobile-apps-d3502531be12 (Accessed: 4 June 2024).

8 Awais (2021) 'Dart: A powerful programming language', Medium. Available at: https://awaiscs.medium.com/dart-a-powerful-programming-language-cd5c7d81257f (Accessed: 4 June 2024).

9 The Droids on Roids Team (2023) 'Flutter vs Kotlin – The ultimate comparison', The Droids on Roids Blog. Available at: https://www.thedroidsonroids.com/blog/flutter-vs-kotlin-comparison (Accessed: 4 June 2024).

10 Mobikul (2021) 'JIT and AOT Compilers in Dart', Mobikul Blog. Available at: https://mobikul.com/jit-and-aot-compilers-in-dart/ (Accessed: 4 June 2024).

11 GeekBlogger411 'Dart Programming Language', Issuu. Available at: https://issuu.com/geekblogger411/docs/dart_programming_language (Accessed: 4 June 2024).

12 Secoda 'Data Security vs Data Privacy', Secoda. Available at: https://www.secoda.co/learn/data-security-vs-data-privacy (Accessed: 5 June 2024).

13 Meteor Forums (2021) 'Performance and stability problems at scale', Meteor Forums. Available at: https://forums.meteor.com/t/performance-and-stability-problems-at-scale/56230/5 (Accessed: 5 June 2024).

14 Namtheartist95 (2020) 'What is backend development?', Medium. Available at: https://medium.com/@namtheartist95/what-is-backend-development-bcc6a15f8472 (Accessed: 5 June 2024).

15 Simform (2023) 'Mobile Application Architecture: Choosing the right strategy', Simform. Available at: https://www.simform.com/blog/mobile-application-architecture/ (Accessed: 5 June 2024).

16 Hamber (2022) 'Flutter State Management Guide', Medium. Available at: https://medium.com/@hamber/flutter-state-management-guide-697ea0bc16b3 (Accessed: 6 June 2024).

17 Herrarte, J. (2021) 'What is State Management and Why is it Important?', Dev.to. Available at: https://dev.to/jennherrarte/what-is-state-management-and-why-is-it-important-1i8d (Accessed: 6 June 2024).

18 Flutter 'State management options', Flutter Documentation. Available at: https://docs.flutter.dev/data-and-backend/state-mgmt/options (Accessed: 6 June 2024).

19 DhiWise 'Achieving optimal performance with Flutter provider state management', DhiWise Blog. Available at: https://www.dhiwise.com/post/achieving-optimal-performance-with-flutter-provider-state-management (Accessed: 8 June 2024).

20 Stfalcon 'Ключові компоненти успішних мобільних додатків', Stfalcon Blog. Available at: https://stfalcon.com/uk/blog/post/key-components-of-successful-mobile-apps (Accessed: 8 June 2024).

21 Interaction Design Foundation 'UI Color Palette', Interaction Design Foundation. Available at: https://www.interaction-design.org/literature/article/ui-color-palette (Accessed: 10 June 2024).

22 BSS 'The Advantages of Figma's Community-Driven Features', BSS. Available at: https://bss.mc/the-advantages-of-figmas-community-driven-features/ (Accessed: 10 June 2024).

23 Cleveroad 'How to Select Fonts', Cleveroad. Available at: https://www.cleveroad.com/blog/how-to-select-fonts/ (Accessed: 10 June 2024).

24 Hosen, I. 'Importance of Logo Design for Business Branding and Recognition', LinkedIn. Available at: https://www.linkedin.com/pulse/importance-logo-design-business-branding-recognition-imran-hosen/ (Accessed: 10 June 2024).

25 Firebase. 'Data model', Firestore Documentation. Available at: https://firebase.google.com/docs/firestore/data-model (Accessed: 12 June 2024).

26 ImageKit. 'Optimize and Resize Images in Firebase Storage', ImageKit Blog. Available at: https://imagekit.io/blog/optimize-and-resize-images-in-firebase-storage/ (Accessed: 12 June 2024).

27 Stfalcon. 'Creating Mobile App Interface Design', Stfalcon Blog. Available at: https://stfalcon.com/en/blog/post/creating-mobile-app-interface-design (Accessed: 12 June 2024).

28 Attract Group. 'Mobile App Development Best Practices', Attract Group Blog. Available at: https://attractgroup.com/blog/mobile-app-development-best-practices/ (Accessed: 13 June 2024).

29 Ionic Themes (2023) 'Simplify User Identity with Firebase Authentication', Ionic Themes Blog. Available at: https://ionicthemes.com/tutorials/simplify-user-identity-with-firebase-authentication (Accessed: 13 June 2024).

30 Іщейкіна Ю. О., Філатова В. Л., Нечепасва Л. В., Буря Л. В., Саргош О. Д. та Бичко М. В., 2013. Основи безпеки життєдіяльності людини. Полтава: Вид-во ПО “ІШидкоДрук”.

31 Харківський національний університет міського господарства імені О.М. Бекетова, Департамент цивільного захисту Харківської обласної військової адміністрації та Loughborough University, 2023. Актуальні питання охорони праці у контексті сталого розвитку та європейської інтеграції України: матеріали IV Міжнар. наук.-практ. інтернет-конф., Харків, 09-11 листоп. 2023 р. Харків: ХНУМГ ім. О. М. Бекетова.

32 Гусєв, А. М., 2023. Охорона праці та цивільний захист. Національний технічний університет України “Київський політехнічний інститут імені Ігоря Сікорського”.

33 Арламов, О. Ю., 2018. Безпека життєдіяльності та цивільний захист. Київ: Інститут енергозбереження та енергоменеджменту.

34 Березюк О.В., Лемешев М. С. 2011. Безпека життєдіяльності. Вінниця: Вінницикий національний технічний університет.

35 Березуцький, В.В., Васьковець, Л.А., Вершиніна, Н.П., Горбенко, В.В., Лісогор, О.С., Райко, В.Ф., 2005. Безпека життєдіяльності. Харків: Видавництво «Факт».

ДОДАТКИ

Додаток А

Лістинг файлу «product_model.dart»

```
class ProductModel {
  String productId;
  String productName;
  double productPrice;
  String productImageURL;
  int productQuantity;
  String productDescription;
  bool isHoney;
  bool? isDiscount;
  int? discountPercentage;
  double? discountPrice;

  ProductModel({
    required this.productId,
    required this.productName,
    required this.productPrice,
    required this.productImageURL,
    required this.productQuantity,
    required this.productDescription,
    required this.isHoney,
    this.isDiscount,
    this.discountPercentage,
    this.discountPrice,
  });

  double get calculatedNewPrice {
    if (isDiscount == true) {
      double discountPercentage2 = (100 - discountPercentage!) /
100;
      double newPrice = (productPrice * discountPercentage2);
      return newPrice;
    } else {
      return productPrice;
    }
  }

  Map<String, dynamic> toMap() {
    return {
      'productId': productId,
      'productName': productName,
      'productPrice': productPrice,
      'productImageURL': productImageURL,
      'productQuantity': productQuantity,
      'productDescription': productDescription,
      'isHoney': isHoney,
      'isDiscount': isDiscount,
      'discountPercentage': discountPercentage,
      'discountPrice': discountPrice,
    };
  }
```

```
  factory ProductModel.fromMap(Map<String, dynamic> map) {
    return ProductModel(
      productId: map['productId'] ?? '',
      productName: map['productName'] ?? '',
      productPrice: (map['productPrice']) ?? 0,
      productImageURL: map['productImageURL'] ?? '',
      productQuantity: map['productQuantity'] ?? 0,
      productDescription: map['productDescription'] ?? '',
      isHoney: map['isHoney'] ?? true,
      isDiscount: map['isDiscount'] ?? false,
      discountPercentage: map['discountPercentage'] ?? 0,
      discountPrice: map['discountPrice'] ?? 0,
    );
  }
}
```

Додаток Б

Лістинг файлу «order_model.dart»

```
import 'package:intl/intl.dart';

class OrderModel {
  static final DateFormat dateFormat = DateFormat("yyyy-MM-dd
HH:mm");

  final String userFullName;
  final String userPhoneNumber;
  final String userAddress;
  final String selectedWayOfDelivery;
  final String userPostOfficeData;
  final String userComment;
  final String userId;
  final double totalSumOfOrder;
  final num usedBonuses;
  final DateTime createdAt;
  final bool isFinished;
  final bool isVisibleForAdmin;
  final String? salePoint;
  final Map<String, CartItemModel> products;

  OrderModel({
    required this.userFullName,
    required this.userPhoneNumber,
    required this.userAddress,
    required this.selectedWayOfDelivery,
    required this.userPostOfficeData,
    required this.userComment,
    required this.userId,
    required this.totalSumOfOrder,
    required this.usedBonuses,
    required this.createdAt,
    required this.isFinished,
    required this.isVisibleForAdmin,
    required this.products,
    this.salePoint,
  });

  Map<String, dynamic> toMap() {
    String formattedCreatedAt = dateFormat.format(createdAt);

    return {
      'userFullName': userFullName,
      'userPhoneNumber': userPhoneNumber,
      'userAddress': userAddress,
      'selectedWayOfDelivery': selectedWayOfDelivery,
      'userPostOfficeData': userPostOfficeData,
      'userComment': userComment,
```

```
      'userId': userId,
      'totalSumOfOrder': totalSumOfOrder,
      'usedBonuses': usedBonuses,
      'createdAt': formattedCreatedAt,
      'isFinished': isFinished,
      'isVisibleForAdmin': isVisibleForAdmin,
      'products': products,
      'salePoint': salePoint,
    };
  }
}
```

Додаток В

Лістинг файлу «profile_data_model.dart»

```
class ProfileDataModel {
  String userId;
  String? phoneNumber;
  String? deliveryPhoneNumber;
  String? nameOfAccount;
  String? fullName;
  String? userImageURL;
  String? userAddress;
  String? userPostOfficeData;
  String? selectedWayOfDelivery;
  double? ordersTotalSum;
  int? bonuses;

  ProfileDataModel({
    required this.userId,
    required this.phoneNumber,
    required this.deliveryPhoneNumber,
    required this.fullName,
    required this.nameOfAccount,
    required this.userImageURL,
    required this.userAddress,
    required this.selectedWayOfDelivery,
    required this.userPostOfficeData,
    required this.ordersTotalSum,
    required this.bonuses,
  });

  Map<String, dynamic> toMap() {
    return {
      'userId': userId,
      'phoneNumber': phoneNumber,
      'deliveryPhoneNumber': deliveryPhoneNumber,
      'fullName': fullName,
      'nameOfAccount': nameOfAccount,
      'userImageURL': userImageURL,
      'userAddress': userAddress,
      'selectedWayOfDelivery': selectedWayOfDelivery,
      'userPostOfficeData': userPostOfficeData,
      'ordersTotalSum': ordersTotalSum,
      'bonuses': bonuses,
    };
  }

  factory ProfileDataModel.fromMap(Map<String, dynamic> map) {
    return ProfileDataModel(
      userId: map['userId'] as String,
      phoneNumber: map['phoneNumber'] as String,
      deliveryPhoneNumber: map['deliveryPhoneNumber'] as String,
```

```
      fullName: map['fullName'] as String?,
      nameOfAccount: map['nameOfAccount'] as String?,
      userImageURL: map['userImageURL'] as String?,
      userAddress: map['userAddress'] as String?,
      selectedWayOfDelivery: map['selectedWayOfDelivery'] as
String?,
      userPostOfficeData: map['userPostOfficeData'] as String?,
      ordersTotalSum: map['ordersTotalSum'] as double?,
      bonuses: map['bonuses'] as int?,
    );
  }
}
```

Додаток Д

Лістинг файлу «admin_data_model.dart»

```
class AdminDataModel {
  final String adminPhoneNumber;
  final String adminName;
  final String adminEmail;
  final String adminImageURL;
  final String certificateURL;
  final String aboutStore;
  final List<Map<String, dynamic>> salePoints;

  AdminDataModel({
    required this.adminPhoneNumber,
    required this.adminName,
    required this.adminEmail,
    required this.adminImageURL,
    required this.certificateURL,
    required this.aboutStore,
    required this.salePoints,
  });

  factory AdminDataModel.fromMap(Map<String, dynamic> map) {
    final List<Map<String, dynamic>> salePointsList =
        List<Map<String, dynamic>>.from(map['salePoints'] ?? []);

    return AdminDataModel(
      adminPhoneNumber: map['adminPhoneNumber'] ?? '',
      adminName: map['adminName'] ?? '',
      adminEmail: map['adminEmail'] ?? '',
      adminImageURL: map['adminImageURL'] ?? '',
      certificateURL: map['certificateURL'] ?? '',
      aboutStore: map['aboutStore'] ?? '',
      salePoints: salePointsList,
    );
  }

  AdminDataModel.initial()
      : adminPhoneNumber = '',
        adminName = '',
        adminEmail = '',
        adminImageURL = '',
        certificateURL = '',
        aboutStore = '',
        salePoints = [];

  Map<String, dynamic> toMap() {
    return {
      'adminPhoneNumber': adminPhoneNumber,
      'adminName': adminName,
      'adminEmail': adminEmail,
```

```
      'adminImageURL': adminImageURL,
      'certificateURL': certificateURL,
      'aboutStore': aboutStore,
      'salePoints': salePoints,
    };
  }
}
```

Додаток Е

Лістинг модуля OTP-аутентифікації

```
@override
  void initState() {
    _verifyPhone();
    super.initState();
  }

  void _verifyPhone() async {
    await FirebaseAuth.instance.verifyPhoneNumber(
      phoneNumber: '+380${widget.phone}',
      codeSent: _onCodeSent,
      verificationCompleted: _onVerificationCompleted,
      verificationFailed: _onVerificationFailed,
      codeAutoRetrievalTimeout: _onCodeAutoRetrievalTimeout,
      timeout: const Duration(seconds: 120),
    );
  }

  void _onCodeSent(String? verificationID, int? resendToken) {
    setState(() {
      _verificationCode = verificationID;
    });
  }

  void _onVerificationAndPinCompleted(String pin,
      {PhoneAuthCredential? credential}) async {
    try {
      credential ??= PhoneAuthProvider.credential(
        verificationId: _verificationCode!,
        smsCode: pin,
      );

      UserCredential authResult =
          await
FirebaseAuth.instance.signInWithCredential(credential);

      String uid = authResult.user!.uid;
      _handleVerificationSuccess(widget.name, widget.phone, uid);
    } catch (e) {
      ScaffoldMessenger.of(context).showSnackBar(const SnackBar(
        content: Text('Код введено невірно. Повторіть спробу.'),
      ));
    }
  }

  void _onPinCompleted(String pin) {
    _onVerificationAndPinCompleted(pin);
  }
```

```
  void _onVerificationCompleted(PhoneAuthCredential credential) {
    _onVerificationAndPinCompleted('', credential: credential);
  }

  void _handleVerificationSuccess(String name, String phone,
String uid) async {
    try {
      final CollectionReference usersCollection =
          FirebaseFirestore.instance.collection('users');
      if (widget._authMode == AuthMode.signup) {
        try {
          await usersCollection.doc(uid).set({
            'phoneNumber': '+380$phone',
            'name': name,
          });
        } catch (error) {
          print('Error saving user data: $error');
        }
      }

      Navigator.pushAndRemoveUntil(
        context,
        MaterialPageRoute(builder: (context) => const MyApp()),
        (route) => false,
      );
    } catch (error) {
      print('Error saving user data: $error');
    }
  }

  void _onVerificationFailed(FirebaseAuthException e) {
    print(e.message);
  }

  void _onCodeAutoRetrievalTimeout(String verificationID) {
    setState(() {
      _verificationCode = verificationID;
    });
  }
```