Міністерство освіти і науки України
Криворізький національний університет
Кафедра моделювання та програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти бакалавра

зі спеціальності 121Інженерія програмного забезпечення

На тему: Розробка та реалізація мобільного додатку для автоматизації процесу покупок в мережі магазинів з використанням технології штрих-кодів

Засвідчую, що в цій кваліфікаційній роботі немає запозичень із праць інших авторів без відповідних посилань.

Студент гр. ІПЗ-20-1
_____ / Троцик М.І /

Керівник
кваліфікаційної роботи _____ / А. В. Козиков /

Завідувач кафедри _____ / А. М. Стрюк /

Кривий Ріг
2024

Криворізький національний університет
Факультет: Інформаційних технологій
Кафедра: Моделювання та програмного забезпечення
Ступінь вищої освіти: бакалавр
Спеціальність: 121 - Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Зав. кафедри
_____ А. М. Стрюк
«___» _____ 20__ р.

ЗАВДАННЯ
на кваліфікаційну роботу
студента групи ІПЗ-20-1 Троцика М.І.

Тема: Розробка та реалізація мобільного додатку для автоматизації процесу покупок в мережі магазинів з використанням технології штрих-кодів

Терміни подання студентом закінченої роботи: 15 червня 2024 р.

Вихідні дані по роботі: розроблена система повинна забезпечувати швидке додавання товарів до кошика за допомогою QR-кодів, автоматичний підрахунок загальної суми покупок та зручне управління списком товарів. Мінімальна точність розпізнавання штрих-кодів - 95%.

Зміст пояснювальної записки:

- Актуальність розробки додатку.
- Аналіз ринку мобільних додатків для покупок.
- Визначення цільової аудиторії та формулювання вимог до функціональності.
- Розробка архітектури додатку.
- Створення прототипу інтерфейсу користувача (UX/UI).
- Реалізація основного функціоналу: сканування QR-кодів, ручний ввід штрих-кодів, підрахунок загальної суми покупок.
- Інтеграція з локальною базою даних.
- Проведення модульного, інтеграційного та функціонального тестування.
- Підготовка до захисту кваліфікаційної роботи.

КАЛЕНДАРНИЙ ПЛАН

№	Найменування етапів кваліфікаційної роботи	Термін виконання етапів роботи
1	Збір вимог та аналіз ринку мобільних додатків для покупок	27.02.2024 – 10.03.2024
2	Визначення цільової аудиторії та формулювання вимог до функціональності	11.03.2024 – 20.03.2024
3	Розробка архітектури додатку	21.03.2024 – 30.03.2024
4	Створення прототипу інтерфейсу користувача (UX/UI)	01.04.2024 – 10.04.2024
5	Реалізація функціоналу сканування QR-кодів та ручного вводу штрих-кодів	11.04.2024 – 20.04.2024
6	Розробка модуля для автоматичного підрахунку загальної суми покупок	21.04.2024 – 30.04.2024
7	Інтеграція з локальною базою даних	01.05.2024 – 10.05.2024
8	Проведення модульного та інтеграційного тестування	11.05.2024 – 20.05.2024
9	Тестування користувацького інтерфейсу та функціональне тестування	21.05.2024 – 31.05.2024
10	Підготовка документації та матеріалів для захисту	01.06.2024 – 15.06.2024

Реферат

Вступ

Актуальність розробки "ShopHelper" зумовлена потребою споживачів у спрощенні процесу вибору та купівлі товарів. Мета додатку - надання користувачам контролю над покупками та бюджетом у реальному часі.
Реферат описує процес розробки "ShopHelper" та демонструє, як мобільні додатки можуть трансформувати процес покупок.

Функціональність

- Сканування QR-кодів для швидкого додавання товарів до корзини.
- Ручний ввід штрихкодів.
- Автоматичний підрахунок загальної суми покупок.
- Зручний перегляд та управління списком товарів.

Тестування

Тестування "ShopHelper" включало:

- Модульне тестування алгоритмів обробки QR-кодів та взаємодії з базою даних.
- Інтеграційне тестування взаємодії між модулями.
- Тестування користувацького інтерфейсу для забезпечення інтуїтивності.
- Функціональне тестування для перевірки загальної роботи додатку.

Результати підтвердили високу надійність, продуктивність та зручність додатку.

Переваги

- Сканування QR-кодів прискорює процес вибору товарів.
- Локальна база даних забезпечує швидкий доступ до інформації без підключення до Інтернету.
- Інтуїтивний інтерфейс спрощує навігацію та використання додатку.

Виклики та обмеження

- Забезпечення сумісності з різноманітними пристроями Android.
- Питання безпеки даних користувачів.

Майбутній розвиток

Майбутнє "ShopHelper" передбачає розширення бази даних товарів, інтеграцію з онлайн-магазинами, використання штучного інтелекту для персоналізації пропозицій.

Висновок

1. Ефективність сканування QR-кодів: Технологія спрощує додавання товарів до корзини.

2. Локальне зберігання даних: SQLite забезпечує швидкий доступ до даних без постійного підключення до мережі.

3. Користувацький інтерфейс: Інтуїтивний дизайн спрощує використання додатку.

4. Потенціал для розвитку: Інтеграція з онлайн-магазинами та використання штучного інтелекту можуть підвищити цінність додатку.

Abstract

Introduction

The relevance of developing "ShopHelper" is driven by the need for consumers to simplify the process of selecting and purchasing goods. The app aims to provide users with real-time control over their purchases and budgets.

Functionality

- QR code scanning for quick addition of items to the cart.
- Manual input of barcodes.
- Automatic calculation of the total purchase amount.
- Convenient viewing and management of the list of added items.

Testing

Testing of "ShopHelper" included:

- Module testing of QR code processing algorithms and database interaction.
- Integration testing of interactions between different modules.
- User interface testing to ensure intuitiveness.
- Functional testing to check the overall performance of the application.

Advantages

- QR code scanning speeds up the product selection process.
- A local database provides quick access to information without an Internet connection.
- The intuitive interface simplifies navigation and app usage.

Future Development

The future of "ShopHelper" includes expanding the product database, integrating with online stores, and using artificial intelligence for personalized recommendations.

Conclusion

1. -QR Code Scanning Efficiency: - The technology simplifies adding items to the cart.
2. -Local Data Storage: - SQLite ensures quick access to data without a constant network connection.
3. -User Interface: - Intuitive design makes the app easy to use.
4. -Development Potential: - Integration with online stores and the use of artificial intelligence can enhance the app's value

ЗМІСТ

Вступ

Сучасний світ швидко змінюється, і технології відіграють вирішальну роль у спрощенні повсякденного життя. Особливо це стосується сфери роздрібної торгівлі, де інноваційні рішення можуть значно покращити досвід покупців. Актуальність розробки мобільного додатку для автоматизації процесу покупок у мережі магазинів з використанням технології штрих-кодів не викликає сумнівів. Усе більше споживачів прагнуть заощаджувати свій час та мати можливість контролювати витрати у реальному часі. Тому проект "ShopHelper" має велике значення для сучасного ринку роздрібної торгівлі.

Основна мета розробки додатку "ShopHelper" полягає у створенні інструменту, який спростить процес вибору та купівлі товарів для користувачів. Додаток повинен надати можливість сканування QR-кодів товарів для їх швидкого додавання до віртуальної корзини, автоматичний підрахунок загальної суми покупок, а також зручний перегляд та управління списком товарів. Завданнями проекту є:

1. Розробка інтерфейсу користувача: забезпечення інтуїтивного та зручного у використанні інтерфейсу, який полегшить взаємодію користувачів з додатком.

2. Інтеграція функції сканування QR-кодів: реалізація можливості швидкого та точного сканування товарів за допомогою камери мобільного пристрою.

3. Автоматизація підрахунків: забезпечення автоматичного підрахунку загальної суми покупок у режимі реального часу.

4. Тестування та валідація: проведення всебічного тестування додатку для гарантування його надійності, продуктивності та безпеки.

Для розробки мобільного додатку "ShopHelper" було обрано технологію штрих-кодів, яка є однією з найпоширеніших та найефективніших для автоматизації процесів у роздрібній торгівлі. Штрих-коди дозволяють швидко ідентифікувати товари, що значно скорочує час на їх вибір та оплату. Крім того, використання локальної бази даних забезпечує швидкий доступ до інформації без необхідності підключення до Інтернету, що робить додаток більш зручним та автономним.

- Сканування QR-кодів: швидке додавання товарів до віртуальної корзини за допомогою сканування QR-кодів.

- Ручний ввід штрихкодів: можливість ручного введення штрихкодів у разі, якщо сканування неможливе.

- Автоматичний підрахунок: підрахунок загальної суми покупок у реальному часі.

- Перегляд списку товарів: зручний інтерфейс для перегляду та управління списком товарів у корзині.

- Безпека даних: забезпечення конфіденційності та захисту персональних даних користувачів.

Проект "ShopHelper" має ряд значних переваг для користувачів та магазинів:

- Для користувачів: економія часу, зручність та точність у підрахунках, можливість контролювати витрати, отримувати сповіщення про знижки та акції.

- Для магазинів: підвищення рівня задоволеності клієнтів, зменшення черг на касах, можливість зібрання даних про вподобання та поведінку покупців для подальшого аналізу та поліпшення сервісу.

Цей проект має потенціал стати важливим інструментом у сфері роздрібної торгівлі, відповідаючи сучасним вимогам споживачів та сприяючи подальшому розвитку інноваційних технологій у цьому напряму.

1 СУЧАСНИЙ СТАН ТА АКТУАЛЬНІСТЬ РОБОТИ

1.1 Постановка проблеми

Ера цифрових технологій принесла незаперечні переваги, які радикально змінили та покращили багато аспектів сучасного людського життя та повсякденного досвіду, включно з тим, як ми купуємо товари та послуги. Хоча зручність і ефективність онлайн-покупок неможливо переоцінити, практика відвідування традиційних фізичних роздрібних магазинів все ще є дуже трудомістким процесом, який може призвести до значного стресу, втоми та розчарування для багатьох споживачів. Доводиться витрачати значну й часто важку кількість часу на те, щоб добиратися до магазинів і повертатися з них, шукати в ряду ряди потрібні продукти, ретельно порівнювати ціни на різні варіанти, а потім чекати в довгих чергах на касах, щоб оплатити останні покупки. Ця громіздка і іноді божевільна послідовність необхідних дій створює явну потребу в інноваційних технологічних рішеннях, які дозволять споживачам всебічно оптимізувати та різко спростити кожен етап процесу покупки, тим самим скорочуючи надмірний період часу, який покупці повинні витрачати на фізичну присутність у роздрібних магазинах, і, зрештою, підвищення загальної задоволеності покупців тим, що зазвичай вважається необхідною, але виснажливою роботою сучасного життя.

1.2 Мета роботи

Головною метою та рушійною мотивацією цього конкретного починання є розробка та впровадження інноваційної передової мобільної програми, розробленої спеціально для операційної системи Android. Використовуючи ключові функції, такі як можливість користувачам безперешкодно додавати бажані продукти до віртуального кошика для покупок, ця програма намагається безпосередньо вирішити неприємні проблеми та болючі точки, описані раніше щодо часто важкої та стресової природи традиційної особистої зустрічі. досвід покупок. Однак амбіції та ціннісні пропозиції програми виходять далеко за рамки простого скорочення надмірних часових вимог і вимог до розкладу споживача, коли справа доходить до покупки необхідних товарів.

Програма ретельно розроблена, щоб забезпечити інтуїтивно зрозумілий, зручний і зручний інтерфейс, який спрощує й оптимізує весь процес керування списком покупок. Від легкого створення, редагування та впорядкування детальних списків покупок до виконання комплексного аналізу порівняння цін у кількох роздрібних продавців лише кількома

дотикамипрограма має на меті консолідувати та ущільнити те, що наразі є роз'єднаними та фрагментованими аспектами подорожі до покупки та покупки. Крім того, це надасть користувачам можливість миттєво отримувати доступ до інформації в реальному часі про поточні акції, знижки, розпродажі та спеціальні пропозиції, доступні у фізичних торгових точках у їхньому регіоні. Цей потужний синтез функцій покликаний не тільки заощадити дорогоцінний час і енергію покупців, але й фундаментально перетворити процес покупки на ефективний, приємний і персоналізований.

1.3 Завдання дослідження

1. Провести детальний аналіз ринку мобільних додатків для шопінгу з метою виявлення незадоволених потреб користувачів.
2. Визначити ключові функції, які повинен мати додаток, для забезпечення високого рівня користувацького досвіду.
3. Розробити прототип мобільного додатку, включаючи його архітектуру та дизайн інтерфейсу.
4. Здійснити програмну реалізацію додатку, враховуючи вимоги до функціональності та користувацького інтерфейсу.
5. Протестувати додаток з метою ідентифікації та виправлення помилок, а також оцінки зручності користування.
6. Проаналізувати зібрані дані з тестування, зробити висновки про ефективність додатку та його потенціал на ринку.

1.4 Актуальність роботи

Розробка інноваційної додатки для мобільних пристроїв, яка дозволяє користувачам оптимізувати та оптимізувати кожен аспект процесу покупки, є надзвичайно важливою справою в умовах сучасного сучасного ринку роздрібної торгівлі, що швидко розвивається. У зв'язку зі справді широким розповсюдженням і майже повсюдним використанням складних смартфонів серед демографічних груп існує значний і стрімко зростаючий інтерес споживачів до використання можливостей мобільних додатків для полегшення виконання звичайних завдань і справ, які традиційно вимагали значного часу та вкладення зусиль.

Таким чином, створення продумано розробленої та повнофункціональної програми, яка безпосередньо відповідає цим широко поширеним потребам і бажанням споживачів, щоб зробити покупки максимально ефективними та легкими, має величезний потенціал для широкого впровадження та активного використання значною частиною населення. Концентруючи такі дії, які наразі є відключеними, як-от створення

списків, дослідження продукту, порівняння цін і завершення транзакцій, у єдине цілісне мобільне рішення, ця програма представляє неймовірну цінну пропозицію, яка цілком може кардинально змінити підхід сучасних споживачів до громіздкої, але необхідної роботи. покупки.

Оскільки глобальна база користувачів дедалі більше демонструє тенденції до пошуку зручності за будь-якої нагоди за допомогою технологічних рішень, розробка програми, спеціально присвяченої радикальному спрощенню процесу покупки від початку до кінця, позиціонує продукт для потенційного неперевершеного успіху та захопленого захоплення. Ринковий попит і апетит до таких інновацій, безсумнівно, присутні та розширюються з кожним роком, оскільки наша спільна залежність від мобільних пристроїв та інтеграція з ними все глибше вкорінюється в повсякденному житті та рутині.

1.5 Огляд існуючих рішень

Поглиблений і всебічний аналіз, який оцінює поточний ландшафт існуючих мобільних торгових програм, доступних сьогодні на ринку, показав, що переважна більшість цих рішень, як правило, зосереджується переважно на найпростіших фундаментальних функціях і функціях. Головною серед загальних основних можливостей є можливість для користувачів вручну створювати та керувати списками покупок товарів, які вони мають намір придбати, а також використовувати камери пристрою та технологію сканування штрих-кодів/QR-кодів для швидкого та зручного додавання продуктів до цих віртуальних списків.

Проте аналіз суттєво виявив кілька ключових областей і розширених функцій, які представляють значні прогалини та можливості для інновацій, які існуючі програми не змогли належним чином усунути або інтегрувати безперебійним, надійним способом. Примітно, що хоча створення списків може бути стандартним, надання справді уніфікованого досвіду віртуального кошика для покупок, який зберігається протягом сеансів і дозволяє користувачам без особливих зусиль переносити перераховані товари безпосередньо в чергу для оформлення замовлення, вкрай бракує серед поточних пропозицій.

Крім того, аналіз визначив, що надання користувачам оновлень у режимі реального часу про зміни цін, тимчасові знижки/акції та фактичний рівень запасів у магазинах у роздрібних торговців є безцінною функцією, яка може помітно покращити досвід покупок, але така, що не застосовується всебічно досягнуто до цього часу. Мабуть, найважливішим є те, що розробка та впровадження інтелектуальних алгоритмів, здатних вивчати унікальні вподобання користувачів і купівельні звички з часом, щоб забезпечити повністю персоналізований, підібраний досвід, адаптований до кожного

окремого споживача, є практично недослідженою територією, яка дозріла для творчого підриву.

Ця кульмінація висновків чітко показує, що, незважаючи на те, що поточні пропозиції програмного забезпечення на ринку можуть задовольнити фундаментальні основи, залишаються величезні можливості та потенціал для нових інноваційних рішень, які зосереджуються на підвищенні функціональності, продуктивності та загальної якості мобільного шопінгу до рівня нові висоти, раніше не досягнуті. Ефективне усунення цих прогалин дає шанс встановити новий стандарт досконалості мобільних додатків для покупок.

1.6 Визначення цільової аудиторії

Передбачається, що цільова демографічна аудиторія цього інноваційного мобільного додатку для покупок охопить досить широке коло потенційних користувачів з різних верств суспільства та різного походження. Проте певні сегменти позиціонуються так, щоб отримати особливо значну цінність від основних можливостей і ціннісних пропозицій програми.

Один із ключових сегментів складається з зайнятих працюючих професіоналів, які надають перевагу максимізації ефективності та гарантують, що жодна хвилина їх сильно обмеженого вільного часу не буде витрачена даремно. Для цих користувачів можливість безперешкодно планувати, організовувати та виконувати свої обов’язки по покупкам виключно зі свого мобільного пристрою під час очікування таксі, під час поїздки на роботу чи під час обідньої перерви представляє кардинальний шанс повернути дорогоцінні години, витрачені інакше фізично. присутні в переповнених магазинах.

Інший важливий сегмент існує у вигляді сімей і домогосподарств, які постійно шукають ефективні способи ретельної оптимізації своїх процесів купівлі та ретельного планування стійкого бюджету для необхідних товарів. Використовуючи такі функції, як автоматичне порівняння цін, динамічне коригування списків покупок у режимі реального часу та інтелектуальне відстеження бюджету/витрат, ці користувачі можуть стратегічно максимально використовувати спільні фінанси своєї родини.

Важливо, що особлива увага та наголос також приділяються активному зверненню до унікальних потреб людей, які вже щиро прийняли інтеграцію мобільних додатків для полегшення та оптимізації своїх повсякденних завдань, домашніх справ і способу життя. Оскільки ця «рідна» демографічна група продовжує експоненціально зростати, надання ефектного рішення для покупок, яке органічно вписується в їхню існуючу поведінку та цифрові екосистеми, матиме першочергове значення.

Зрештою, завдяки поєднанню основних переваг, які одночасно задовольняють потреби в зручності для тих, хто відчуває брак часу, можливості планування бюджету для фінансово обізнаних та інтуїтивно зрозумілий користувальницький досвід для технічно підкованих і орієнтованих на програми, ця програма позиціонує себе як надзвичайну цінність. надзвичайно широкій, але цільовій групі сучасних споживачів.

1.7 Мета та завдання дослідження у контексті цільової аудиторії

Основна мета цього дослідження полягає в тому, щоб розробити надійну мобільну програму, яка максимально повно відповідає різноманітним потребам і вподобанням визначених сегментів цільової аудиторії та задовольняє їх. Для досягнення цієї головної мети потрібно ретельно виконати кілька критичних завдань і вимог:

Розробка інтуїтивно зрозумілого, зручного інтерфейсу з дотриманням передових практик UX/UI має першочергове значення. Інтерфейс програми має забезпечувати легкий і легкий досвід завдяки логічним потокам навігації, чутливій взаємодії та чистій, візуально привабливій естетиці дизайну. Широке тестування зручності використання вдосконалюватиме інтерфейс користувача, поки він не досягне чудової взаємодії з користувачем.

Необхідно тісно інтегрувати серверні системи додатка з базами даних і цифровою інфраструктурою основних роздрібних партнерів через безпечні API. Це відкриває доступ до даних про продукти в реальному часі, інформації про ціни, наявності асортименту та діючих акцій/знижок. Підтримуючи цю безперебійну інтеграцію, користувачі мають під рукою найновішу роздрібну інформацію.

Розробка програми з нуля з масштабованістю та високою продуктивністю як критично важливими принципами дизайну гарантує оперативне реагування, здатне впоратися зі збільшенням використання в масштабі. Потрібні оптимізований код на стороні клієнта, ефективна мережа та надійна інфраструктура на стороні сервера, розрахована на стрибки попиту.

Впровадження багаторівневих засобів контролю безпеки та механізмів захисту даних, таких як шифрування, автентифікація та дотримання правил конфіденційності даних, як-от GDPR, захищає конфіденційну інформацію користувачів і платіжні дані від злому та використання. Рекомендується співпрацювати з надійними постачальниками засобів безпеки.

У сукупності гармонійна конвергенція видатного UX, точних даних про роздрібну торгівлю в режимі реального часу, високоякісної технічної продуктивності та провідної в галузі безпеки стануть кульмінацією в **потужному інструменті для покупок, ретельно створеному, щоб задовольнити** цільових користувачів у будь-якій якості..

1.8 Технологічні аспекти розробки

Розпочинаючи розробку надійної та продуктивної мобільної програми, надзвичайно важливо ретельно розглянути та вибрати оптимальний набір технологій, які сформують базову основу та архітектуру. Вибір правильного технологічного стеку, який охоплює мови програмування, бібліотеки, фреймворки, інструменти та допоміжне програмне забезпечення, є ключовим визначальним фактором, який безпосередньо впливатиме на загальну ефективність програми, швидкість роботи, масштабованість і такі аспекти взаємодії з кінцевим користувачем, як інтуїтивна простота- використання.

Для додатків, націлених саме на мобільну операційну систему Android, досвідчені розробники вибирають дві мови програмуванняKotlin і Java. Kotlin, сучасна крос-платформна мова, пропонує численні переваги, такі як підвищена стислість коду, захист від нульових значень, функції розширення та бездоганна взаємодія з Java. Крім того, багато розробників обирають Java, враховуючи її давню повсюдність, можливості продуктивності та міцну інтеграцію з Android.

Незалежно від вибраної мови фактично стандартним інтегрованим середовищем розробки (IDE) і фреймворком, які використовуються, є Android Studio, потужний інструментарій, наданий Google. Android Studio надає повний набір функцій, утиліт і робочих процесів, спеціально розроблених для прискорення розробки високоякісних програм для Android. Це включає в себе розширений редактор коду, інструменти налагодження/тестування, гнучку систему збірки та великі бібліотеки/інтерфейси API для використання всіх можливостей платформи Android.

Окрім основних технологій, допоміжні інструменти та служби відіграють допоміжні ролі. Системи контролю версій, такі як Git, платформи для співпраці, такі як GitHub, хмарні серверні/інфраструктурні рішення, служби push-повідомлень і навіть API машинного навчання можуть бути використані та інтегровані. Конвеєри безперервної інтеграції/розгортання, увімкнені через такі платформи, як Azure DevOps, забезпечують ефективні процеси збірки, тестування та випуску.

Стратегічно зібравши правильну колекцію найсучасніших, але перевірених часом технологій, мов програмування, фреймворків і інструментів розробки, розробники додатків можуть досягти успіху в створенні високофункціонального, продуктивного та зручного мобільного додатка, який безперебійно працює на Операційна система Android для багатьох пристроїв. Ця міцна технологічна основа має ключове значення для реалізації амбітних цілей і інноваційних наборів функцій, передбачених для програми.

1.9 Безпека та конфіденційність даних

Захист безпеки та конфіденційності даних користувача та деталей транзакцій має розглядатися як найвищий пріоритет протягом усього життєвого циклу розробки мобільного додатку для покупок. Застосування найсучасніших криптографічних методів для шифрування всіх даних, що передаються до/з програми, а також даних у спокої буде важливим для запобігання несанкціонованому доступу або перехопленню конфіденційної інформації зловмисними третіми сторонами.

Надійні механізми автентифікації користувачів, як-от багатофакторна автентифікація, біометрія (розпізнавання відбитків пальців/обличчя) або ключі безпеки, повинні бути включені, щоб гарантувати доступ до своїх облікових записів і даних лише законним користувачам. Крім того, потоки автентифікації програми повинні використовувати безпечні протоколи, такі як OAuth 2.0, для інтеграції з постачальниками ідентифікаційної інформації.

Необхідно суворо дотримуватися найкращих практик безпеки, таких як перевірка вхідних даних, кодування вихідних даних і захист від поширених веб-уразливостей, таких як впровадження SQL, XSS, CSRF. Проведення ретельного тестування на проникнення та перевірок безпеки дозволить виявити й усунути потенційні вразливості перед випуском.

З точки зору архітектури, додаток має бути розроблено з урахуванням принципів безпеки, таких як мінімальні привілеї, глибокий захист і безпечні принципи за умовчанням. Впровадження таких механізмів, як безпечні постачальники сховищ ключів, закріплення сертифікатів, рандомізація макета адресного простору ще більше посилить безпеку.

Важливо, що програма також має гарантувати конфіденційність конфіденційної інформації про покупки та особистих даних користувача, дотримуючись правил конфіденційності даних, таких як GDPR і CCPA. Це включає надання користувачам видимості того, які дані збираються, надання механізмів видалення даних і збір/збереження лише мінімальних даних, які абсолютно необхідні.

Співпраця зі спеціалізованою фірмою з безпеки мобільних додатків може надати додаткові рекомендації щодо моделювання загроз, тестування на проникнення, впровадження контролю безпеки та аудиту для сертифікації за глобальними стандартами безпеки.

Зрештою, надання пріоритету безпеці та захисту даних із самого початку через комплексний життєвий цикл безпечної розробки програмного забезпечення буде критично важливим для того, щоб програма надавала безпечний, надійний і конфіденційний досвід для всіх користувачів.

1.10 Інтеграція з іншими системами

Деякі ключові аспекти, пов’язані з інтеграцією API:

Сервер програми має надавати набір безпечних, добре задокументованих API, які відповідають принципам архітектури REST. Ці API діятимуть як міст, дозволяючи мобільному клієнту запитувати та отримувати дані з роздрібних баз даних.

Необхідно докласти великих зусиль для розробки узгодженого, інтуїтивно зрозумілого контракту API, що складається з чітких кінцевих точок ресурсів, структур запитів/відповідей, потоків автентифікації та обробки помилок.

API слід створювати з упором на продуктивність, кешування та стійкість під навантаженням. Використання хмарної інфраструктури, як-от Azure API Management, може забезпечити такі функції, як дроселювання та обмеження швидкості.

Дані, отримані від роздрібних API, мають бути ретельно зіставлені, за потреби перетворені та відформатовані в моделі, оптимізовані для використання мобільним додатком, щоб забезпечити безперебійну роботу користувача.

Додатку може знадобитися підтримувати синхронізований офлайн-кеш або перегляд бази даних про продукт/ціни, щоб забезпечити базову функціональність навіть без підключення до мережі.

Інтерфейси програмного інтерфейсу (API) слід ретельно перевіряти, контролювати та мати відповідні засоби контролю доступу/посилення безпеки, реалізовані для захисту від загроз.

Методи інтеграції, як-от зворотні виклики веб-хука або події, надіслані сервером, можуть увімкнути оновлення в реальному часі, коли дані продавців змінюються.

За потреби програма може надавати API у зворотному напрямку, щоб передавати транзакційні дані, як-от покупки, списки покупок тощо, у системи роздрібних партнерів.

Віддаючи пріоритет надійному, стійкому рівню інтеграції API, дотримуючись найкращих практик, програма гарантує, що користувачі завжди матимуть доступ до найновіших точних даних про продукт, ціни та рекламні акції, забезпечуючи ключову пропозицію щодо оптимізації цифрових покупок.

1.11 Користувацький досвід (UX) та інтерфейс (UI)

Розробка інтуїтивно зрозумілого та привабливого інтерфейсу є вирішальним фактором успіху мобільного додатку. Інвестування часу та ресурсів у належний дизайн UX/UI дозволяє створити програму, яка не тільки

відповідає всім функціональним вимогам, але й забезпечує користувачам приємні враження від її використання. Особливу увагу слід приділити логіці потоків навігації, щоб забезпечити інтуїтивно зрозумілі шляхи використання функцій програми. Швидкість реакції на дії та взаємодію користувача також є життєво важливою - будь-яка повільність або затримки розчарують користувачів. Нарешті, загальна естетика та візуальний дизайн мають знайти баланс між привабливістю та лаконічністю. Елементи інтерфейсу користувача, типографіка, кольори, іконографіка та макетусе це сприяє приємному чи неприємному користуванню. Застосування найкращих практик людсько-орієнтованого дизайнерського мислення та проведення ітераційних сеансів тестування зручності використання з цільовими користувачами будуть ключовими для вдосконалення інтерфейсу користувача, поки він не досягне легкого та чудового досвіду.

1.12 Висновок до розділу

Розробка мобільного додатку для оптимізації процесу покупок у супермаркетах є актуальним та перспективним проектом, який відповідає потребам сучасного споживача. Виконання поставлених завдань, включаючи глибокий аналіз ринку, розробку зручного інтерфейсу, забезпечення безпеки даних та інтеграцію з системами магазину, потребує комплексного підходу та використання сучасних технологій.

Широкий аналіз ринку закладає важливу основу, визначаючи ключові прогалини в існуючих рішеннях і підтверджуючи попит на додаток, який спрощує потоки покупок продуктів. Розробка інтуїтивно зрозумілого та візуально привабливого інтерфейсу користувача за допомогою найкращих практик UX/UI матиме першочергове значення для надання чудового досвіду, який зацікавить користувачів.

Надійні заходи безпеки, такі як шифрування, автентифікація та дотримання правил конфіденційності даних, не підлягають обговоренню для захисту конфіденційної інформації користувача та платіжних даних. Повна інтеграція з базами даних роздрібних продавців через API дозволить додатку надавати дані про продукт/ціни/акції в реальному часі для максимальної зручності.

Реалізація повного обсягу цих вимог вимагає використання передових технологій і архітектурних шаблонів. Для Android сучасні мови, такі як Kotlin, разом із фреймворками, такими як Android Studio, можуть забезпечити швидку, але безпечну розробку. Допоміжні інструменти для CI/CD, контролю версій, співпраці та хмарних служб сприяють ефективним робочим процесам. Успішна реалізація цього проекту має потенціал не тільки радикально покращити досвід покупки продуктів для споживачів шляхом економії часу та

зусиль, але й фундаментально порушити та підвищити стандарти в усій галузі роздрібної торгівлі. У міру розвитку мобільних звичок споживачів роздрібні продавці, які використовують інноваційні додатки, отримають довгостроковий успіх.

2 АНАЛІЗ РИНКУ ВИЗНАЧЕННЯ ВИМОГ ДО ДОДАТКУ

2.1 Аналіз ринку мобільних додатків для шопінгу

Аналізуючи сучасний ринок мобільних додатків для покупок, можна виділити кілька основних трендів, що формують вимоги споживачів:

- Персоналізація: персоналізація є основною тенденцією та пріоритетом для сучасних користувачів мобільних додатків. Вони очікують інтелектуальних механізмів персоналізації, що керуються даними, які можуть аналізувати їх унікальну поведінку, уподобання та контекстні дані, щоб динамічно налаштовувати та курувати весь досвід спеціально для них як особистості.

Деякі ключові можливості персоналізації, які будуть важливі для програми для покупок:

- Персоналізовані рекомендації щодо продуктів на основі минулої історії покупок у всіх категоріях продуктів. Складні алгоритми рекомендацій можуть ідентифікувати схеми спорідненості між продуктами.

- Персоналізовані акції, знижки та пропозиції з’являлися на помітному місці на основі активності веб-переглядача, улюблених брендів/продуктів, місця розташування та інших сигналів.

- Розумні пропозиції списку покупок, які можуть автоматично заповнюватися на основі прогнозованих потреб на основі моделей споживання та циклів поповнення.

- Динамічний мерчандайзинг і вміст, який регулює те, що відображається, залежно від інтересів користувачів, демографічних показників і контексту, наприклад часу доби.

- Індивідуалізовані повідомлення в додатку, сповіщення та сповіщення, орієнтовані на потреби кожного користувача.

Чим більше складних особистих даних і неявних/явних уподобань програма може отримувати з часом, тим багатшими та ціннішими стають ці можливості персоналізації. Застосування машинного навчання може навіть допомогти передбачити майбутні інтереси.

Забезпечення такого рівня продуманої персоналізації, коли програма проактивно націлюється на кожного користувача на основі його індивідуальних потреб, буде критично важливим для побудови стійкої довгострокової взаємодії. Користувачі дедалі більше покладають високі очікування щодо персоналізації, якій програми повинні відповідати або перевищувати.

У сучасній надконкурентній сфері мобільних додатків надання високо персоналізованих і контекстуалізованих користувальницьких можливостей, адаптованих до кожної людини, є основним завданням. Користувачі звикли до того, що системи передбачають їхні потреби та завчасно надають

відповідний, підібраний досвід. Програми, у яких не вдається реалізувати складні можливості персоналізації, ризикують відтоком і залишенням.

- Інтеграція з магазинами: споживачі мають високий попит на програми для покупок, тісно інтегровані з інфраструктурою та базами даних певних роздрібних мереж або бакалійних магазинів, які вони часто відвідують. Це дає змогу отримати доступ до наявності запасів у найближчих місцях у реальному часі, можливість переглядати поточні ціни/акції та користуватися ексклюзивними винагородами та пропозиціями програми лояльності. По суті, користувачі хочуть безперебійного багатоканального досвіду.

Деякі з ключових можливостей інтеграції, очікуваних користувачами, включають:

- Дані про запаси/запаси в реальному часі, щоб перед відвідуванням побачити наявність продуктів у найближчих магазинах
- Доступ до поточних цін, знижок, акцій і деталей пропозицій в Інтернеті та в магазині
- Можливість накопичувати та обмінювати винагороди/бали програми лояльності на різних каналах
- Перегляд історії покупок і списків покупок онлайн або в магазині
- Постійні кошики для покупок, які можуть переходити між додатком і магазином

Ця тісна синхронізація вимагає створення надійних інтеграційних рівнів і підключень API до систем роздрібної торгівлі. Потоки даних можуть бути двонаправленими, споживаючи дані, а також надсилаючи дані, як-от покупки, назад роздрібним продавцям.

Провідні роздрібні продавці все більше відкривають доступ через API та уніфіковані торговельні платформи з такими можливостями, як системи керування замовленнями та керування інформацією про продукти, що перетікають у централізований досвід.

2.2 Надійні рівні інтеграції

- Створення стійких, масштабованих рівнів інтеграції для взаємодії з API і службами роздрібних продавців
- Обробка автентифікації API, відображення даних, кешування відповідей, обробки помилок, повторних спроб тощо.
- Потенційне використання шлюзів API, сервісних мереж для покращеного контролю трафіку

2.2.1 Двонаправлена синхронізація даних

- Не просто споживання даних про продукт/інвентар від роздрібних торговців, а й надсилання даних про покупки користувачам назад
- Це замикає цикл, дозволяючи отримати уніфікований багатоканальний погляд на поведінку клієнтів
- Системи керування замовленнями можуть безперешкодно обробляти мобільні транзакції

2.2.2 Уніфіковані комерційні платформи

- Великі роздрібні торговці надають уніфіковані API, що охоплюють точки взаємодії в магазині, в Інтернеті та на мобільних пристроях
- Інформація про продукт, профілі клієнтів, ціни, запаси, керування замовленнями доступні через платформу
- Дозволяє мобільним додаткам підключатися до централізованої мережі комерційних даних у реальному часі

2.2.3 Встановлені стандарти API

- Стимулювання роздрібних торговців дотримуватися галузевих стандартів щодо контрактів API, безпеки та управління
- Спрощує інтеграцію для сторонніх розробників, які створюють мобільний/багатоканальний досвід
- Такі стандарти, як OpenAPI, GraphQL, OAuth, роблять впровадження більш масштабованим

Завдяки інвестиціям у надійні розширювані архітектури інтеграції мобільні додатки можуть забезпечувати безперебійний багатоканальний досвід, синхронізований із даними роздрібних продавців у реальному часі. Оскільки все більше постачальників розвивають свої серверні платформи, щоб вони були дружніми до API/інтеграції, це відкриває величезний потенціал для інноваційних мобільних подорожей, що входять у їх власні екосистеми.

Для користувачів ця інтеграція окупається завдяки підвищеній зручностівони можуть ефективніше планувати поїздки в магазини, водночас користуючись перевагами цифрового досвіду. Це запобігає фрагментації даних і історії через різні точки дотику.

Інтеграція дає такі функції, як перевірка запасів продуктів у найближчих місцях перед відвідуванням, щоб переконатися в наявності. Користувачі можуть переглядати в Інтернеті, створювати списки покупок, а потім безперешкодно передавати ці списки в магазин для отримання або навігації в проході. Історії покупок залишаються єдиними незалежно від того, чи здійснювалися вони з мобільного пристрою, в Інтернеті чи особисто.

З точки зору роздрібного продавця, двонаправлена інтеграція означає нарешті отримання багатоканального погляду на своїх клієнтів. Дані про перегляд мобільних додатків, покупки, налаштування можна синхронізувати з центральними профілями та системами керування замовленнями. Ці уніфіковані дані відкривають можливості для персоналізації, механізмів рекомендацій і оптимізації стратегій виконання.

У міру того, як очікування споживачів розвиваються в бік вимогливого досвіду покупок, незалежно від каналу, роздрібні продавці, які пропонують API та уніфіковані торгові платформи, матимуть кращі позиції порівняно з конкурентами. Мобільні додатки, інтегровані в їхню екосистему, забезпечують додаткові точки взаємодії для покращення подорожей користувачів. Розширюваність через API стає конкурентною перевагою.

Зрештою, цей перетин надійної інтеграції мобільних додатків у поєднанні з платформами роздрібної торгівлі, керованими API, і уніфікацією даних прокладає шлях для інноваційних, безпроблемних покупок, які втілюють філософію багатоканальності. Це симбіотичні відносини, де дотримання надійних інтеграційних архітектур забезпечує взаємну вигоду для всіх зацікавлених сторін.

Оскільки роздрібні партнери визнають силу взаємодії з додатками, ті, хто відкриває можливості інтеграції, матимуть кращі позиції для підвищення лояльності порівняно з конкурентами. Створення цих з’єднань у реальному часі швидко стає настільним ставками.

- Швидкість і зручність: у нашу еру миттєвого задоволення користувачі додатків віддають перевагу рішенням, які можуть максимізувати швидкість і

зручність на кожному кроці шляху до покупки. Це означає можливість швидкого сканування штрих-кодів/QR-кодів, ефективні методи введення даних, як-от голосовий ввід, потоки покупок одним дотиком і, що важливо, надзвичайно інтуїтивно зрозумілий інтерфейс користувача, розроблений для легкої навігації. Будь-які больові точки або точки непотрібного тертя будуть швидко залишені.

- Швидке сканування штрих-кодів/QR за допомогою камери пристрою, щоб швидко додавати елементи до списків/кошиків одним дотиком або голосовою командою. Комп'ютерний зір також може ідентифікувати продукти за зображеннями.

- Спрощене введення даних виходить за межі простого введення вручну. Голосове введення, сканування квитанцій та інтеграція з потоками даних, як-от цифрові списки продуктів, можуть легко заповнювати інформацію.

- Можливості здійснення покупок одним клацанням миші, як-от інтеграція Apple Pay і Google Pay, щоб уникнути тривалих процесів оформлення повторних покупок.

- Інтелектуальні передбачувані можливості, які можуть попередньо заповнювати часті замовлення, автоматизувати створення списків і проактивно показувати часто куповані товари.

- Надзвичайно вдосконалений, інтуїтивно зрозумілий інтерфейс користувача та навігаційна модель, яка керує користувачами основними завданнями лише кількома дотиками. Чіткі заклики до дії та контекстна допомога.

Будь-яка повільність завантаження, надскладні процеси оформлення або непотрібне введення користувача може вбити коефіцієнт конверсії. Користувачі мають дуже низьку толерантність до марнотраченого часу.
Проведення обширних досліджень зручності використання, шляхів користувача та циклів ітерації інтерфейсу користувача стане ключовим фактором для отримання максимально легкого та чудового досвіду. У мобільну еру імпульсивних звичок зручність завжди перемагає.

Персоналізація. Впровадження інтелектуальних алгоритмів і моделей машинного навчання для динамічної персоналізації роботи програми на основі унікальних уподобань кожного користувача, історії покупок, поведінки веб-переглядача та контекстних даних, як-от місцезнаходження. Персоналізовані рекомендації щодо продуктів, рекламні акції та автономне створення списків покупок - це ставки на стіл.

Глибока інтеграція з роздрібною торгівлею. Налагодження надійного підключення API та тісної синхронізації даних із основними роздрібними партнерами, щоб забезпечити доступ у реальному часі до рівня запасів, цін,

рекламних акцій і можливості безперешкодно накопичувати/використовувати винагороди за лояльність у цифрових каналах і каналах у магазинах.

Одержима зосередженість на швидкості/зручностіоптимізація кожного аспекту взаємодії з користувачем для досягнення максимальної ефективності завдяки безпроблемній навігації, оформленню замовлення одним дотиком, можливостям голосового введення та використанню новітнього комп'ютерного бачення для надшвидкого сканування штрих-кодів/зображень. Будь-яке непотрібне тертя завадить усиновленню.

Розробники повинні вийти за рамки простого встановлення прапорців у списках функційвони повинні гармонізувати ці потужні можливості в цілісну платформу, створену для споживачів, орієнтованих на мобільні пристрої. Надання пріоритету дизайну, орієнтованому на людину, елегантній реалізації UI/UX, надійній масштабованій архітектурі та елементам керування безпекою/конфіденційністю корпоративного рівня буде життєво важливим.
Ті, хто вміло поєднують глибоку персоналізацію, багатоканальні зручності завдяки інтеграції з роздрібною торгівлею та безпроблемний користувальницький досвід, зможуть завоювати аудиторію мобільних споживачів, які все частіше очікують, що процес покупок буде розумнішим, швидшим і більш адаптованим до їхніх індивідуальних потреб.

2.3 Визначення вимог до додатку

На основі поглибленого аналізу ринку та визначених потреб споживачів було сформульовано кілька додаткових ключових вимог до мобільного додатку для покупок:

- Сервіси геолокації: використання можливостей геолокації пристрою є потужним способом надання користувачам гіперрелевантної контекстної інформації на основі їх фізичного розташування в будь-який момент. Деякі ключові випадки використання геолокації в програмі для покупок:

- Виявлення найближчих роздрібних магазинів і відображення такої інформації, як адреса, години роботи, рівень запасів продуктів і будь-які поточні акції/розпродажі, що відбуваються в цих конкретних місцях.

- Можливості геозонування для запуску сповіщень/сповіщень на основі місцезнаходження, коли користувач перебуває в певній близькості від магазину, як-от пропозиція самовивозу з магазину.

- Поєднання геолокації з картографічними/навігаційними послугами для надання вказівок і маршрутів до бажаних місць роздрібної торгівлі або місць розпродажу.

- Використання маршрутів розташування для аналізу даних про переміщення користувачів і моделей, щоб передбачити, коли їм може знадобитися поповнити певні предмети домашнього вжитку, або запропонувати найближчі магазини на основі їхніх частих маршрутів.

- Персоналізація роботи в додатку та рекомендованого вмісту на основі поточного регіону/району користувача, щоб виділити локалізовані тенденції, події та смаки.

Чим точніше програма може визначати дані про місцезнаходження користувача, тим релевантніша інформація та враження, які вона може отримати у відповідь. Ця обізнаність про місцезнаходження надає користувачам величезну цінність з точки зору економії часу, отримання інформації про можливості заощаджень поблизу та отримання унікально персоналізованого додатка, адаптованого для їхнього регіону.

З дозволу користувача дані про геолокацію можуть перетворити програму на незамінного інтелектуального компаньйона, який безперешкодно направляє користувача через найкращі варіанти покупок, що відповідають його реальному контексту.

- Інтерактивні елементи: включення соціальних функцій і функцій, керованих спільнотою, може допомогти створити базу зацікавлених користувачів навколо програми для покупок. Деякі ключові інтерактивні елементи, які слід враховувати:

- Оцінки/відгуки користувачів: дозвольте користувачам залишати оцінки зірками, писати відгуки та залишати відгуки про продукти, які вони придбали. Цей краудсорсинговий соціальний доказ може значно вплинути на рішення про покупку.

- Обговорення продуктів: створюйте спеціальні форуми або потоки коментарів для продуктів/брендів, де користувачі можуть ставити запитання, ділитися порадами та обговорювати свій досвід. Це сприяє цінним розмовам з однолітками.

- Можливості обміну: увімкніть легкий обмін деталями продукту, списками покупок, покупками на соціальних платформах або у власній соціальній мережі програми, щоб отримати рекомендації.

- Профілі користувачів: заохочуйте користувачів створювати різноманітні профілі, що відображатимуть історію їх покупок, рейтинги, корисні публікації та розвивайте репутацію довірених порадників у категоріях продуктів.

- Слідкуйте за моделями: дозвольте людям слідкувати один за одним, а також стежити за певними категоріями продуктів, брендами чи впливовими людьми на основі спільних інтересів.

- Мультимедійний вміст: заохочуйте користувачів надсилати фотографії/відео у своїх оглядах і обговореннях. Цей створений користувачами вміст додає багатства.

- Покупки в прямому ефірі: дивіться відеотрансляції в реальному часі, де впливові особи або представники брендів можуть демонструвати продукти та спілкуватися з глядачами в режимі реального часу.

Чим більше інтерактивних даних зберігається історично, тим ціннішою з часом стає спільнота як база знань. Гейміфікація, винагороди та програми визнання впливових осіб можуть ще більше підвищити рівень участі.
Можливості соціальних обчислень можуть перетворити програму на платформу для відкриття та надати «людський» елемент друзів/особистостей, який багато покупців цінують так само, як відгуки та рекомендації.

- Автоматизовані списки покупок: впровадження інтелектуальної автоматизації для аналізу індивідуальних моделей споживання та історії покупок для створення персоналізованих прогнозованих списків покупок. Це позбавляє від клопоту, пов’язаного з підготовкою списків вручну, і гарантує, що користувачі ніколи не пропустять поповнення запасів побутових скріпок.

- Geofencing/Beaconing: використання геозонування та технології маяків Bluetooth для ініціювання сповіщень на основі місцезнаходження, нагадувань і мікролокаційної навігації, коли користувачі входять у приміщення роздрібного партнера. Ця функціональність навколишнього середовища забезпечує безпроблемний досвід здійснення покупок.

- Мультимодальне сканування:вихід за межі простого сканування штрих-кодів шляхом реалізації мультимодального сканування на основі комп’ютерного зору AI відкриває нові зручності для користувачів. Деякі ключові можливості:

- Розпізнавання зображень. Дозвольте користувачам просто сфотографувати продукт, і програма ідентифікуватиме його за допомогою моделей розпізнавання зображень машинного навчання, зіставлених із базою даних продуктів. Не потрібно шукати/сканувати штрих-код.

- Виявлення об'єктів - виходьте за межі ідентифікації одного продукту, дозволяючи користувачам робити знімок цілої полиці або комори. Потім програма може виявити та автоматично заповнити список усіх продуктів, розпізнаних за допомогою штучного інтелекту виявлення об’єктів.

- Оптичне розпізнавання символів. Для продуктів без штрих-кодів програма могла зчитувати назву продукту, розмір і важливі деталі безпосередньо з тексту на упаковці за допомогою технології OCR.

- Виявлення аудіо - голосові можливості можуть дозволити користувачам вимовляти назву продукту, яка обробляється за допомогою розпізнавання мовлення, щоб додати до свого кошика без використання рук.
Впроваджуючи мультимодальний штучний інтелект, який поєднує зір, мову та інші модальності, окрім штрих-кодів, він зменшує величезне тертя та когнітивне навантаження для користувачів. Вони можуть просто використовувати свій голос, камеру або датчики навколишнього середовища, щоб легко додавати елементи.
Оскільки ці моделі штучного інтелекту вдосконалюються за допомогою машинного навчання та даних про використання, досвід з часом ставатиме

точнішим. Перетворення мультимодального сканування на ключову частину базової архітектури додатка позиціонує його на передньому краї безпроблемного введення даних.

2.4 Технічні аспекти реалізації

Впровадження RESTful API як комунікаційного рівня забезпечить плавну інтеграцію та обмін даними між мобільним клієнтом і серверними системами. Ця керована API архітектура забезпечує чіткий розподіл обов'язків і модульну, розширювану кодову базу.

Серверні компоненти слід розробляти з урахуванням горизонтального масштабування, щоб легко справлятися зі збільшеним трафіком і навантаженням шляхом додавання додаткових ресурсів. Хмарні проекти з використанням контейнеризації та оркестровки з Kubernetes можуть полегшити це.

Архітектура на основі мікросервісів, де різні бізнес-домени, як-от керування кошиками, рекламні акції, інвентаризація тощо, ізольовано в незалежні служби, що розгортаються, запобігає монолітним кодовим базам і забезпечує гнучкість.

Рівень даних має використовувати гнучкі сховища даних, такі як бази даних NoSQL, які можна легко масштабувати та підтримувати різноманітні моделі даних у міру появи нових ліній продуктів/функцій.

Інтеграція з системами роздрібних партнерів для даних про продукт/ціноутворення/рекламу також має відбуватися через надійні безпечні API, які відповідають галузевим стандартам. Шлюзи та адаптери можуть ізолювати код основного додатка від попередніх змін.

Архітектура, керована подіями, з асинхронними шаблонами зв'язку може додатково роз'єднати компоненти та запобігти тісному зв'язку між системами.

З точки зору клієнта, основна логіка програми повинна бути належним чином розподілена на окремі модулі, що обробляють інтерфейс користувача, мережу, доступ до даних тощо. Це забезпечує ефективну паралельну розробку та масштабування команд.

Такі принципи, як безперервна інтеграція/розгортання та культура DevOps, забезпечать швидкий і надійний розвиток додатка в темпі інновацій.

Розробка справді перспективної, масштабованої архітектури має вирішальне значення для довгострокового успіху, оскільки програма з часом стає все складнішою.

Рівень даних повинен використовувати гнучкі сховища даних, такі як бази даних NoSQL, які можуть легко масштабуватися та підтримувати різноманітні моделі даних у міру появи нових лінійок продуктів/функцій.

Інтеграція із системами роздрібних партнерів для отримання даних про продукти/ціноутворення/промоакції також має здійснюватися через надійні захищені API, які відповідають галузевим стандартам. Шлюзи та адаптери можуть ізолювати основний код додатка від змін у сторонніх системах.
Архітектура, керована подіями, з асинхронними шаблонами комунікації може додатково роз'єднати компоненти та запобігти тісному зв'язку між системами.
Для клієнтської частини логіка основного додатка має бути належним чином розділена на окремі модулі, що охоплюють інтерфейс користувача, мережеві взаємодії, доступ до даних тощо. Це дозволяє ефективно розпаралелювати розробку та масштабувати команди.

Дотримання принципів безперервної інтеграції/розгортання та культури DevOps забезпечить швидку та надійну еволюцію додатка в ногу з інноваціями.
Розробка дійсно майбутньо-стійкої, масштабованої архітектури від самого початку є критично важливою для довгострокового успіху, оскільки додаток стає більш досконалим з часом.

З точки зору клієнта, основна логіка програми повинна бути належним чином розподілена на окремі модулі, що обробляють інтерфейс користувача, мережу, доступ до даних тощо. Це забезпечує ефективну паралельну розробку та масштабування команд.

Такі принципи, як безперервна інтеграція/розгортання та DevOps kultur, забезпечать можливість швидкого та надійного розвитку додатка в темпі інновацій.
Розробка справді перспективної масштабованої архітектури має вирішальне значення для довгострокового успіху, оскільки програма з часом стає все складнішою.

2.5 Прототипування та дизайн

Розробка прототипу додатку є ключовим етапом, який дозволяє визначити користувацький інтерфейс, основні екрани та навігацію. Використання інструментів дизайну, таких як Figma або Sketch, сприяє ефективній комунікації між дизайнерами, розробниками та зацікавленими сторонами.

Створення прототипу програми на ранніх стадіях розробки є критично важливим для визначення та узгодження основних аспектів користувацького інтерфейсу та взаємодії з додатком. Цей крок дозволяє:

1. Візуалізувати та обговорити макети екранів, розташування елементів UI, потоки навігації тощо до написання коду.

2. Ефективно донести ідеї та концепції дизайну до всіх зацікавлених сторін - розробників, керівників проєктів, замовників.

3. Виявити та вирішити потенційні проблеми з юзабіліті на ранньому етапі, уникаючи дорогих змін пізніше.

4. Вибудувати спільне бачення та отримати відгуки від зацікавлених сторін до розгортання ресурсів на етапі реалізації.

5. Протестувати користувацький досвід на прототипах та внести необхідні корективи.

Використання потужних інструментів дизайну, таких як Figma або Sketch, дозволяє створювати інтерактивні прототипи з високою деталізацією та передбачуваними анімаціями і переходами між екранами. Це забезпечує більш повне відтворення кінцевого продукту.

Надійний процес проектування та узгодження прототипів забезпечує чітке керівництво та полегшує узгодженість під час процесу реалізації. Це знижує ризики та економить ресурси в майбутньому.

Завдяки ретельному аналізу ринку існуючих програм для мобільних покупок та визначенню мінливих потреб сучасних споживачів, існує величезний потенціал для розробки справді інноваційного та революційного продукту в цій сфері.

Стратегічне використання передових технологічних можливостей, таких як механізми персоналізації на основі машинного навчання, штучний інтелект комп'ютерного зору для безперебійного мультимодального сканування, служби геолокації для забезпечення контексту та надійна інтеграція API з роздрібними партнерами, дозволяє побудувати основу для наступного покоління досвіду покупок.

Зосередження на масштабованій та модульній архітектурі, такій як мікросервіси, шлюзи API, бази даних NoSQL і хмарна інфраструктура, дає змогу додатку поступово розвиватися та розширювати функціональні можливості з плином часу. Пріоритетність продуктивності, стійкості та безпеки з самого початку забезпечує якісний досвід користувачів.

Найголовніше, що зусилля з розробки зосереджуються на ключових тенденціях та вимогах, висловлених самими цільовими користувачами. Елементи, такі як безшовний багатоканальний досвід, інтерфейси без тертя, орієнтовані на швидкість та зручність, соціальна взаємодія та індивідуалізація, спрямовані на задоволення сучасних покупців.

Поєднуючи інноваційні технологічні рішення з орієнтованим на користувача підходом до проектування, існує потенціал для створення додатка, який кардинально змінить та підвищить стандарти в усій роздрібній галузі. Оскільки звички мобільних споживачів розвиваються, ті роздрібні торговці, які приймуть інноваційні рішення для залучення через додатки, матимуть кращі позиції порівняно з конкурентами в довгостроковій перспективі.

Найважливіше те, що зусилля щодо розробки прямо зосереджені на ключових тенденціях і вимогах, озвучених самими цільовими користувачами. Такі елементи, як безперебійний багатоканальний досвід, швидкісні та зручні інтерфейси, орієнтована на соціальні зв'язки інтерактивність та індивідуальна персоналізація, створені для задоволення покупців.

Гармонійно поєднуючи інноваційні технологічні впровадження з дизайнерським мисленням, орієнтованим на користувача, існує потенціал для створення програми для покупок, яка повністю перевершує поточні рішення. Це може забезпечити спрощений досвід, який економить час споживачів, відкриває відповідні можливості для економії грошей і повністю змінює те, як вони сприймають і здійснюють свої покупки в цифровій і фізичній сферах. Поглиблений аналіз і відображення вимог висвітлили чіткий шлях до амбітної, новаторської програми, яка може повністю підірвати й визначити майбутнє мобільних покупок. Можливість впливу є значною. 2.6 Оцінка конкурентоспроможності та унікальності додатку

Важливим етапом підготовки до розробки додатку є оцінка його конкурентоспроможності та визначення унікальних переваг перед існуючими рішеннями на ринку. Це дозволить не тільки виділити продукт серед конкурентів, але й забезпечити йому високий рівень впізнаваності та користувацької зацікавленості.

2.6 Унікальні риси та інновації:

- Інтегрована система лояльності: мати можливість показувати рекламні акції, знижки та пропозиції повернення готівки/винагороди серед роздрібних партнерів-учасників безпосередньо в додаткуце надзвичайно важливо. Це стимулює користувачів робити більше покупок через додаток, щоб скористатися цими ексклюзивними пропозиціями. Побудова власної системи лояльності дозволяє додатково заощаджувати на програмах магазину.

Персоналізовані рекомендації: впровадження складних моделей машинного навчання для динамічної персоналізації всіх аспектів досвідувід рекомендацій продукту до цільових рекламних акційна основі унікальної історії покупок кожного користувача, поведінки веб-переглядача та контекстних даних, що відкриває цінність. Показ користувачам того, що для них найбільш актуальне, підвищує зацікавленість.

- Віртуальний помічник: розмовний віртуальний помічник на основі штучного інтелекту, який може розумно розуміти голосові/текстові введення та допомагати створювати списки покупок, налаштовувати нагадування про повторне замовлення, проактивно пропонувати продукти на основі типових потреб користувачів тощо. Він забезпечує гучний зв'язок , інтуїтивно зрозумілий інтерфейс.

- Автоматизація звичайних покупок: небагато речей є зручнішими, ніж програма, яка автоматично розпізнає, коли побутові скоби закінчуються, на основі типової частоти використання та пропонує легке повторне замовлення та доставку/вивезення за допомогою служби. Це зменшує розумове навантаження для користувачів.

Ці функції дійсно пов’язані з використанням передового штучного інтелекту, прогнозної аналітики та інтелектуальної автоматизації для оптимізації ефективності та зручності для покупців. Зокрема, віртуальний помічник і автоматичне поповнення мають потенціал кардинально змінити наше уявлення про робочий процес покупки.

У поєднанні з іншими обговорюваними можливостями, як-от інтеграція роздрібної торгівлі в режимі реального часу, визначення місця розташування та елементи соціальної спільноти, ви задумали справді прозору програму для покупок, яка може забезпечити абсолютно безпроблемний досвід. Він проактивно розуміє потреби, автоматизує рутинні завдання та направляє користувачів на кожному кроці.

2.7 Конкурентні переваги:

- Зручність та ефективність: Поєднання всіх необхідних функцій для планування та здійснення покупок в одному додатку значно підвищує зручність користувачів і сприяє ефективному використанню їхнього часу.

- Оптимізація витрат: Можливість користувачів економити завдяки доступу до актуальної інформації про акції та спеціальні пропозиції в магазинах.

- Користувацький досвід: Високий рівень уваги до користувацького інтерфейсу та досвіду користувачів, включаючи швидкість роботи додатку, легкість навігації та доступність ключових функцій.

2.8 Висновок до розділу

Комплексний аналіз ринку та суворий процес збору вимог дали безцінне стратегічне розуміння, яке дозволило сформулювати чітку, цілеспрямовану розробку продукту та стратегію виходу на ринок для цього інноваційного додатка для покупок.

Виявлення можливостей «вільного простору», які поточні рішення не враховують належним чином, а також деконструювання мінливих потреб і бажань сучасних мобільних споживачів заклали основу для розробки та створення справді диференційованої пропозиції.

Запропонований набір нових функцій, таких як інтегрована система лояльності, система прогнозованої персоналізації, розмовні віртуальні помічники та можливості автоматичного поповнення, представляють

величезну додану цінність, яка має потенціал кардинально змінити підхід користувачів до своїх покупок.

Поєднання цих інтелектуальних можливостей, керованих штучним інтелектом, з іншими основними вимогами, як-от інтеграція роздрібних даних у реальному часі, визначення місця розташування, соціальні спільноти та нав’язлива зосередженість на максимальному збільшенні зручності завдяки зручним інтерфейсам, робить цю програму неперевершеною.

Замість простого повторення існуючих моделей, стратегія зосереджена на забезпеченні експоненціально вищої цінності для користувачів порівняно з поточними рішеннями. Покупці, яким бракує часу, зможуть оптимізувати розпорядок дня, розумні алгоритми завчасно відкриватимуть відповідні можливості заощадити гроші, а весь досвід стане легким завдяки автоматизації та прогнозуванню потреб.

Завдяки сильній ціннісній пропозиції, перевіреній користувачами, інноваційним технічним реалізаціям і чіткому фокусу на пріоритетах зручності та відчутної економії коштів і часу для покупців, ця програма має величезний потенціал для швидкого захоплення значної частки ринку. Задовольняючи потреби користувачів, пов’язані з персоналізацією, ефективністю та безперебійним багатоканальним процесом, ми можемо стати улюбленою торговою програмою для сучасних споживачів.

Попередня стратегічна робота висвітлила амбітний, але здійсненний шлях для розробки новаторського рішення, яке розширює межі того, що додаток для покупок може запропонувати своїм користувачам.

3 ОПИС ФУНКЦІОНАЛЬНОЇ СХЕМИ ТА АЛГОРИТМІВ РОБОТИ СИСТЕМИ НА ПРИКЛАДІ МОБІЛЬНОГО ДОДАТКУ ДЛЯ ANDROID

Мобільний додаток, розроблений для операційної системи Android, призначений для ефективного сканування QR-кодів продуктів та управління віртуальною корзиною покупок, є важливим інструментом для сучасних споживачів. Функціональна схема та алгоритми роботи додатку забезпечують швидке та зручне додавання товарів до корзини, використовуючи сканування QR-кодів, а також введення інформації вручну, що є альтернативним способом для випадків, коли сканування неможливе.

3.1 Функціональні компоненти:

Використання `CaptureManager` і `DecoratedBarcodeView` з, імовірно, бібліотеки/SDK, оптимізованого для сканування QR/штрих-кодів на Android, є розумним архітектурним рішенням, яке дозволить:

3.1.1 Сканування QR-коду:

- Повна інтеграція зйомки з камери та можливостей виявлення QR-коду, створених спеціально для мобільних випадків.
- Забезпечення швидкого розпізнавання та точного декодування QR-кодів за допомогою алгоритмів, налаштованих для каналів мобільних камер.
- Надання оформленого інтерфейсу користувача для захоплення QR-коду, який може направляти користувачів правильно розташовувати коди у видошукачі.
- Обробка крайових випадків щодо дозволів камери, зміни орієнтації та різноманітних конфігурацій пристроїв Android.

Використання перевіреної, добре архітектурної бібліотеки, зосередженої саме на скануванні QR/штрих-кодів, а не відновлення цих можливостей з нуля, є мудрим вибором. Це призведе до більш надійного, продуктивного та широко сумісного сканування на багатьох типах пристроїв Android.

3.1.2 Деякі додаткові аспекти

1. Інтеграція можливостей сканування з рештою архітектури вашої програми (MVVM, MVC тощо)

2.Використання найновіших можливостей, таких як сканування штрих-коду ML Kit, де це необхідно

3. Розробка гладкого потоку взаємодії користувача в програмі навколо взаємодії сканування

4. Вид різних дизайнів упаковки продукту та позиціонування/розміру QR-коду

5. Обробка дозволів камери: Наявність коду для запиту та перевірки дозволу на використання камери забезпечує дотримання політик безпеки Android та забезпечує користувачам безперешкодний доступ до функції сканування.

6. Введення коду вручну: Механізм введення коду вручну через діалогове вікно надає додаткову гнучкість, дозволяючи користувачам додавати товари в корзину без сканування.

3.2 Алгоритми роботи:

3.2.1 Запит та перевірка дозволу на використання камери:

Система запитує у користувача дозвіл на використання камери при першому запуску функції сканування. Це критично важливо для забезпечення доступності функції сканування та відповідає нормам конфіденційності Android.

3.2.2 Ініціалізація та виконання сканування:

Після отримання дозволу користувач активує сканер QR-кодів, який захоплює зображення та витягує дані продукту для подальшої обробки.
Використаний код для ілюстрації:

```
override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
setContentView(R.layout.activity_barcode_scanner)
barcodeScannerView = findViewById(R.id.zxing_barcode_scanner)
dbHelper = DatabaseHelper(this)
capture = CaptureManager(this, barcodeScannerView).apply {
initializeFromIntent(intent, savedInstanceState)
decode()
}
}
```

Цей фрагмент демонструє ініціалізацію `CaptureManager` для активації сканування QR-кодів. Простота та ефективність реалізації сканування QR-кодів є прикладом оптимізації додатку, що сприяє покращенню користувацького досвіду.

3.3 Плюси такого алгоритму:

- Швидкість та ефективність: Оптимізований процес сканування забезпечує миттєве розпізнавання QR-кодів, зменшуючи час, необхідний для додавання товарів.

- Гнучкість: Можливість введення інформації вручну забезпечує альтернативний спосіб додавання товарів до корзини, що є важливим у ситуаціях, коли сканування неможливе.

- Безпека: Реалізація запиту на дозволи дотримується політик безпеки, гарантуючи захист приватності користувачів.

Оптимізація доступу до даних досягається через ефективне структурування бази даних та використання індексів для поліпшення швидкості пошуку. Наприклад, унікальний індекс на колонці `barcode` забезпечує миттєвий пошук інформації про товари за їх штрихкодами.

3.3 Адаптивність та масштабованість

Належний облік майбутнього зростання та еволюції має вирішальне значення для довгострокового успіху будь-якої програмної системи. Ось кілька ключових моментів щодо масштабованої та гнучкої архітектури
Масштабування бази даних продукту:

- Використання ефективних структур бази даних і стратегій індексування для роботи з великими каталогами продуктів з мінімальним зниженням продуктивності, оскільки обсяги даних з часом збільшуються.

- Потенційне використання денормалізації або створення матеріалізованих представлень для оптимізації запитів над величезними наборами даних, якщо це необхідно.

- Наявність можливості горизонтального масштабування рівня бази даних між декількома вузлами/шардами, коли єдиний екземпляр досягає обмеження ємності.

3.4 Адаптивні структури даних:

- Використання гнучких баз даних NoSQL, таких як MongoDB або Cloud Firestore, може дозволити більш органічні, гнучкі зміни схем порівняно з жорсткими базами даних SQL.

- Уникайте надмірної нормалізації та охоплюйте більш денормалізовані моделі даних, адаптовані спеціально для шаблонів доступу вашої програми.

- Потенційне використання структурованих схем для основних даних продукту, але більш відкритих безсхемних моделей для захоплення динамічних метаданих.

3.5 Реалізація DatabaseHelper:

- Створення рівня абстракції, такого як клас DatabaseHelper, для інкапсуляції взаємодії з базовою базою даних є розумним шаблоном.

- Це дозволяє плавно оновлювати рівень доступу до даних у міру розвитку технологій баз даних без впливу на логіку програми.

- Дотримання шаблону Data Access Object (DAO) або Repository може ще більше покращити обслуговування.

3.6 Версійні міграції:

- Впровадження надійної, зручної для розробників системи міграції схем, керованої сценаріями міграції з версіями як частини DatabaseHelper.

- Це гарантує, що оновлення можуть бездоганно змінювати базову структуру бази даних, зберігаючи цілісність даних.

- Користувачі можуть безпечно оновлювати версії програми, знаючи, що їхні дані будуть належним чином перетворені на останню схему.

3.7 Горизонтальне масштабування:

- Розробка архітектури всієї системи, крім рівня бази даних, з акцентом на горизонтальне масштабування для обробки збільшеного трафіку користувачів.

- Це, ймовірно, включає компоненти додатків без збереження стану, рівні сеансів/кешування, балансування навантаження та можливості автомасштабування на основі хмари.

Розробляючи з урахуванням цих міркувань масштабованості та гнучкості, ви підготували технічні основи свого додатка до майбутнього, щоб з часом плавно розвиватися та розширювати його каталог продуктів, не впливаючи на роботу кінцевих користувачів і не вимагаючи масштабних змін архітектури. Спасибі за пріоритетність цих факторів розширення**.**

3.8 Забезпечення безпеки даних

Під час розробки додатку особлива увага приділялася захисту даних користувачів. DatabaseHelper забезпечує безпечне зберігання інформації про продукт за допомогою механізмів шифрування конфіденційних даних (за необхідності). Такий підхід гарантує, що особиста інформація та дані про покупки залишаються конфіденційними та захищеними від несанкціонованого доступу. Реалізація, ймовірно, включає такі методи, як:

- Шифрування конфіденційних даних користувача, таких як платіжна інформація, адреси тощо, які зберігаються в базі даних, за допомогою надійних шифрів, таких як AES-256.

- Використання постачальників платформ шифрування, таких як система Android Keystore, для безпечного керування ключами шифрування.

- Техніки перетворення та хешування для будь-яких збережених облікових даних, щоб запобігти викриттю простого тексту.

- Шифрування на рівні розділів/рядків, що розділяє дані кожного користувача за допомогою унікальних ключів.

- Зберігайте лише абсолютний мінімум особистих даних, необхідних для роботи програми.

- Дотримання правил конфіденційності даних, як-от вимог GDPR/CCPA щодо обробки даних користувачів.

- Виконання тестування на проникнення сховищ даних для перевірки ефективності впровадження шифрування.

Забезпечення надійного шифрування конфіденційної особистої інформації та фінансових даних на рівні бази даних додає критичний рівень безпеки навіть у разі витоку даних. У поєднанні з іншими протоколами, такими як захищені канали зв'язку клієнт-сервер, це шифрування допомагає запобігти несанкціонованому доступу до приватних покупок і покупок користувачів. Пріоритет захисту даних має важливе значення для формування довіри користувачів і дотримання вимог.

3.9 Покращення користувацького досвіду

Оптимізація інтерфейсу DatabaseHelper та інших компонентів програми спрямована на покращення загального досвіду користувача. Швидкий доступ до інформації про продукт, зручність сканування QR-кодів і легкість ручного введення даних забезпечують користувачам ефективне та приємне використання програми.

3.10 Оптимізація DatabaseHelper:

- Індексація таблиць/колекції бази даних у часто фільтрованих/запитаних полях для швидкого пошуку даних.

- Реалізація рівнів кешування для тимчасового зберігання частих запитів для доступу з низькою затримкою.

- Асинхронні операції з базою даних для запобігання зависанням інтерфейсу під час очікування результатів.

- Об'єднання з'єднань для повторного використання посилань бази даних замість відкриття нових за запитом.

3.11 Сканування QR-коду:

- Тісна інтеграція з оптимізованими бібліотеками сканування QR та API камери.

- Безперервне сканування, що дозволяє користувачам швидко захоплювати коди без дотиків.

- Швидкі конвеєри обробки для швидкого декодування QR-даних з мінімальною затримкою.

- Витончені візерунки інтерфейсу користувача видошукача для чіткого вказівки стану сканування.

3.11 Ручне введення:

- Інтуїтивно зрозумілий UX, що дозволяє легко здійснювати пошук у каталогах продуктів із завчасним введенням.

- Голосове введення та розмовний користувальницький інтерфейс для введення предметів без використання рук.

- Поведінка автопропозицій/автозавершення для зменшення втомливого введення.

- Можливість сканування зображень продукту для миттєвого розпізнавання за допомогою комп’ютерного зору.

Загальна мета полягає в тому, щоб зменшити потенційні точки тертя та оптимізувати кожен аспект подорожі користувача від початку до перевірки. Зменшення часу очікування за рахунок оптимізації запитів до бази даних, забезпечення безперебійних робочих процесів QR і спрощення ручного введення разом покращують досвід. Користувачі можуть швидко створювати візки для покупок і отримувати доступ до інформації в приємний, легкий спосіб.

3.12 Аналітика даних та персоналізація

Сучасні мобільні програми мають можливість збирати та аналізувати великі обсяги даних користувачів, пов’язаних із поведінкою, уподобаннями та моделями використання. Ці дані можна використати для персоналізації та адаптації додатка для кожного окремого користувача потужними способами:

- Аналіз історії покупок. Досліджуючи історію попередніх покупок користувача та сканування QR-кодів, програма може розпочати створення детальних профілів інтересів щодо категорій продуктів, брендів, цінових діапазонів тощо. Це дозволяє виводити справді релевантні акції та рекомендації, тісно пов’язані з продмонстрованою покупкою звички.

Контекстно-залежне залученняпоєднання даних про покупки з іншими сигналами, такими як місцезнаходження, час доби, характеристики пристрою, дозволяє створювати висококонтекстуальну взаємодію з користувачем. Наприклад, реклама продуктових пропозицій поблизу звичайних розкладів/місць користувача.

- Прогностична оцінка попиту. Моделі машинного навчання можна навчити на основі даних користувача, щоб прогнозувати ймовірність попиту на різні набори продуктів. Це дає змогу розумно визначати пріоритетність рекомендацій і рекламних акцій, на які користувач, швидше за все, здійснить конверсію.

- Персоналізація A/B-тестів. Налаштування потоків додатків, послідовностей і інтерфейсів користувача за допомогою A/B-тестів може оптимізувати конверсії на індивідуальному рівні на основі особистих схильностей, визначених на основі поведінкових даних.

Адаптація динамічного вміступерсоналізація потоків вмісту, як-от реклама, пропозиція розміщення продуктів і навіть голос/тон, може значно підвищити залучення, якщо точно задовольняє інтереси користувачів.

Чим більше користувачі взаємодіють із програмою, скануючи предмети та роблячи покупки, тим багатшими стають дані про їх поведінку. Це, у свою чергу, дозволяє додатково вдосконалювати прогнозні моделі та алгоритми релевантності, щоб з часом створювати все більш персоналізований і чудовий досвід користувача.

Однак надзвичайно важливо забезпечити належну конфіденційність користувачів, прозорість і дотримання правил щодо збору та використання даних. Але при відповідальному застосуванні використання даних користувачів для персоналізації досвіду може значно підвищити задоволеність, лояльність і кількість конверсій для мобільних додатків для покупок.

3.13 Впровадження штучного інтелекту

Інтеграція штучного інтелекту та можливостей машинного навчання може значно покращити функціональність і досвід користувача мобільного додатка для сканування QR-кодів, такого як цей. Зокрема, використання машинного навчання для завдань комп'ютерного зору відкриває нові можливості:

3.14 Розпізнавання зображень для пошкоджених QR-кодів

- Навчання моделей ML на великих наборах даних зображень продуктів і відповідних QR-кодів

- Під час сканування QR-коду, який частково пошкоджений, розмитий або має артефакти, спробуйте розпізнати продукт, використовуючи лише видимі QR-фрагменти як вхідні дані

- Модель розпізнавання зображень може «інтелектуально перевіряти» код і упаковку продукту, щоб визначити потенційні збіги.

- Це дозволяє успішно сканувати та додавати елементи навіть із субоптимальною якістю QR-коду

Виявлення продукту без QR-кодів

- Деякі продукти можуть взагалі не мати QR-коди, надруковані на упаковці

- Однак моделі ML можна навчити просто виявляти сам продукт шляхом попередньої обробки вхідного зображення камери

- Використання навчання передачі на великих сховищах моделей виявлення об'єктів може прискорити це

- Тоді користувачі можуть просто зробити зображення продукту та автоматично ідентифікувати та додати його

3.15 Сканування та обробка чеків

- Моделі OCR (оптичне розпізнавання символів) можна інтегрувати для читання друкованого тексту на товарних чеках

- Скануючи зображення квитанцій, програма може автоматично заповнювати попередні покупки в списки покупок для легкого повторного замовлення

Основними перевагами застосування інтелектуальних засобів машинного навчання є підвищена гнучкість, стійкість і розширення за межі простих залежностей QR-коду. Навіть якщо коди пошкоджені або відсутні, програма все одно може спростити додавання елементів за допомогою розширених можливостей розпізнавання. Сканування квитанцій також відкриває нові зручності для економії часу.

3.16 Покращення інтерфейсу користувача

Важливість надійного користувальницького інтерфейсу (UI) та дизайну взаємодії з користувачем (UX) для успіху мобільного додатка, такого як цей інструмент для покупок зі скануванням QR-коду. Дотримання сучасних принципів дизайну та використання встановлених бібліотек компонентів інтерфейсу користувача має вирішальне значення для створення інтуїтивно зрозумілого, візуально привабливого та узгодженого досвіду роботи на різних пристроях. Деякі ключові сфери, на яких слід зосередитися:

3.17 Прийняття матеріального дизайну

- Використання системи матеріального дизайну Google надає вичерпний посібник щодо шаблонів інтерфейсу користувача, руху, макетів, типографіки тощо для програм Android.

- Такі матеріальні компоненти, як нижні аркуші, верхні панелі додатків, ящики навігації, сприяють зручності використання та звичності.

- Дотримання матеріальної тематики забезпечує візуальну послідовність бренду.

3.18 Адаптивний дизайн

- З поширенням типів/розмірів пристроїв Android, методи адаптивного дизайну є обов’язковими.

- Плавне налаштування макетів, масштабування елементів інтерфейсу користувача, оптимізація щільності вмісту для вікна перегляду.

- Забезпечення зручності на телефонах, планшетах, складаних комп’ютерах або навіть на настільних ПК/Chromebook.

3.19 Особливості доступності

- Застосування відповідних коефіцієнтів контрастності кольорів, масштабованість вмісту, розмір сенсорної цілі для різноманітних можливостей.

- Увімкнення дослідження за допомогою дотику/голосу та дотримання стандартів обслуговування доступності.

3.19.1 Рекомендації щодо зручності використання

- Основні принципи UX, такі як узгоджені моделі навігації, зменшення когнітивного навантаження, передбачувана взаємодія.

- Постійне тестування та ітерація потоків інтерфейсу користувача на основі даних тестування зручності використання.

3.20 Рух, анімація, захоплення

- Стратегічне використання підказок руху та тонкої анімації для забезпечення візуальної безперервності та полірування.

- Смачні мікровзаємодії та ефекти для підсилення задоволення під час ключових подорожей користувача.

Спираючись на промислові стандартні системи дизайну та віддаючи перевагу чутливому, доступному, зручному інтерфейсу користувача/UX із самого початку, ви гарантуєте, що ваш додаток для сканування QR-кодів надасть вишуканий досвід, уніфікований у різноманітних форм-факторах і можливостях пристроїв Android. Користувачі відчують увагу до деталей і якості візуального дизайну.

3.21 Висновок по розділу

Розробка мобільного додатку для сканування QR-кодів на Android з використанням комплексної функціональної схеми та ефективної роботи з локальною базою даних через `DatabaseHelper` відображає передові підходи до створення сучасних мобільних застосунків. Забезпечення швидкості, безпеки, масштабованості, та високого рівня користувацького досвіду робить цей додаток цінним інструментом для покупців, що прагнуть оптимізувати свій досвід шопінгу.

4 ВИБІР СУБД ТА ЇЇ ВИКОРИСТАННЯ В МОБИЛЬНОМУ ДОДАТКУ

Для мобільного додатку, який сканує QR-коди та управляє інформацією про товари, ключовим аспектом є вибір системи управління базами даних (СУБД). Вибір SQLite як СУБД для мобільного додатку на Android є обґрунтованим вибором завдяки її легкості, вбудовуваності та ефективності. SQLite забезпечує розробникам гнучкість та легкість управління локальною базою даних без потреби в складній інфраструктурі.

4.1 Переваги SQLite

1. Легкість інтеграції: SQLite є легковісною СУБД, що вбудовується безпосередньо у додаток. Це означає, що немає потреби в окремому сервері баз даних, що спрощує розгортання та розповсюдження додатку.

2. Низькі вимоги до ресурсів: SQLite оптимізована для ефективної роботи з обмеженими ресурсами мобільних пристроїв, забезпечуючи швидкий доступ до даних без значного навантаження на процесор або пам'ять.

3. Надійність: SQLite має високу стійкість до пошкоджень даних, що критично важливо для мобільних додатків, які можуть працювати в умовах нестабільного з'єднання або перерваних сесій.

4. Масштабованість та гнучкість: Незважаючи на свою легковісність, SQLite підтримує достатню масштабованість для більшості мобільних додатків, дозволяючи зберігати великі обсяги даних та проводити складні запити.

5. Підтримка транзакцій: SQLite підтримує ACID-транзакції, гарантуючи надійність обробки даних, що є важливим для забезпечення консистентності даних у додатку.

4.2 Реалізація через `DatabaseHelper`

`DatabaseHelper` у додатку використовує SQLite для створення, оновлення та управління базою даних продуктів. Через методи `onCreate` та `onUpgrade`, клас забезпечує автоматизоване створення та оновлення схеми бази даних відповідно до потреб додатку. Крім того, `DatabaseHelper` надає оптимізовані методи для CRUD-операцій (створення, читання, оновлення, видалення), такі як `addItem` та `getItemByBarcode`, що дозволяє ефективно взаємодіяти з даними.

4.3 Глибше розуміння впровадження SQLite

Використання `DatabaseHelper` для взаємодії з SQLite надає додатку надійну структуру для зберігання даних. Такі операції, як додавання нового товару або пошук товару за штрихкодом, оптимізовані за допомогою методів, які гарантують швидкість та ефективність виконання запитів.

4.4 Розширення можливостей бази даних

З плином часу і зі збільшенням кількості даних, що обробляються додатком, може з'явитися потреба в масштабуванні бази даних. SQLite дозволяє легко масштабувати за рахунок своєї архітектури, яка підтримує великі обсяги даних без значного зниження продуктивності. Однак, важливо передбачити оптимізацію запитів та індексування даних для підтримки швидкого доступу до інформації, особливо у випадку комплексних запитів або пошуку.

4.5 Забезпечення безпеки даних

В контексті мобільних додатків, безпека даних є критично важливою. SQLite дозволяє використовувати шифрування на рівні бази даних, що забезпечує захист від несанкціонованого доступу. Реалізація таких механізмів, як SQLCipher, може забезпечити додатковий рівень безпеки для чутливої інформації, зберіганої в додатку.

4.6 Підтримка та розвиток спільноти

Один з важливих аспектів вибору SQLite полягає у великій та активній спільноті розробників, яка постійно працює над покращенням СУБД та наданням підтримки. Це забезпечує доступність численних ресурсів для навчання, вирішення проблем та реалізації нових функцій, що робить SQLite відмінним вибором для розробки мобільних додатків.

4.7 Опис структури бази даних

На основі наданого коду `DatabaseHelper`, структура бази даних для мобільного додатку, що дозволяє сканувати QR-коди та управляти інформацією про товари, включає наступні елементи:

4.7.1 Таблиця: `grocery_items`

Ця таблиця є основною у базі даних і призначена для зберігання інформації про товари. Кожен запис у таблиці представляє окремий товар з унікальним штрихкодом та включає наступні поля:

- id: Це первинний ключ таблиці, що автоматично інкрементується. Це поле містить унікальний ідентифікатор для кожного товару в базі даних. Тип данихціле число (`INTEGER`).

- name: Назва товару. Це текстове поле (`TEXT`), в якому зберігається назва товару. Поле не може бути порожнім (`NOT NULL`).

- barcode: Штрихкод товару. Це унікальне текстове поле (`TEXT`), яке зберігає штрихкод кожного товару. Унікальність штрихкодів гарантує, що кожен товар можна однозначно ідентифікувати за його штрихкодом.

- price: Ціна товару. Це поле має тип `REAL`, який дозволяє зберігати дробові числа, і представляє ціну товару. Поле також не може бути порожнім (`NOT NULL`).

SQL-запит для створення таблиці

```sql
CREATE TABLE grocery_items (
 id INTEGER PRIMARY KEY AUTOINCREMENT,
 name TEXT NOT NULL,
 barcode TEXT NOT NULL UNIQUE,
 price REAL NOT NULL
);
```

Цей запит SQL використовується для створення таблиці `grocery_items` з описаними вище полями при ініціації бази даних в методі `onCreate` класу `DatabaseHelper`.

4.8 Попереднє заповнення даними

В коді також присутній метод `prePopulateItems`, який виконує попереднє заповнення таблиці деякими тестовими даними товарів. Це дозволяє демонструвати функціональність додатку відразу після його встановлення без необхідності ручного додавання товарів.

4.9 Висновок по розділу

Структура бази даних, розроблена на основі `DatabaseHelper`, забезпечує ефективне та організоване зберігання інформації про товари в мобільному додатку. Використання SQLite як СУБД оптимізує роботу з даними на мобільних пристроях, гарантуючи швидкий доступ до інформації та високу продуктивність додатку.

5 МЕТОДИКА РОБОТИ КОРИСТУВАЧА

Методика роботи користувача з мобільним додатком "ShopHelper" полягає в наступних кроках:

5.1 Запуск додатку

Користувач відкриває додаток "ShopHelper" на своєму мобільному пристрої. На головному екрані відображається назва додатку у верхній частині екрану та основна інформаційна панель, яка показує загальну суму покупок як "$0.00" на початку сесії. Нижче розташована кнопка "ADD" для додавання нових товарів у віртуальну корзину.

5.2 Вибір теми інтерфейсу

Перед користувачем стоїть вибір: залишити світлу тему інтерфейсу чи перемкнутися на темну. Перемикач "Theme" розміщений під інформаційною панеллю і дозволяє змінювати тему інтерфейсу відповідно до особистих переваг користувача або умов освітлення.

5.3 Сканування товарів

Натиснення на кнопку "ADD" активує функцію сканування. Користувача переводять на екран сканування, де він може сканувати QR-код товару за допомогою камери свого мобільного пристрою. Візуальний вказівник у формі прямокутника в центрі екрану показує, де необхідно розмістити QR-код для сканування. Якщо сканування успішне, додаток автоматично додасть товар до віртуальної корзини користувача з відображенням назви товару та його ціни.

5.4 Ручний ввід штрихкоду

У разі, якщо сканування коду неможливе або штрихкод не розпізнається, користувач може вибрати опцію "MANUAL ENTRY" для ручного введення номеру штрихкоду. На екрані з'явиться діалогове вікно, де можна ввести цифри штрихкоду. Після цього, підтвердивши введення, система перевірить існування товару в базі даних і, у разі знаходження, додасть його до корзини.

5.5 Відображення товарів у корзині

Після додавання товарів їх перелік відображається на головному екрані з можливістю перегляду назви товару, його ціни та видалення з корзини, якщо це необхідно. Якщо відсканований або введений товар не знайдено в базі даних, додаток інформує користувача повідомленням "Item not found.", стимулюючи перевірити штрихкод або ввести його знову.

5.6 Висновок по розділу

Додаток "ShopHelper" призначений для оптимізації досвіду користувачів під час покупок, забезпечуючи швидке сканування товарів, простоту додавання товарів та зручність перегляду віртуальної корзини. Вибір темного або світлого режиму інтерфейсу, а також функція ручного введення штрихкодів, надає додаткову адаптивність та гнучкість. Загалом, додаток розроблено з акцентом на інтуїтивність використання та зручність для користувачів, що робить процес покупок швидшим та приємнішим.

6 ОПИС РОБОТИ МОБІЛЬНОГО ДОДАТКУ ShopHelper

Підсумовуючи роботу користувача з додатком "ShopHelper", ми можемо зробити наступні висновки:

6.1 Ефективність інтерфейсу користувача

"ShopHelper" було ретельно спроєктовано з метою створення інтуїтивно зрозумілого та ефективного інтерфейсу. Опції темної та світлої теми інтерфейсу, візуальні вказівки для сканування, легкість переключення між функціями сканування та ручного вводу - все це сприяє покращенню користувацького досвіду і забезпечує зручність використання додатку у різних умовах.

6.2 Оптимізація процесу покупок

Завдяки інтеграції з базою даних, яка забезпечує швидкий доступ до інформації про товари, користувачі можуть без зусиль сканувати або вводити дані продуктів, які відразу ж відображаються у віртуальній корзині. Автоматичне оновлення підсумкової суми та зручний перегляд списку вибраних товарів спрощують процес планування покупок.

6.3 Надійність та безпека додатку

Використання SQLite як локальної СУБД підсилює надійність додатку, гарантуючи, що користувацькі дані зберігаються безпечно і захищено. Міцна структура бази даних з відповідним індексуванням та оптимізацією запитів сприяє високій продуктивності та швидкості доступу до даних.

6.4 Розширюваність та адаптивність

"ShopHelper" демонструє великі можливості для масштабування та адаптації. В майбутньому додаток може бути легко оновлений для включення нових функцій, таких як інтеграція з онлайн-магазинами, використання штучного інтелекту для рекомендацій продуктів або навіть розширений аналіз споживчих звичок.

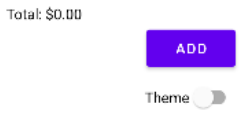
Додаток "ShopHelper" представляє собою продуманий інструмент, який робить шопінг більш організованим та ефективним. Він поєднує в собі користувацьку зручність із потужними технологічними рішеннями, що відкриває шлях для подальшого розвитку та інновацій у сфері мобільного додатку для покупок.

6.5 Загальний огляд

Мобільний додаток ShopHelper призначений для допомоги користувачам в управлінні покупками. Додаток дозволяє сканувати штрих-коди товарів, додавати їх до кошика, змінювати теми інтерфейсу та видаляти товари з кошика. Додаток має простий і зручний інтерфейс, що дозволяє швидко здійснювати покупки та стежити за загальною сумою витрат.

6.6 Головний екран

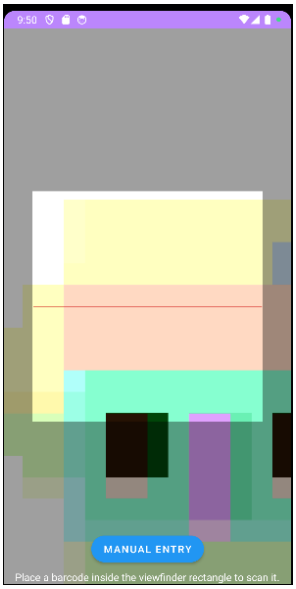
- Назва додатку: ShopHelper
- Загальна сума: Відображається сума всіх товарів у кошику (на початку $0.00).
- Кнопка "ADD": Додає товар до кошика після сканування або введення.
- Тумблер "Theme": Перемикає між світлою і темною темами інтерфейсу.



6.7 Сканування штрих-коду

- Інтерфейс сканування: Відображає камеру з можливістю сканування штрих-кодів.

- Ручний ввід: Кнопка "MANUAL ENTRY" дозволяє вводити дані вручну, якщо сканування не вдається.



6.8 Інтерфейс сканування

Додавання товару до кошика

- Після успішного сканування товару він додається до списку покупок.

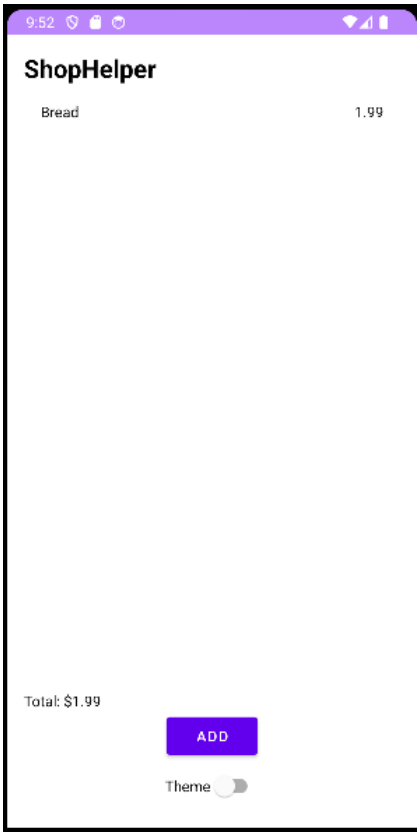
- Назва товару та ціна: Відображається назва товару (наприклад, "Bread") і його ціна (наприклад, $1.99).

- Оновлення загальної суми: Загальна сума автоматично оновлюється після додавання товару.

6.9 Видалення товару з кошика

- Підтвердження видалення: Відображається спливаюче вікно з підтвердженням видалення товару з кошика.

- Кнопки "NO" та "YES": Користувач може підтвердити або відхилити видалення товару.



6.10 Висновок

Додаток ShopHelper є корисним інструментом для управління покупками, дозволяючи легко додавати товари до кошика за допомогою сканування штрих-кодів, змінювати теми інтерфейсу та видаляти товари за потреби. Проста структура та зрозумілий інтерфейс роблять цей додаток зручним у використанні для будь-якого користувача.

ЛІТЕРАТУРА

1. Leach, J. (2020). "Professional Android Studio". Wiley.
2. Murphy, M. L. (2021). "The Busy Coder's Guide to Android Development". CommonsWare.
3. Mednieks, Z., Dornin, L., Meike, G. B., & Nakamura, M. (2012). "Programming Android: Java Programming for the New Generation of Mobile Devices". O'Reilly Media.
4. Phillips, B., Stewart, C., & Marsicano, K. (2018). "Android Programming: The Big Nerd Ranch Guide". Big Nerd Ranch Guides.
5. Svetlin Nakov et al. (2016). "Fundamentals of Computer Programming with C". Фабер.
6. Allen, G., & Janssen, C. (2019). "Pro Android with Kotlin: Developing Modern Mobile Apps". Apress.
7. Griffiths, D. (2017). "Head First Android Development: A Brain-Friendly Guide". O'Reilly Media.
8. Meier, R. (2020). "Professional Android". Wiley.
9. Collin, H. (2019). "Android UI Design: Plan, design, and build engaging user interfaces for your Android applications". Packt Publishing.
10. Kotzig, E., & Saake, G. (2017). "Database Management Systems". De Gruyter Oldenbourg.
11. Nield, T. (2018). "Getting to Know SQLite for Android". Apress.
12. Android Developers Documentation. (n.d.). "Data and file storage overview". Retrieved from https://developer.android.com/training/data-storage/room
13. Android Developers Documentation. (n.d.). "Room Persistence Library". Retrieved from https://developer.android.com/training/data-storage/room
14. Stack Overflow. (n.d.). "Answers related to Android development and SQLite". Retrieved from https://stackoverflow.com/questions/tagged/android+sqlite

ВИСНОВОК

Розробка мобільного додатку "ShopHelper" для автоматизації процесу покупок у мережі магазинів з використанням технології штрих-кодів є важливим і актуальним завданням, яке відповідає потребам сучасних споживачів. Завдяки функціоналу сканування QR-кодів, автоматичному підрахунку суми покупок та зручному управлінню списком товарів, додаток забезпечує значні переваги для користувачів, спрощуючи процес вибору та купівлі товарів.

Під час розробки додатку було проведено детальний аналіз ринку мобільних додатків для покупок, що дозволило виявити ключові потреби цільової аудиторії та визначити основні функціональні вимоги до додатку. В результаті цього було розроблено інтуїтивно зрозумілий інтерфейс користувача (UX/UI) та архітектуру додатку, що забезпечує надійність і продуктивність системи.

Реалізація функціоналу додатку включала:

- Сканування QR-кодів для швидкого додавання товарів до корзини.
- Можливість ручного вводу штрих-кодів.
- Автоматичний підрахунок загальної суми покупок.
- Інтерфейс для перегляду та управління списком товарів.
- Інтеграцію з локальною базою даних для збереження та обробки інформації.

Проведене тестування, включаючи модульне, інтеграційне та функціональне тестування, підтвердило високу надійність та зручність використання додатку. Було враховано всі виявлені помилки та оптимізовано код для забезпечення найкращої продуктивності.

Майбутній розвиток додатку передбачає розширення бази даних товарів, інтеграцію з онлайн-магазинами та використання штучного інтелекту для персоналізації пропозицій, що підвищить цінність додатку для користувачів.

Загалом, мобільний додаток "ShopHelper" має значний потенціал для покращення досвіду покупок споживачів, забезпечуючи швидкість, зручність та надійність у використанні. Завдяки інноваційним технологіям і комплексному підходу до розробки, додаток здатен задовольнити потреби сучасних користувачів та стати важливим інструментом у їх повсякденному житті.

ДОДАТОК

MainActivity.kt

Головна сторінка додатку

Цей файл представляє головну активність додатку, яка відповідає за відображення кошика та взаємодію з користувачем.

Основні функції та елементи:

- Ініціалізація UI компонентів.
- Обробка подій користувача.
- Виклик функцій для управління кошиком.

Код

```
package com.example.barcodescannerapp

import android.content.Intent
import android.os.Bundle
import android.widget.Button
import android.widget.TextView
import android.widget.Toast
import androidx.activity.result.contract.ActivityResultContracts
import androidx.appcompat.app.AppCompatActivity
import androidx.appcompat.app.AppCompatDelegate
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.google.android.material.switchmaterial.SwitchMaterial
import com.example.barcodescannerapp.adapters.ItemAdapter
import com.example.barcodescannerapp.database.DatabaseHelper
import
com.example.barcodescannerapp.interfaces.ItemClickListener
import com.example.barcodescannerapp.models.Item
import android.app.Activity
import androidx.appcompat.app.AlertDialog

class MainActivity : AppCompatActivity(), ItemClickListener {
    private lateinit var itemsRecyclerView: RecyclerView
    private lateinit var addItemButton: Button
    private lateinit var themeSwitch: SwitchMaterial
    private lateinit var itemAdapter: ItemAdapter
    private var itemsList: MutableList<Item> = mutableListOf()
    private lateinit var dbHelper: DatabaseHelper
    private lateinit var totalPriceTextView: TextView
    private var totalCost: Double = 0.0
```

```
    private val scanBarcodeLauncher =
registerForActivityResult(ActivityResultContracts.StartActivityF
orResult()) { result ->
        if (result.resultCode == Activity.RESULT_OK &&
result.data != null) {
            val data = result.data!!
            val itemFound = data.getBooleanExtra("ITEM_FOUND",
false)
            if (itemFound) {
                val itemName = data.getStringExtra("ITEM_NAME")
?: return@registerForActivityResult
                val itemPrice =
data.getDoubleExtra("ITEM_PRICE", 0.0)
                val newItem = Item(name = itemName, barcode =
"", price = itemPrice) // The barcode is not passed back here
                addItemToList(newItem)
            } else {
                Toast.makeText(this, "Item not found.",
Toast.LENGTH_LONG).show()
            }
        }
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        dbHelper = DatabaseHelper(this)
        itemsRecyclerView = findViewById(R.id.rvItemsList)
        addItemButton = findViewById(R.id.btnAddItem)
        themeSwitch = findViewById(R.id.themeSwitch)
        totalPriceTextView = findViewById(R.id.tvTotalPrice)

        itemsRecyclerView.layoutManager =
LinearLayoutManager(this)
        itemAdapter = ItemAdapter(itemsList, this)
        itemsRecyclerView.adapter = itemAdapter

        addItemButton.setOnClickListener {
            val intent = Intent(this,
BarcodeScannerActivity::class.java)
            scanBarcodeLauncher.launch(intent)
        }

        themeSwitch.isChecked = isDarkThemeEnabled()
        themeSwitch.setOnCheckedChangeListener { _, isChecked ->
            setDarkThemeEnabled(isChecked)
        }
```

```
        updateTotalCostDisplay()
    }

    private fun addItemToList(item: Item) {
        val insertPosition = itemsList.size
        itemsList.add(item)
        totalCost += item.price
        itemAdapter.notifyItemInserted(insertPosition)
        updateTotalCostDisplay()
    }


    private fun updateTotalCostDisplay() {
        totalPriceTextView.text = String.format("Total: $%.2f",
totalCost)
    }


    private fun isDarkThemeEnabled(): Boolean {
        val sharedPrefs =
getSharedPreferences("AppSettingsPrefs", MODE_PRIVATE)
        return sharedPrefs.getBoolean("IsDarkThemeEnabled",
false)
    }

    private fun setDarkThemeEnabled(enabled: Boolean) {
        val sharedPrefs =
getSharedPreferences("AppSettingsPrefs", MODE_PRIVATE)
        with(sharedPrefs.edit()) {
            putBoolean("IsDarkThemeEnabled", enabled)
            apply()
        }
        AppCompatDelegate.setDefaultNightMode(if (enabled)
AppCompatDelegate.MODE_NIGHT_YES else
AppCompatDelegate.MODE_NIGHT_NO)
    }

    override fun onItemClick(item: Item) {
        AlertDialog.Builder(this)
            .setTitle("Delete Item")
            .setMessage("Do you want to delete this item from
the basket?")
            .setPositiveButton("Yes") { _, _ ->
                deleteItemFromBasket(item)
            }
            .setNegativeButton("No", null)
            .show()
    }
```

```
    private fun deleteItemFromBasket(item: Item) {
        val position = itemsList.indexOf(item)
        if (position != -1) {
            // Remove the item and notify the adapter
            itemsList.removeAt(position)
            itemAdapter.notifyItemRemoved(position)

            // Adjust the total cost and update display
            totalCost -= item.price
            updateTotalCostDisplay()
        } else {
            Toast.makeText(this, "Item not found in the
basket.", Toast.LENGTH_LONG).show()
        }
    }
}
```

BarcodeScannerActivity.kt

Сканер Штрих-кодів

Цей файл відповідає за реалізацію функціоналу сканування штрих-кодів для додавання товарів до кошика.

Основні функції та елементи:

- Ініціалізація камери та налаштування сканера.
- Обробка результатів сканування.
- Передача відсканованих даних до MainActivity для додавання товару в кошик.

Код

```
package com.example.barcodescannerapp

import android.app.Activity
import android.content.Intent
import android.os.Bundle
import android.text.InputType
import android.widget.Button
import android.widget.EditText
import android.widget.Toast
import androidx.appcompat.app.AlertDialog
import androidx.appcompat.app.AppCompatActivity
import com.example.barcodescannerapp.database.DatabaseHelper
import com.journeyapps.barcodescanner.CaptureManager
import com.journeyapps.barcodescanner.DecoratedBarcodeView
import android.content.pm.PackageManager

class BarcodeScannerActivity : AppCompatActivity() {

    private lateinit var capture: CaptureManager
    private lateinit var barcodeScannerView:
DecoratedBarcodeView
    private lateinit var manualEntryButton: Button
    private lateinit var dbHelper: DatabaseHelper  // Database
helper

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_barcode_scanner)

        barcodeScannerView =
findViewById(R.id.zxing_barcode_scanner)
        manualEntryButton =
findViewById(R.id.button_manual_entry)
        dbHelper = DatabaseHelper(this)  // Initialize the
database helper
```

```
        capture = CaptureManager(this, barcodeScannerView)
        capture.initializeFromIntent(intent, savedInstanceState)
        capture.decode()

        manualEntryButton.setOnClickListener {
            showManualEntryDialog()
        }
    }
    private fun initializeScanner(savedInstanceState: Bundle?) {
        // Initialize the CaptureManager with the
DecoratedBarcodeView and savedInstanceState
        capture = CaptureManager(this, barcodeScannerView)
        capture.initializeFromIntent(intent, savedInstanceState)
        capture.decode()
    }

    override fun onRequestPermissionsResult(requestCode: Int,
permissions: Array<out String>, grantResults: IntArray) {
        super.onRequestPermissionsResult(requestCode,
permissions, grantResults)
        if (requestCode == PERMISSION_REQUEST_CAMERA) {
            if (grantResults.isNotEmpty() && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
                // Permission was granted. Do the camera
initialization.
                initializeScanner(null)
            } else {
                // Permission was denied.
                Toast.makeText(this, "Camera permission is
required to scan barcodes", Toast.LENGTH_SHORT).show()
                finish()
            }
        }
    }


    override fun onResume() {
        super.onResume()
        capture.onResume()
    }

    override fun onPause() {
        super.onPause()
        capture.onPause()
    }

    override fun onDestroy() {
        super.onDestroy()
```

```
        capture.onDestroy()
    }

    override fun onSaveInstanceState(outState: Bundle) {
        super.onSaveInstanceState(outState)
        capture.onSaveInstanceState(outState)
    }

    companion object {
        private const val PERMISSION_REQUEST_CAMERA = 1
    }

    //Manual code entry
    private fun showManualEntryDialog() {
        val editText = EditText(this).apply {
            inputType = InputType.TYPE_CLASS_NUMBER
            hint = "Enter barcode"
        }

        AlertDialog.Builder(this)
            .setTitle("Manual Barcode Entry")
            .setView(editText)
            .setPositiveButton("OK") { _, _ ->
                val barcode = editText.text.toString()
                handleManualBarcode(barcode)
            }
            .setNegativeButton("Cancel", null)
            .show()
    }

    private fun handleManualBarcode(barcode: String) {
        val item = dbHelper.getItemByBarcode(barcode)
        if (item != null) {
            val returnIntent = Intent()
            returnIntent.putExtra("ITEM_FOUND", true)
            returnIntent.putExtra("ITEM_NAME", item.name)
            returnIntent.putExtra("ITEM_PRICE", item.price)
            setResult(Activity.RESULT_OK, returnIntent)
            finish()
        } else {
            Toast.makeText(this, "Item not found.",
Toast.LENGTH_LONG).show()
            finish()
        }
    }
}
```

ItemAdapter.kt

Адаптер для відображення товарів

Цей файл містить адаптер, який використовується для відображення списку товарів в RecyclerView.

Основні функції та елементи:

- Налаштування ViewHolder для відображення окремих товарів.
- Зв'язування даних товарів з елементами інтерфейсу.
- Обробка подій кліків на товарах.

Код

```
package com.example.barcodescannerapp.adapters

import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.TextView
import androidx.recyclerview.widget.RecyclerView
import com.example.barcodescannerapp.R
import com.example.barcodescannerapp.models.Item
import
com.example.barcodescannerapp.interfaces.ItemClickListener

class ItemAdapter(
    private val items: List<Item>,
    private val listener: ItemClickListener
) : RecyclerView.Adapter<ItemAdapter.ViewHolder>() {

    override fun onCreateViewHolder(parent: ViewGroup, viewType:
Int): ViewHolder {
        val view = LayoutInflater.from(parent.context)
            .inflate(R.layout.item, parent, false)
        return ViewHolder(view)
    }

    override fun onBindViewHolder(holder: ViewHolder, position:
Int) {
        val item = items[position]
        holder.itemName.text = item.name
        holder.itemPrice.text = item.price.toString()
        holder.itemView.setOnClickListener {
listener.onItemClick(item) }
    }

    override fun getItemCount(): Int = items.size
```

```
    class ViewHolder(itemView: View) :
RecyclerView.ViewHolder(itemView) {
        val itemName: TextView =
itemView.findViewById(R.id.itemNameTextView)
        val itemPrice: TextView =
itemView.findViewById(R.id.itemPriceTextView)
    }
}
```

ItemClickListener.kt

Слухач кліків на товарах

Цей файл реалізує інтерфейс для обробки подій кліків на товарах в списку.

Основні функції та елементи:

- Оголошення методів для обробки кліків.
- Використовується в ItemAdapter для визначення дій при натисканні на товар.

Код

```
package com.example.barcodescannerapp.interfaces

import com.example.barcodescannerapp.models.Item

interface ItemClickListener {
    fun onItemClick(item: Item)
}
```

DatabaseHelper.kt

Допоміжний клас для роботи з базою даних

Цей файл містить допоміжні функції для взаємодії з базою даних, включаючи створення, читання, оновлення та видалення записів.

Основні функції та елементи:

- Створення та налаштування бази даних.
- Операції CRUD (створення, читання, оновлення, видалення).
- Метод для отримання списку товарів з бази даних.

Код

```
package com.example.barcodescannerapp.database

import android.content.ContentValues
import android.content.Context
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper
import com.example.barcodescannerapp.models.Item

class DatabaseHelper(context: Context) :
SQLiteOpenHelper(context, DATABASE_NAME, null, DATABASE_VERSION)
{

    override fun onCreate(db: SQLiteDatabase) {
        val createTableStatement = """
            CREATE TABLE $TABLE_NAME (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                name TEXT NOT NULL,
                barcode TEXT NOT NULL UNIQUE,
                price REAL NOT NULL
            );
        """.trimIndent()

        db.execSQL(createTableStatement)

        // Insert pre-defined items
        prePopulateItems(db)
    }

    override fun onUpgrade(db: SQLiteDatabase, oldVersion: Int,
newVersion: Int) {
        db.execSQL("DROP TABLE IF EXISTS $TABLE_NAME")
        onCreate(db)
    }

    private fun prePopulateItems(db: SQLiteDatabase) {
        val items = listOf(
```

```
            Item(name = "Bread", barcode = "1111111111", price =
1.99),
            Item(name = "Milk", barcode = "22222222", price =
1.49),
            Item(name = "Butter", barcode = "3333333333", price
= 0.29),
            Item(name = "Beacon", barcode = "4444444444", price
= 1.99),
            Item(name = "Cheese", barcode = "5555555555", price
= 1.29),
            Item(name = "Egg", barcode = "6666666666", price =
2.49),
            Item(name = "Potato", barcode = "7777777777", price
= 2.69),
            Item(name = "Donut", barcode = "8888888888", price =
2.29),
            Item(name = "Roll", barcode = "9999999999", price =
1.99),
            Item(name = "Tomato", barcode = "1212121212", price
= 1.29),
            Item(name = "Cucumber", barcode = "1313131313",
price = 2.09),
            Item(name = "Chips", barcode = "1414141414", price =
1.29),
            Item(name = "Cola", barcode = "1616161616", price =
1.99),
            Item(name = "Sprite", barcode = "1717171717", price
= 2.99)
        )

        for (item in items) {
            val cv = ContentValues().apply {
                put(COL_NAME, item.name)
                put(COL_BARCODE, item.barcode)
                put(COL_PRICE, item.price)
            }
            db.insert(TABLE_NAME, null, cv)
        }
    }

    // Existing methods (addItem, getItemByBarcode)...
    fun addItem(item: Item): Boolean {
        val db = this.writableDatabase
        val cv = ContentValues()
        cv.put(COL_NAME, item.name)
        cv.put(COL_BARCODE, item.barcode)
        cv.put(COL_PRICE, item.price)
        val insertResult = db.insert(TABLE_NAME, null, cv)
        db.close()
```

```
        return insertResult != -1L
    }

    fun getItemByBarcode(barcodeParam: String): Item? {
        val db = this.readableDatabase
        val cursor = db.query(
            TABLE_NAME,
            arrayOf(COL_ID, COL_NAME, COL_BARCODE, COL_PRICE),
            "$COL_BARCODE = ?",
            arrayOf(barcodeParam),
            null, null, null
        )
        var item: Item? = null
        if (cursor.moveToFirst()) {
            val id =
cursor.getInt(cursor.getColumnIndexOrThrow(COL_ID))
            val name =
cursor.getString(cursor.getColumnIndexOrThrow(COL_NAME))
            val barcode =
cursor.getString(cursor.getColumnIndexOrThrow(COL_BARCODE))
            val price =
cursor.getDouble(cursor.getColumnIndexOrThrow(COL_PRICE))
            item = Item(id, name, barcode, price)
        }
        cursor.close()
        db.close()
        return item
    }

    companion object {
        private const val DATABASE_NAME = "grocery.db"
        private const val TABLE_NAME = "grocery_items"
        private const val DATABASE_VERSION = 1

        private const val COL_ID = "id"
        private const val COL_NAME = "name"
        private const val COL_BARCODE = "barcode"
        private const val COL_PRICE = "price"
    }
}
```

Color.kt

Кольорова схема

Цей файл містить визначення кольорів, які використовуються в додатку.

Основні функції та елементи:

- Оголошення кольорових значень для теми додатку.
- Визначення кольорів для різних елементів інтерфейсу.

Код

```
package com.example.barcodescannerapp.ui.theme

import androidx.compose.ui.graphics.Color

val Purple80 = Color(0xFFD0BCFF)
val PurpleGrey80 = Color(0xFFCCC2DC)
val Pink80 = Color(0xFFEFB8C8)

val Purple40 = Color(0xFF6650a4)
val PurpleGrey40 = Color(0xFF625b71)
val Pink40 = Color(0xFF7D5260)
```

Theme.kt

Тема додатку

Цей файл відповідає за налаштування загального вигляду додатку, включаючи кольори, шрифти та стилі.

Основні функції та елементи:

- Налаштування теми додатку.
- Визначення стилів для різних UI компонентів.

Код

```
package com.example.barcodescannerapp.ui.theme

import android.app.Activity
import android.os.Build
import androidx.compose.foundation.isSystemInDarkTheme
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.darkColorScheme
import androidx.compose.material3.dynamicDarkColorScheme
import androidx.compose.material3.dynamicLightColorScheme
import androidx.compose.material3.lightColorScheme
import androidx.compose.runtime.Composable
import androidx.compose.runtime.SideEffect
import androidx.compose.ui.graphics.toArgb
```

```
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.platform.LocalView
import androidx.core.view.WindowCompat

private val DarkColorScheme = darkColorScheme(
    primary = Purple80,
    secondary = PurpleGrey80,
    tertiary = Pink80
)

private val LightColorScheme = lightColorScheme(
    primary = Purple40,
    secondary = PurpleGrey40,
    tertiary = Pink40

    /* Other default colors to override
    background = Color(0xFFFFFBFE),
    surface = Color(0xFFFFFBFE),
    onPrimary = Color.White,
    onSecondary = Color.White,
    onTertiary = Color.White,
    onBackground = Color(0xFF1C1B1F),
    onSurface = Color(0xFF1C1B1F),
    */
)

@Composable
fun BarcodeScannerAppTheme(
    darkTheme: Boolean = isSystemInDarkTheme(),
    // Dynamic color is available on Android 12+
    dynamicColor: Boolean = true,
    content: @Composable () -> Unit
) {
    val colorScheme = when {
        dynamicColor && Build.VERSION.SDK_INT >=
Build.VERSION_CODES.S -> {
            val context = LocalContext.current
            if (darkTheme) dynamicDarkColorScheme(context) else
dynamicLightColorScheme(context)
        }

        darkTheme -> DarkColorScheme
        else -> LightColorScheme
    }
    val view = LocalView.current
    if (!view.isInEditMode) {
        SideEffect {
            val window = (view.context as Activity).window
            window.statusBarColor = colorScheme.primary.toArgb()
```

```
            WindowCompat.getInsetsController(window,
view).isAppearanceLightStatusBars = darkTheme
        }
    }

    MaterialTheme(
        colorScheme = colorScheme,
        typography = Typography,
        content = content
    )
}
```

Type.kt

Типи даних

Цей файл містить визначення різних типів даних, які використовуються в додатку.

Основні функції та елементи:

- Оголошення enum класів для різних категорій товарів.
- Визначення типів даних для відображення та збереження інформації.

Код

```
package com.example.barcodescannerapp.ui.theme

import androidx.compose.material3.Typography
import androidx.compose.ui.text.TextStyle
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.sp

// Set of Material typography styles to start with
val Typography = Typography(
    bodyLarge = TextStyle(
        fontFamily = FontFamily.Default,
        fontWeight = FontWeight.Normal,
        fontSize = 16.sp,
        lineHeight = 24.sp,
        letterSpacing = 0.5.sp
    )
    /* Other default text styles to override
    titleLarge = TextStyle(
        fontFamily = FontFamily.Default,
        fontWeight = FontWeight.Normal,
        fontSize = 22.sp,
        lineHeight = 28.sp,
        letterSpacing = 0.sp
```

```
    ),
    labelSmall = TextStyle(
        fontFamily = FontFamily.Default,
        fontWeight = FontWeight.Medium,
        fontSize = 11.sp,
        lineHeight = 16.sp,
        letterSpacing = 0.5.sp
    )
    */
)
```

Build.gradle.kts

Конфігураційний файл проєкту

Цей файл містить налаштування та залежності для вашого Android проєкту.

Основні функції та елементи:

Плагіни: додаються плагіни для Android і Kotlin.

Конфігурації Android: namespace, compileSdk, defaultConfig, buildTypes, compileOptions, kotlinOptions, buildFeatures, composeOptions, packaging.

DefaultConfig: базові налаштування додатку (applicationId, minSdk, targetSdk, versionCode, versionName).

BuildTypes: конфігурації для різних типів збірок (release).

CompileOptions і KotlinOptions: налаштування сумісності з версіями Java та Kotlin.

BuildFeatures і ComposeOptions: включення та налаштування Jetpack Compose.

Packaging: винятки для ресурсів.

Dependencies: перелік залежностей проєкту, включаючи бібліотеки для AndroidX, Jetpack Compose, ZXing для сканування штрих-кодів, Google ML Kit для сканування штрих-кодів, матеріальні компоненти, ConstraintLayout та інші.

Залежності для тестування: junit, androidTest, debug.

Код

```
plugins {
    id("com.android.application")
    id("org.jetbrains.kotlin.android")
}

android {
    namespace = "com.example.barcodescannerapp"
    compileSdk = 34
```

```
    defaultConfig {
        applicationId = "com.example.barcodescannerapp"
        minSdk = 24
        targetSdk = 34
        versionCode = 1
        versionName = "1.0"

        testInstrumentationRunner =
"androidx.test.runner.AndroidJUnitRunner"
        vectorDrawables {
            useSupportLibrary = true
        }
    }

    buildTypes {
        release {
            isMinifyEnabled = false
            proguardFiles(

getDefaultProguardFile("proguard-android-optimize.txt"),
                "proguard-rules.pro"
            )
        }
    }
    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_1_8
        targetCompatibility = JavaVersion.VERSION_1_8
    }
    kotlinOptions {
        jvmTarget = "1.8"
    }
    buildFeatures {
        compose = true
    }
    composeOptions {
        kotlinCompilerExtensionVersion = "1.5.1"
    }
    packaging {
        resources {
            excludes += "/META-INF/{AL2.0,LGPL2.1}"
        }
    }
}

dependencies {

    implementation ("androidx.appcompat:appcompat:1.6.1")
    implementation ("androidx.compose.ui:ui:1.6.3")
```

```
    implementation
("com.journeyapps:zxing-android-embedded:4.1.0")
    implementation ("androidx.compose.material:material:1.6.3")
    implementation ("com.google.mlkit:barcode-scanning:17.2.0")
    implementation
("androidx.compose.ui:ui-tooling-preview:1.6.3")
    implementation("androidx.core:core-ktx:1.12.0")

implementation("androidx.lifecycle:lifecycle-runtime-ktx:2.7.0")
    implementation("androidx.activity:activity-compose:1.8.2")

implementation(platform("androidx.compose:compose-bom:2023.08.00
"))
    implementation("androidx.compose.ui:ui")
    implementation("androidx.compose.ui:ui-graphics")
    implementation("androidx.compose.ui:ui-tooling-preview")
    implementation("androidx.compose.material3:material3")
    implementation("androidx.appcompat:appcompat:1.6.1")

implementation("com.google.android.material:material:1.11.0")

implementation("androidx.constraintlayout:constraintlayout:2.1.4
")
    testImplementation("junit:junit:4.13.2")
    androidTestImplementation("androidx.test.ext:junit:1.1.5")

androidTestImplementation("androidx.test.espresso:espresso-core:
3.5.1")

androidTestImplementation(platform("androidx.compose:compose-bom
:2023.08.00"))

androidTestImplementation("androidx.compose.ui:ui-test-junit4")
    debugImplementation("androidx.compose.ui:ui-tooling")
    debugImplementation("androidx.compose.ui:ui-test-manifest")
}
```