Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська дипломна робота

на тему: «Розробка програмного засобу для аналізу складу харчових продуктів»

Виконав: студент IV курсу, групи ЗПІ-20б

спеціальності

121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)

Щербань Костянтин Олександрович
(прізвище та ініціали)

Керівник: д.т.н., професор Ліщинська Л.Б.
(прізвище та ініціали)

Рецензент: професор Захарченко С.М.
(прізвище та ініціали)

Допущено до захисту
Зав. кафедри Романюк О. Н.
«10» червня 2024 р.

ВНТУ – 2024

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший (бакалаврський)
Галузь знань 12 – «Інформаційні технології»
Спеціальність 121 – «Інженерія програмного забезпечення»
Освітньо-професійна програма – «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
д.т.н., професор
_____О. Н. Романюк
«12» березня 2024 р.

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Щербаню Костянтину Олександровичу

Тема роботи – «Розробка програмного засобу для аналізу складу харчових продуктів»

Керівник роботи: Ліщинська Людмила Броніславівна, д.т.н., професор, затверджені наказом вищого навчального закладу від 11 березня 2024 р. № 80.

2. Строк подання студентом роботи 3 червня 2024 р.

3. Вихідні дані до роботи: база даних харчових продуктів; харчова цінність продуктів; інформація про склад; програмні інструменти для аналізу зображень; алгоритми розпізнавання штрих-коду;

4. Зміст розрахунково-пояснювальної записки: аналіз стану мобільних систем з використанням засобів аналізу складу харчових продуктів; порівняльний аналіз аналогів; аналіз методів розв'язання поставленої задачі; постановка задачі розробки програмного забезпечення розробка моделі системи; розробка алгоритмів роботи системи; розробка структури проєкту; варіантний аналіз і обгрунтування вибору мови програмування, ; вибір середовища розробки; проектування бази даних для зберігання інформації про харчові продукти та їх склад; розробка модуля сканеру для зчитування штрих-кодів;

розробка архітектури програмного забезпечення; аналіз методів тестування програмного забезпечення; тестування розробленого програмного продукту; розробка інструкції користувача; висновки; список використаних джерел; додатки.

5. Перелік ілюстративного матеріалу: аналоги програмного застосунку; структура проєкту; база даних застосунку; модуль сканера штрих-кодів; інтерфейс застосунку; тестування розробленого програмного продукту.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1-4	Ліщинська Л.Б., д.т.н., професор	13.03.24	03.06.24

7. Дата видачі завдання 13 березня 2024 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
1	Обґрунтування вибору методу розробки та постановка задачі дослідження	14.03.24 – 22.03.24	Виконано
2	Проектування бази даних	14.03.24 – 22.03.24	Виконано
3	Розробка модуля сканеру штрих-кодів	25.03.24 – 19.04.24	Виконано
4	Розробка застосунку	22.04.24 – 10.05.24	Виконано
5	Тестування застосунку	13.05.24 – 24.05.24	Виконано
6	Оформлення матеріалів до захисту БДР	27.05.24 – 30.05.24	Виконано

Студент _____ **К.О. Щербань**
(підпис) (ініціали та прізвище)

Керівник бакалаврської дипломної роботи _____ **Л.Б. Ліщинська**
(підпис) (ініціали та прізвище)

АНОТАЦІЯ

УДК 00492

Щербань К.О. Розробка програмного засобу для аналізу складу харчових продуктів : бакалаврська дипломна робота зі спеціальності 121 – інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2024. 88 с.

На укр. мові. Бібліогр. : 24 назв ; рис. : 22; табл. 6.

У бакалаврській дипломній роботі проведено детальний аналіз стану питання розробки застосунку. Було проведено аналіз аналогів, визначено їх переваги і недоліки та доведено доцільність розробки власного мобільного застосунку.

Було обґрунтовано вибір мови програмування Java та середовища розробки Android Studio, а також технологій, які були використані для розробки застосунку.

Розроблено модуль сканеру для зчитування штрих-кодів для аналізу складу в харчових продуктах, розроблено модель, структуру та алгоритми роботи системи. Було аналізовано методи тестування програмного забезпечення та проведено тестування розробленого програмного продукту.

Розроблено програмний засіб для аналізу складу харчових продуктів.

Отримані в бакалаврській дипломній роботі результати можна використати для аналізу складу харчових продуктів, що дозволить визначити їх поживну цінність, виявити можливі алергени та забезпечити відповідність стандартам харчової безпеки.

Ключові слова: програмне забезпечення, харчові продукти, аналіз складу, здорове харчування.

ABSTRACT

UDC 00492

Shcherban K.O. Development of a software tool for analyzing the composition of food products : bachelor's thesis in specialty 121 - Software Engineering, educational program - Software Engineering. Vinnytsia: VNTU, 2024. 88 c.

In Ukrainian. Bibliography: 24 titles; Figures: 22; Table 6.

A detailed analysis of the state of application development was carried out in the bachelor thesis. An analysis of analogues was carried out, their advantages and disadvantages were determined, and the feasibility of developing one's own mobile application was proven.

The choice of the Java programming language and the Android Studio development environment, as well as the technologies that were used to develop the application, were justified.

A scanner module was developed for reading barcodes for analyzing the composition of food products, the model, structure and algorithms of the system were developed. Software testing methods were analyzed and the developed software product was tested.

A software tool for analyzing the composition of food products has been developed.

The results obtained in the bachelor thesis can be used to analyze the composition of food products, which will allow to determine their nutritional value, identify possible allergens and ensure compliance with food safety standards.

Key words: software, food products, composition analysis, healthy nutrition.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Харчування є фундаментальною складовою здоров'я та благополуччя кожної людини. Не лише частота та кількість споживаних продуктів мають значення, але й якість та склад їжі, яку ми вживаємо щоденно [1]. У зв'язку зі зростаючою увагою до здорового харчування, існує велика потреба у інструментах, які дозволяють споживачам з легкістю розуміти та контролювати якість харчових продуктів.

Якість харчових продуктів є однією з ключових проблем сучасного суспільства. Зростаюча глобалізація, індустріалізація виробництва та постійне розширення ринків збуту призводять до зниження контролю над якістю продукції [2]. Основні проблеми, пов'язані з якістю харчових продуктів, включають:

1. Наявність шкідливих речовин: Пестициди, гербіциди, антибіотики та інші хімічні речовини можуть потрапляти в їжу під час її виробництва.
2. Недостатність поживних речовин: Інтенсивне сільське господарство може виснажувати ґрунти, що призводить до зниження вмісту вітамінів і мінералів у продуктах.
3. Неправильне маркування: Нечесні виробники можуть приховувати або неправильно вказувати склад продуктів, що може мати серйозні наслідки для споживачів, особливо для тих, хто має алергії чи специфічні дієтичні потреби.
4. Контамінація: Продукти можуть бути заражені бактеріями, вірусами або іншими патогенами під час виробництва, зберігання або транспортування.
5. Фальсифікація продуктів: Використання дешевих замінників і підробка продуктів харчування можуть суттєво вплинути на їх якість та безпеку.

Програми для аналізу складу харчових продуктів можуть сприяти підвищенню якості харчових продуктів. Вони дозволять користувачам отримувати точну інформацію про склад продуктів, включаючи вміст

поживних речовин, вітамінів і мінералів, що допоможе приймати обґрунтовані рішення щодо свого харчування. Автоматично визначати та виділяти потенційні алергени у складі продуктів допоможе споживачам з алергіями уникати небезпечних для них продуктів. Автоматичне визначення та виділення потенційних алергенів у складі продуктів допоможе споживачам з алергіями уникати небезпечних для них продуктів.

У зв'язку з чим програмний засіб для аналізу складу харчових продуктів зможе частково вирішити проблему якості харчових продуктів, забезпечуючи споживачів точною та надійною інформацією, що сприятиме зниженню ризиків для їхнього здоров'я та покращенню загальної якості харчування [3].

Існує декілька програмних продуктів для аналізу складу харчових продуктів, серед яких найпопулярніші: Cronometer, FoodData Central від USDA та Nutritional Analysis Tool.

Cronometer - Програма з високою точністю даних та великим обсягом інформації про вітаміни і мінерали. Однак, її інтерфейс може здатися складним для новачків, а деякі функції доступні лише в платній версії.

Ця база даних містить офіційні дані про харчові продукти, що забезпечує високу точність. Однак, її використання може бути менш зручним через відсутність мобільного додатку та менш інтуїтивний інтерфейс.

Nutritional Analysis Tool (NAT) - Надає детальну інформацію про харчову цінність продуктів. Проте, вона є менш відомою і має менш привабливий інтерфейс порівняно з комерційними програмами.

Отже, основними проблемами існуючих програм є або недостатня точність даних через користувацький внесок, або незручний інтерфейс. Це відкриває можливості для створення нового програмного забезпечення, яке поєднувало б високу точність офіційних даних із зручним та інтуїтивним інтерфейсом.

Зв’язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою роботи є покращення процесу аналізу складу харчових продуктів шляхом розробки мобільного застосунку.

Відповідно до поставленої мети необхідно виконати такі завдання:

– Провести огляд та аналіз існуючих програмних засобів для аналізу складу харчових продуктів, визначити їх сильні та слабкі сторони.

– Визначити основні функціональні вимоги до застосунку.

– Розробити та наповнити базу даних харчових продуктів, яка включатиме інформацію про їх поживну цінність, склад, можливі алергени та інші важливі характеристики.

– Розробити модуль сканера штрих-коду.

– Створити зручний та інтуїтивно зрозумілий інтерфейс користувача, який забезпечить легкий доступ.

Об’єкт дослідження - це процес аналізу складу харчових продуктів для визначення їх поживної цінності.

Предмет дослідження - це методи і засоби аналізу складу харчових продуктів, для визначення їх поживної цінності.

Методи дослідження. У процесі досліджень використовувались такі методи:

- методи аналізу складу харчових продуктів, які використовуються для обробки та аналізу даних про харчові продукти;
- Методи тестування програмного застосунка, які включають юніт-тестування, інтеграційне тестування, функціональне тестування, регресійне тестування, системне тестування та прийомне тестування, що забезпечують всебічну перевірку коректності та надійності програмного забезпечення.
- UX та UI методи розробки інтерфейсу користувача.

Новизна отриманих результатів. Удосконалено метод аналізу складу харчових продуктів, який покращений шляхом оптимізації швидкодії бази даних та застосувані нових алгоритмів розпізнавання штрих-кодів, що дає можливість забезпечити швидший доступ до даних та покращення в

продуктивності програми, що призвело до підвищення ефективності та точності аналізу складу продуктів.

Практична цінність отриманих результатів. Розроблений програмний засіб дозволяє користувачам швидко й точно визначати склад харчових продуктів, що допоможе їм робити свідомі вибори щодо свого харчування. Це може сприяти зниженню ризиків розвитку хронічних захворювань, таких як ожиріння, діабет та серцево-судинні захворювання.

Особистий внесок здобувача. Розроблено модуль сканеру для зчитування штрих-кодів, спроектована база даних для зберігання інформації про харчові продукти та їх склад, удосконалено метод аналізу складу харчових продуктів.

Апробація матеріалів бакалаврської кваліфікаційної роботи. Основні положення бакалаврської дипломної роботи доповідалися та обговорювалися на міжнародній студентській конференції - VI Міжнародна студентська конференція «Глобалізація наукових знань: міжнародна співпраця та інтеграція галузей наук» (м. Біла Церква, Україна, 2024)

Публікації. Основні результати досліджень опубліковано у матеріалах конференції [3].

1 ОБҐРУНТУВАННЯ ВИБОРУ МЕТОДУ РОЗРОБКИ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Аналіз стану мобільних систем з використанням засобів аналізу складу харчових продуктів

Завдяки стрімкому розвитку технологій, мобільні додатки, спрямовані на аналіз складу харчових продуктів, стають дедалі популярнішими серед споживачів, які прагнуть до здорового харчування. Розробка програмного засобу для таких систем вимагає високої точності аналізу та інтуїтивно зрозумілого інтерфейсу, щоб користувачі могли надійно дізнаватися про поживні складові та калорійність споживаних продуктів. Важливо, що правильне харчування не обмежується лише знанням калорійності, але й залучає глибший аналіз балансу білків, жирів і вуглеводів, а також мікроелементів і вітамінів, які є необхідними для підтримки оптимального здоров'я.

Однією з важливих переваг сучасних мобільних систем аналізу харчових продуктів є їхня здатність миттєво надавати інформацію про наявність алергенів, поживну цінність, а також деталі про походження і методи виробництва продуктів. Це значно підвищує рівень обізнаності споживачів і сприяє ухваленню обдуманих рішень щодо їхнього харчування. Розуміння того, як компоненти харчування взаємодіють один з одним і впливають на організм, дозволяє споживачам не просто дотримуватися дієти, а формувати здорові харчові звички.

Для досягнення цих цілей розробники повинні взаємодіяти з дієтологами, алергологами та іншими фахівцями у галузі харчування, аби забезпечити високу точність і релевантність наданої інформації. Це включає в себе створення бази даних, яка регулярно оновлюється і містить актуальні дані про широкий асортимент харчових продуктів і їх компоненти. Запровадження інтерактивних елементів, таких як персоналізовані поради щодо харчування та інтеграція з планами харчування, може додатково підвищити корисність

додатку.

Розробка такого програмного засобу може також передбачати інтеграцію з іншими мобільними додатками та сервісами, наприклад, з додатками для моніторингу фізичної активності та загального стану здоров'я, забезпечуючи користувачам комплексний підхід до управління своїм здоров'ям.

Таким чином, створення мобільного додатку для аналізу складу харчових продуктів відкриває широкі перспективи для покращення дієтичних звичок та підвищення здоров'я споживачів. У зв'язку з цим, необхідно стежити за останніми технологічними трендами та інноваціями у галузі харчових технологій та мобільного програмування, щоб забезпечити ефективність та актуальність розроблюваного програмного продукту.

Отже, розробка та впровадження мобільних систем для аналізу складу харчових продуктів є не лише важливим кроком до підвищення обізнаності споживачів про їхнє харчування, але й сприяє формуванню здорових харчових звичок, що в кінцевому результаті веде до покращення загального рівня здоров'я населення.

1.2 Порівняльний аналіз аналогів

На ринку існує чимало додатків, які пропонують аналіз складу харчових продуктів. Ці додатки використовуються для допомоги споживачам у виборі продуктів, враховуючи їх дієтичні обмеження, алергії, харчові нетерпимості, особисті уподобання або навіть цілі щодо схуднення. Вони надають інструменти, які дозволяють користувачам отримати більше контролю над своїм харчуванням, підвищуючи обізнаність щодо вживаних продуктів.

Деякі з найбільш популярних і широко використовуваних додатків включають:

- MyFitnessPal;
- Yuka;
- Fooducate;
- AllergyEats;

- ShopWell.

Розглянемо кожен додаток окремо:

MyFitnessPal [4] - це один із найпопулярніших додатків для відстеження харчування і фітнесу, який дозволяє користувачам відслідковувати їх споживання калорій та фізичну активність для досягнення особистих цілей здоров'я і фітнесу(рис 1.1).

Основні можливості MyFitnessPal:

Велика база даних продуктів: MyFitnessPal має одну з найбільших доступних баз даних продуктів харчування.

Відстеження калорій та макронутрієнтів: Користувачі можуть легко вводити щоденне споживання їжі та отримувати детальну інформацію про калорії, білки, жири та вуглеводи.

Щоденник харчування та вправ: Дозволяє користувачам зберігати історію їх харчування та тренувань, що є корисним інструментом для аналізу звичок і прогресу.

Плюси MyFitnessPal:

- Широкий вибір продуктів: Завдяки великій базі даних, користувачі можуть знайти майже будь-який продукт, який вони споживають, що робить додаток дуже зручним для використання.
- Корисна спільнота: Користувачі можуть спілкуватися з іншими, отримувати підтримку та поради, що значно збільшує мотивацію.

Мінуси MyFitnessPal:

- Реклама та преміум підписка: Базова версія містить рекламу, і багато корисних функцій доступні тільки в преміум версії, що може бути незручно для користувачів.
- Точність даних: Хоча база даних є великою, іноді зустрічаються помилки у даних про продукти, додані користувачами.
- Фокус на калоріях: Додаток сильно зосереджений на калорійності, що може не підходити для людей, які шукають більш глибокий аналіз свого харчування або мають певні медичні стани.

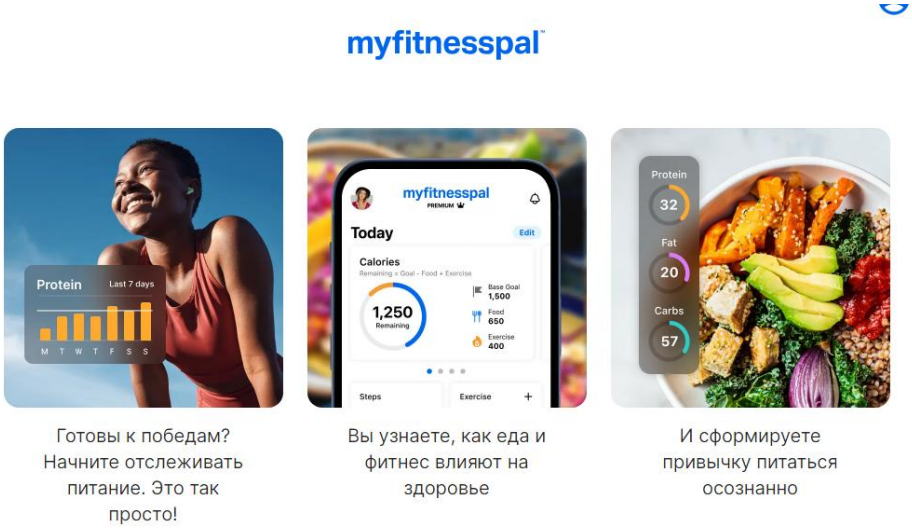

Рисунок 1.1 – Приклад додатку MyFitnessPal

Yuka [5] — це мобільний додаток, який дозволяє сканувати штрих-коди продуктів харчування та косметичних засобів для оцінки їх впливу на здоров'я. Додаток аналізує склад продуктів і надає користувачам інформацію про їхні потенційні ризики та переваги (рис 1.2).

Основні можливості Yuka:

Оцінка продуктів: Кожен продукт отримує оцінку від 0 до 100 на основі якості його інгредієнтів. Високі бали вказують на кращу якість продукту для здоров'я.

Інтеграція з магазинами: Користувачі можуть використовувати Yuka під час покупок, скануючи продукти прямо в магазинах.

Плюси Yuka:

- Легкість використання: Просто скануйте штрих-код, і додаток надає детальну інформацію миттєво.
- Підвищення обізнаності споживачів: Заохочує користувачів робити більш обдумані вибори стосовно продуктів, які вони споживають.

Мінуси Yuka:

- Обмежена база даних: Не всі продукти доступні для сканування, особливо нові або менш популярні бренди.

- Можливі неточності: Оцінки основані на загальнодоступних дослідженнях і можуть не враховувати недавні наукові відкриття або специфічні контексти.
- Зосередженість на негативних аспектах: Іноді занадто великий акцент на потенційній шкоді може вводити користувачів в оману щодо реальних ризиків певних продуктів.

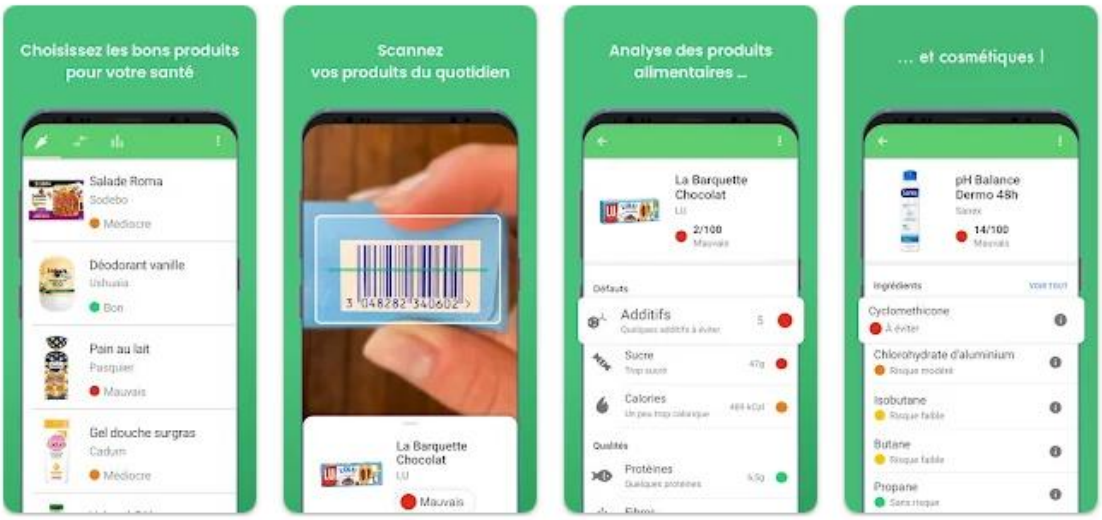

Рисунок 1.2 – Приклад роботи додатку Yuka

Fooducate [6] — це мобільний додаток, розроблений для того, щоб допомогти користувачам робити більш обізнані вибори в їжі та продуктах харчування, враховуючи їхні індивідуальні здоров'я та дієтичні потреби. Додаток аналізує продукти на основі їхнього складу, поживної цінності, наявності добавок і інших параметрів, що впливають на здоров'я (рис 1.3).

Плюси Fooducate:

- Освітній компонент: Надає корисну інформацію про інгредієнти, що сприяє більшій обізнаності споживачів.
- Персоналізація: Програма адаптується під індивідуальні потреби, дозволяючи користувачам досягти своїх цілей здоров'я та харчування.
- Залучення спільноти: Можливість спілкування з іншими користувачами стимулює здорову конкуренцію та взаємодопомогу.

Мінуси Fooducate:

- Потенційні неточності: Оскільки оцінки засновані на загальних даних про інгредієнти, можуть бути випадки, коли індивідуальні реакції на продукт відрізняються.
- Обмежена база даних: Не всі продукти представлені у додатку, що може обмежувати його корисність для деяких користувачів, особливо в невеликих містах чи країнах.

Рисунок 1.3 – Приклад роботи додатку Fooducate

AllergyEats [7] — це мобільний додаток і вебсайт, який допомагає людям з харчовими алергіями та іншими дієтичними обмеженнями знаходити ресторани, де можна безпечно їсти. Це корисний ресурс, який оцінює ресторани на основі відгуків користувачів про їх здатність задовольняти потреби людей з алергіями (рис 1.4).

Плюси AllergyEats:

- Спеціалізований фокус: Цільовий фокус на потреби людей з харчовими алергіями робить його особливо цінним ресурсом для цієї аудиторії.

- Спільнота: Створюється сильне почуття спільноти, де користувачі підтримують один одного, діляться порадами та рекомендаціями.
- Освітній ресурс: Поради та блоги допомагають користувачам краще розуміти, як обирати безпечні місця для їжі поза домом.

Мінуси AllergyEats:

- Обмеженість вибору: У деяких регіонах може бути недостатньо відгуків або інформації про ресторани, що обмежує корисність додатку.
- Залежність від користувацьких відгуків: Якість та точність інформації сильно залежать від активності та відвертості спільноти.
- Відсутність гарантій: Попри корисність відгуків, немає гарантій, що досвід одного користувача буде повторюваний іншими, особливо в плані обслуговування алергій.

Рисунок 1.4 – Приклад роботи вебсайту AllergyEats

ShopWell [8] — це мобільний додаток, який допомагає користувачам робити інформований вибір продуктів харчування, заснований на їхніх особистих дієтичних потребах і здоров'ї. Додаток сканує штрих-коди продуктів та надає детальну інформацію про їхній склад, а також оцінює їх відповідність заздалегідь заданим дієтичним вимогам користувача(рис 1.5).

Плюси ShopWell:

- Інтуїтивно зрозумілий інтерфейс: Простота використання і доступність інформації сприяють зручності користувачів.
- Освітній компонент: Надає корисну інформацію про харчові продукти, допомагаючи користувачам розуміти, як продукти впливають на їхнє здоров'я.

Мінуси ShopWell:

- Обмеженість бази даних: Хоча додаток і має велику базу даних продуктів, він може не завжди розпізнавати менш популярні або місцеві бренди.
- Залежність від оновлень: Точність інформації залежить від частоти оновлень даних, що може вплинути на актуальність інформації про продукти.
- Реклама та просування: Деякі користувачі можуть відчувати, що додаток схильний до підвищеної кількості рекламних та просувальних матеріалів.

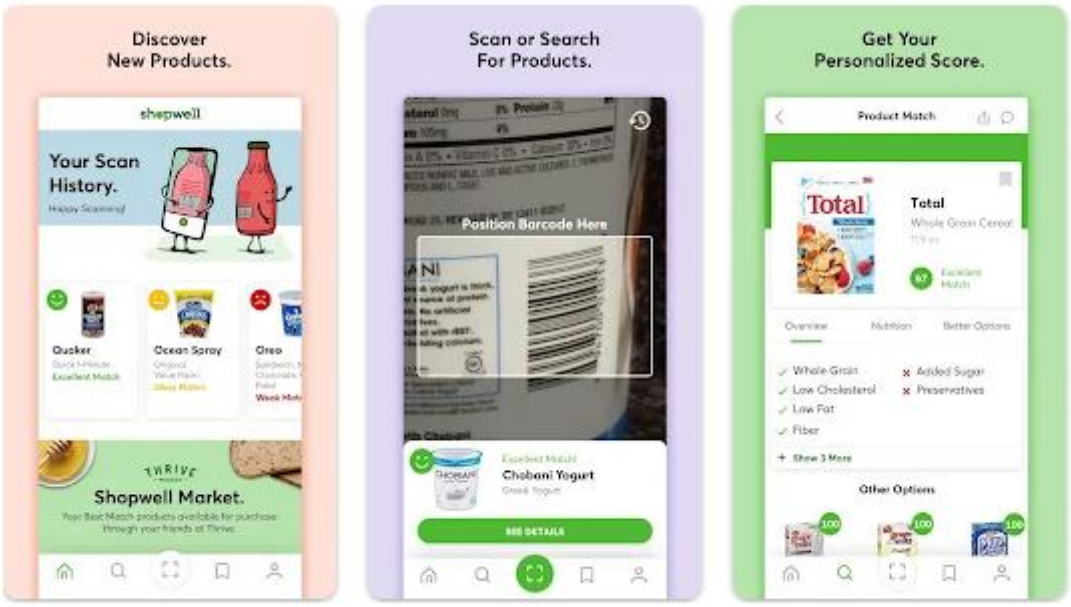

Рисунок 1.5 – Приклад роботи додатку ShopWell

Кожен із цих додатків має свої унікальні функції та спеціалізації, і вони відіграють важливу роль у забезпеченні користувачів інформацією, яка може значно вплинути на їхнє здоров'я і добробут.

Результати порівняння аналогів зведено в табл. 1.1.

Таблиця 1.1 – Порівняльні характеристика онлайн платформ

Критерії	MyFitnessPal	Yuka	Fooducate	AlLergyEats	ShopWell	Власний додаток
Функція сканування штрих-кодів	+	+	+	-	+	+
Аналіз харчового складу	+	-	+	-	-	+
Персоналізовані рекомендації	-	+	-	-	+	+
Інформація про харчову цінність	+	+	+	-	-	+
Наявність мобільного додатку	+	+	+	+	+	+
Загальна оцінка	80%	80%	80%	20%	60%	100%

Відповідно до аналізу існуючих аналогів і їх характеристик, створення власного мобільного додатку з функціями аналізу складу харчових продуктів є обґрунтованим кроком. Новий продукт заповнить прогалини у функціоналі та можливостях, які мають існуючі додатки, та надасть користувачам додаткову цінність завдяки унікальному поєднанню інноваційних технологій і користувацького досвіду.

Отже розробка власного програмного продукту з інтегрованими інноваційними рішеннями дозволить не тільки задовольнити існуючі потреби споживачів, але й відкрити нові можливості для інтерактивної взаємодії з продуктами, що надходять на ринок.

1.3 Аналіз методів розв’язання поставленої задачі

Існують різні методи розробки програмних модулів для аналізу складу харчових продуктів, кожен з яких має свої переваги та недоліки. У цьому розділі детально проаналізовано найбільш ефективні методи розробки таких програмних модулів, щоб визначити найкращий підхід до вирішення поставленої задачі.

Перший метод полягає у створенні відкритих API бібліотек, які дозволяють розробникам легко інтегрувати аналітичні функції в існуючі мобільні додатки. Цей підхід забезпечує велику гнучкість і скорочує час на розробку, оскільки розробники можуть використовувати готові інструменти замість створення їх з нуля. Також API бібліотеки дозволяють легко оновлювати функціонал без втручання в основний код додатка [9]. Проте, цей підхід може включати залежність від стороннього постачальника і потенційні проблеми з безпекою та приватністю даних, оскільки обробка даних відбувається за межами локальної системи.

Другий підхід передбачає створення вбудованих модулів, які інтегровані безпосередньо у мобільний додаток. Вбудовані модулі можуть бути налаштовані для виконання специфічних завдань і оптимізовані для забезпечення високої продуктивності. Однак, розробка таких модулів вимагає значних ресурсів, включаючи час та витрати на розробку та тестування, що може стати обмеженням для проектів із обмеженим бюджетом.

Третій підхід включає використання хмарних обчислень для аналізу даних. Цей метод дозволяє зберігати і обробляти великі масиви даних на хмарних серверах, забезпечуючи високу швидкість обробки та масштабованість ресурсів. Хмарні обчислення забезпечують доступ до потужних обчислювальних можливостей без необхідності інвестування в дороге серверне обладнання. Однак, це також може створити залежність від хмарного провайдера та потенційні ризики для приватності даних, оскільки обробка і зберігання інформації відбуваються за межами організації.

Результати порівняння аналогів зведено в табл. 1.2.

Таблиця 1.2 – Порівняння методів розв’язання поставленої задачі

Метод	Переваги	Недоліки
Використання відкритих API бібліотек	Легка інтеграція в існуючі додатки, скорочення часу на розробку, можливість легко оновлювати функціонал	Залежність від стороннього постачальника, проблеми з приватністю даних через обробку за межами локальної системи
Створення вбудованих модулів	Вищий рівень контролю над обробкою даних, краща безпека, оскільки дані обробляються локально	Значні ресурси на розробку та тестування вищі витрати, довший час на розробку
Використання хмарних обчислень	Висока швидкість обробки, масштабованість ресурсів, доступ до потужних обчислювальних можливостей без інвестицій в дороге обладнання	Залежність від хмарного провайдера Потенційні ризики для приватності даних, дані обробляються і зберігаються за межами організації
Комбінований підхід	Висока продуктивність і безпека, локальна обробка забезпечує контроль і приватність	Складність реалізації, може вимагати додаткових ресурсів для інтеграції і підтримки

Після ретельного аналізу переваг і недоліків кожного методу, було вирішено використовувати комбінований підхід для розробки програмного засобу. Комбінація вбудованих модулів для локальної обробки даних з

інтеграцією хмарних API для обробки великих масивів даних дозволить створити ефективну і гнучку систему. Такий підхід не тільки забезпечить високу продуктивність і безпеку, але й дозволить системі легко адаптуватися до майбутніх технологічних змін та нових вимог користувачів.

Отже, комбінований підхід, що включає використання вбудованих модулів для локальної обробки даних і інтеграцію хмарних API для обробки великих масивів даних, є найбільш ефективним методом розробки програмного засобу для аналізу складу харчових продуктів. Такий підхід дозволяє забезпечити високу продуктивність і безпеку, зберігаючи при цьому гнучкість і адаптивність до майбутніх технологічних змін та нових вимог користувачів.

1.4 Постановка задачі розробки програмного забезпечення

Проаналізувавши переваги та недоліки існуючих програмних рішень, призначених для аналізу складу харчових продуктів, було виявлено ряд важливих аспектів, які потребують удосконалення. Ці аспекти стали основою для формулювання завдань, які має вирішити розроблювана програма. Для ефективної реалізації проекту були визначені наступні ключові завдання:

Функціональні вимоги:

1. Введення даних про харчові продукти:
    - Можливість введення назви продукту.
    - Введення інформації про складові компоненти (білки, жири, вуглеводи, вітаміни, мінерали тощо).
    - Завантаження фотографій продукту
2. База даних продуктів:
    - Доступ до бази даних з інформацією про склад популярних харчових продуктів.
    - Можливість додавання, редагування та видалення продуктів з бази даних.
3. Інтерфейс користувача:
    - Зручний та інтуїтивно зрозумілий інтерфейс для введення та

аналізу даних.

Нефункціональні вимоги

1. Продуктивність:
   - Швидкий відгук на запити користувачів (максимальний час відгуку – 2 секунди).
   - Оптимізація для роботи з великими обсягами даних.
2. Надійність:
   - Високий рівень безвідмовності (не більше 1% допустимих збоїв).
   - Зберігання даних з резервним копіюванням.
3. Масштабованість:
   - Можливість розширення функціональності без значних змін у кодовій базі.
   - Підтримка збільшення кількості користувачів без втрати продуктивності.
4. Юзабіліті:
   - Забезпечення зручного користувацького досвіду.
   - Наявність інструкцій та підказок для користувачів.
5. Підтримка та обслуговування:
   - Наявність документації для користувачів та розробників.
   - Регулярні оновлення та виправлення помилок.

Отже, визначення цих ключових завдань і вимог дозволить створити програму, яка не тільки забезпечить точний аналіз складу харчових продуктів, але й буде зручною, надійною та масштабованою для широкого кола користувачів.

1.5 Висновки

На основі проведеного аналізу стану мобільних систем з використанням засобів аналізу складу харчових продуктів було встановлено, що існуючі рішення мають значні обмеження щодо точності та зручності використання.. Було проведено аналіз популярних додатків, таких як: "Yuka", "Fooducate",

"MyFitnessPal", " ShopWell" і " AllergyEats". Порівняльний аналіз аналогів виявив відсутність комплексного підходу до розв’язання проблеми, що включає інтеграцію сучасних технологій обробки даних та зручного користувацького інтерфейсу. У результаті порівняння було виявлено, що більшість з цих систем зосереджуються на відстеженні калорійності та харчових властивостей продуктів, але меншою мірою звертають увагу на детальний аналіз складних інгредієнтів та виявлення потенційних алергенів.

Проаналізувавши переваги та недоліки існуючих систем, було виявлено, що вони не надають достатньої точності в ідентифікації шкідливих хімічних добавок та алергенів, що використовуються у харчових продуктах. Також була відзначена недостатня інформативність інтерфейсів та обмежена функціональність в контексті персоналізації користувацького досвіду. Таким чином було доведено доцільність розробки власного програмного рішення, яке зосередиться на цих аспектах.

Аналіз методів розв’язання поставленої задачі продемонстрував ефективність застосування програмних методів аналізу складу харчових продуктів у поєднанні з ретельним тестуванням програмного забезпечення та розробкою інтуїтивно зрозумілого інтерфейсу користувача. У результаті було сформульовано задачу розробки програмного забезпечення, що передбачає створення надійного інструменту для аналізу складу харчових продуктів, який забезпечить користувачів точною та достовірною інформацією, сприятиме зниженню ризиків для здоров'я та покращенню якості харчування.

2 РОЗРОБКА СТРУКТУРИ, МЕТОДУ ТА АЛГОРИТМІВ РОБОТИ ПРОГРАМНОГО ПРОДУКТУ

2.1 Розробка моделі системи

Для кращого розуміння роботи програмних модулів мобільної системи для аналізу складу харчових продуктів було вирішено розробити модель роботи системи на прикладі UML діаграми діяльності [10] (рис. 2.1).

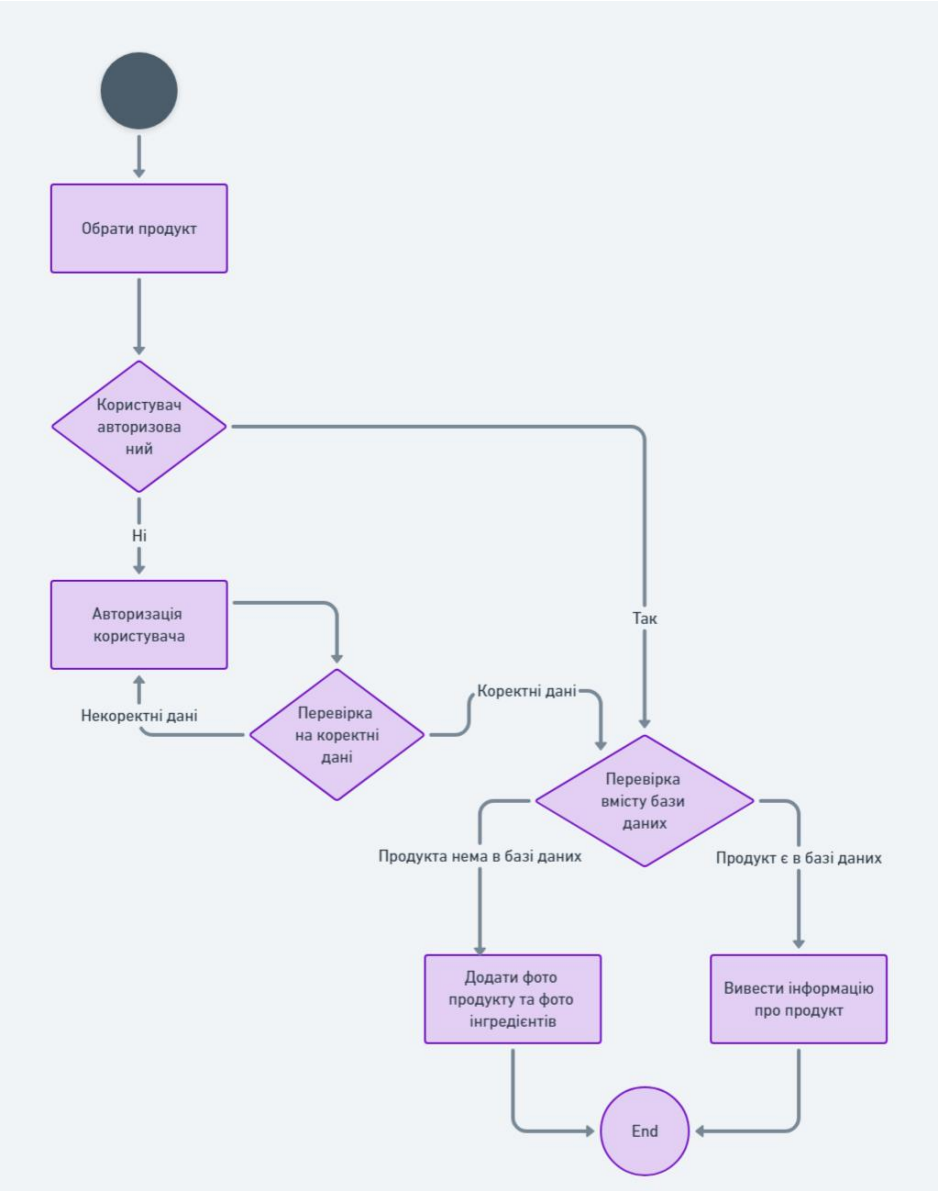

Рисунок 2.1 – Діаграма діяльності мобільної системи

Діаграма зображує процес взаємодії користувача з програмним засобом для аналізу складу харчових продуктів.

Діаграма має наступні етапи:

1. Обрати продукт:
   - Користувач починає процес, обираючи продукт.
2. Перевірка авторизації користувача:
   - Система перевіряє, чи авторизований користувач.
3. Користувач не авторизований:
   - Якщо користувач не авторизований, система проводить процедуру авторизації.
   - Після авторизації система перевіряє, чи введені дані є коректними.
4. Некоректні дані:
   - Якщо дані некоректні, користувач повинен ввести правильні дані знову.
5. Коректні дані:
   - Якщо дані коректні, система переходить до перевірки наявності продукту в базі даних.
6. Перевірка вмісту бази даних:
   - Система перевіряє, чи є обраний продукт в базі даних.
7. Продукта немає в базі даних:
   - Якщо продукта немає в базі даних, користувачу пропонується додати фото продукту та фото інгредієнтів.
8. Продукт є в базі даних:
   - Якщо продукт є в базі даних, система виводить інформацію про продукт.

Процес завершується, коли користувач отримує необхідну інформацію про обраний продукт

2.2 Розробка алгоритмів роботи системи

Для розробки мобільної платформи, для аналізу інгредієнтів у харчових продуктах, необхідно розробити загальний алгоритм, згідно якого буде працювати мобільний додаток (рис. 2.2).

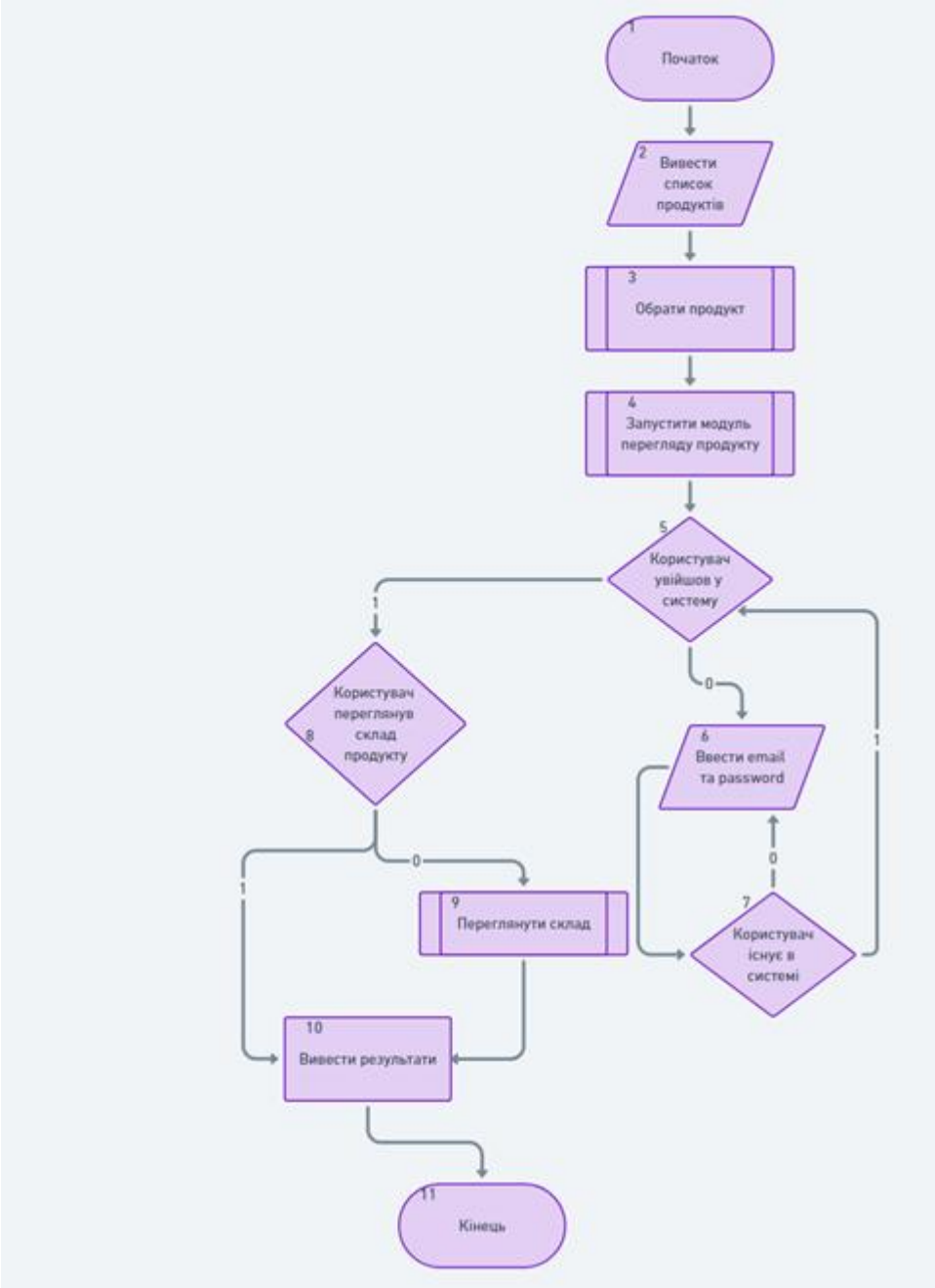

Рисунок 2.2 – Загальний алгоритм мобільної платформи

Згідно даного алгоритму, першим ділом, користувач отримує список харчових товарів. Далі, йому надається можливість обрати певний товар. Обравши товар, йому необхідно переглянути склад. Перед переглядом складу продукту відбувається перевірка статусу користувача в системі. Якщо користувач не увійшов до системи - йому пропонується виконати вхід. Доки користувач не авторизується в системі, у нього не буде права обрати товар, тим

самим, отримавши можливість детально побачити нутрієнтний склад, а також "приховані" небезпеки, такі як надлишок цукру, трансжирів і багато іншого.

Отже, після входу в систему, запускається модуль перегляду продуктів, який відображає на екрані склад, харчового продукту.

Розглянувши даний алгоритми, можна зрозуміти, що розроблений додаток дає змогу користувачам скласти свій раціон відштовхуючись від інформації про харчові продукти.

2.3 Розробка структури проєкту

Розробка програмного засобу для аналізу складу харчових продуктів включає кілька ключових етапів, одним з яких є створення структури застосунку. Цей розділ описує основні компоненти застосунку та їх взаємодію.

Структуру проекту зображено на рисунку 2.3.

Рисунок 2.3 – Структура проекту

Основні компоненти застосунку:

1. Activity:
    - Відповідає за взаємодію з користувачем.

- Реалізує екранні форми для введення даних, відображення результатів аналізу тощо.

2. Adapter:
   - Забезпечує зв'язок між джерелами даних та елементами інтерфейсу користувача.
   - Використовується для динамічного відображення списків та таблиць.
3. Barcode:
   - Модуль сканера штрих-коду.
   - Дозволяє сканувати штрих-коди продуктів для швидкого отримання інформації про них.
   - Інтегрується з базою даних для автоматичного заповнення інформації про продукт.
4. CustomViews:
   - Включає користувацькі елементи інтерфейсу, які не входять до стандартних бібліотек.
   - Забезпечує більш гнучкий та інтуїтивно зрозумілий інтерфейс.
5. Fragment:
   - Використовується для розбиття інтерфейсу на окремі модулі.
   - Дозволяє створювати багаторазово використовувані частини інтерфейсу, які можна комбінувати в різних екранах.
6. Interfaces:
   - Включає визначення інтерфейсів для взаємодії між різними модулями.
   - Забезпечує стандартизацію методів та сприяє модульності коду.
7. OCR (Optical Character Recognition):
   - Модуль оптичного розпізнавання символів.
   - Дозволяє розпізнавати текст на зображеннях, що надає можливість аналізувати склад продуктів за допомогою фотографій.

8. Security:
    - Забезпечує безпеку даних користувачів.
    - Реалізує механізми аутентифікації та авторизації, захисту даних та управління правами доступу.

Таким чином, структура застосунку забезпечує зручне та ефективне виконання основних завдань користувачів, а також легко модифікується та розширюється у разі потреби.

2.4 Висновки

У цьому розділі було успішно створено основу для ефективного функціонування системи аналізу складу харчових продуктів.

Було створено детальну модель системи, яка охоплює всі необхідні компоненти та їх взаємодію, що забезпечує злагоджену роботу системи та зручність для користувачів.

Розроблено ефективні алгоритми, які дозволяють здійснювати точний аналіз харчових продуктів, обробляти введені дані, перевіряти їх коректність та взаємодіяти з базою даних, алгоритми включають процедури авторизації користувача, обробки даних про продукти та виведення результатів.

Визначено чітку структуру проєкту, яка включає всі необхідні модулі та компоненти для забезпечення повного функціоналу системи. Розробка структури також передбачала інтеграцію зручного користувацького інтерфейсу та забезпечення масштабованості та надійності системи.

3 РОЗРОБКА ПРОГРАМНИХ КОМПОНЕНТІВ

3.1 Варіантний аналіз і обґрунтування вибору мови програмування

Вибір мови програмування є ключовим рішенням у процесі розробки програмного засобу. Для проекту, який передбачає аналіз складу харчових продуктів, важливими факторами є продуктивність, надійність, мультиплатформеність та підтримка великої кількості даних. У цьому розділі детально обгрунтовано вибір Java як оптимальної мови програмування для виконання вказаних вимог.

1. Python [11]

Python є однією з найпопулярніших мов програмування завдяки своїй простій і зрозумілій синтаксичній структурі, що прискорює розробку. Вона пропонує велику кількість бібліотек для обробки даних та машинного навчання. Однак Python має свої недоліки. Вона менше продуктивна у порівнянні з компільованими мовами, такими як C++ чи Java, що може бути критично важливим для додатків, які потребують високої продуктивності.

2. C++ [12]

C++ відома своєю високою продуктивністю та ефективністю у використанні ресурсів, що робить її ідеальною для додатків, які потребують великої обчислювальної потужності. Однак C++ має складну синтаксичну структуру і круту криву навчання, що може ускладнити розробку для новачків.

3. JavaScript [13]

JavaScript є основною мовою для веб-розробки та інтеграції з веб-додатками. Вона має велику кількість бібліотек та фреймворків, таких як React, Angular та Vue.js, які спрощують розробку інтерактивних веб-додатків. Однак JavaScript має обмежену продуктивність у порівнянні з іншими мовами програмування і відсутність вбудованих засобів для обробки великих обсягів даних, що може бути проблемою для додатків, які потребують інтенсивної обробки даних.

Результати порівняння мов програмування в табл. 3.1.

Таблиця 3.1 – Порівняння мов програмування

Мова програмування	Переваги	Недоліки	Оцінка
Python	Проста і зрозуміла синтаксична структура Велика кількість бібліотек для обробки даних та машинного навчання	Менша продуктивність у порівнянні з компільованими мовами Обмежена підтримка мобільних платформ	Середня
C++	Висока продуктивність Широкі можливості для низькорівневої обробки даних Підтримка багатьох платформ	Складна синтаксична структура Більший час розробки через необхідність управління пам'яттю	Середня
JavaScript	Підходить для веб-розробки Велика кількість бібліотек та фреймворків Кросплатформеність завдяки використанню у браузерах	Обмежена продуктивність Відсутність вбудованих засобів для обробки великих обсягів даних	Середня
Java	Висока продуктивність та стабільність Велика кількість бібліотек та фреймворків Потужна підтримка об'єктно-орієнтованого програмування	Може вимагати більше ресурсів для виконання	Висока

Кожна мова програмування має свої сильні сторони, однак Java вирізняється завдяки:

- Широкі можливості для розробки як десктопних, так і мобільних додатків - Java використовується для розробки додатків на платформах Android, що становить значну частку мобільного ринку.
- Переносимість коду - "Напиши раз, запускай скрізь" (Write Once, Run Anywhere - WORA) залишається привабливою характеристикою Java, забезпечуючи безпроблемну роботу Java-додатків на різних платформах без необхідності зміни коду.

Завдання аналізу даних великої кількості харчових продуктів вимагає високої обробної потужності та здатності ефективно працювати з базами даних. Java пропонує:

- Масштабованість та багатопоточність - Java дозволяє ефективно розподіляти обчислювальні завдання на кілька потоків, що зменшує час відгуку і підвищує продуктивність обробки даних.
- Безпека - Важливий аспект при роботі з даними, Java надає сильні засоби безпеки на рівні мови та виконавчого середовища.
- Багатий набір інструментів для роботи з базами даних - JDBC (Java Database Connectivity) та інтеграція з ORM фреймворками як Hibernate забезпечують ефективну взаємодію з різноманітними базами даних.

Ця мова програмування має синтаксис, який нагадує звичайну англійську мову, та використовує мінімум складних для запам’ятовування символів. Після встановлення JDK, налаштування PATH і розуміння Classpath, спеціаліст вже може створювати базові програми на Java.

З вищезгаданих аргументів, Java є ідеальним вибором для цього проекту [14]. Вона не тільки підтримує всі необхідні технічні характеристики для розробки високопродуктивного програмного засобу, але й має значні переваги у вигляді переносимості та багатоплатформенності, що робить її оптимальною для розробки програмних продуктів, призначених для великої аудиторії користувачів.

З огляду на перелічені переваги та враховуючи конкретні потреби проекту, Java забезпечує всі необхідні інструменти для створення надійного,

безпечного та ефективного програмного засобу для аналізу складу харчових продуктів. Використання Java дозволить не тільки досягти поставлених цілей, але й забезпечити легке масштабування і супровід програмного продукту

3.2 Вибір середовища розробки

Для розробки мобільного додатку, який дозволяє аналізувати склад харчових продуктів, вибір правильного інтегрованого середовища розробки (IDE) має вирішальне значення. Серед них найбільш відомими є Android Studio, IntelliJ IDEA, Eclipse та NetBeans. Кожне з цих середовищ має свої переваги та недоліки.

1. IntelliJ IDEA [15]

IntelliJ IDEA є універсальним середовищем розробки, яке підтримує широкий спектр мов програмування та платформ, включаючи Java і Android. Багато функцій Android Studio взяті з IntelliJ IDEA, оскільки Android Studio базується на ньому. IntelliJ IDEA надає високоякісні функції автодоповнення, навігації по коду та рефакторингу. Основним недоліком є вартість повної версії (Ultimate), яка може бути недосяжною для окремих розробників або малих команд. Також налаштування середовища може бути складним для новачків.

2. Eclipse [16]

Eclipse є безкоштовним середовищем розробки з відкритим кодом, яке підтримує велику кількість плагінів, що дозволяє налаштувати середовище під конкретні потреби розробника. Воно підходить для широкого спектра мов програмування та платформ. Однак, налаштування Eclipse для розробки Android-додатків може бути складним і вимагати додаткових плагінів. Інтерфейс користувача може здаватися застарілим і менш інтуїтивним порівняно з сучасними середовищами розробки.

3. NetBeans [17]

NetBeans також є безкоштовним і з відкритим кодом середовищем розробки. Його інтерфейс користувача є більш інтуїтивним та простим у використанні, а також забезпечує відмінну підтримку для розробки на Java.

Проте, NetBeans не є основним середовищем для розробки Android-додатків, що може призвести до меншої кількості спеціалізованих інструментів та плагінів, і має менше доступних плагінів порівняно з Eclipse, що обмежує його функціональність.

4. Android Studio [18]

Android Studio є офіційним IDE для розробки Android-додатків, надаючи розробникам комплексний набір інструментів, які оптимізовані для розробки на Android.

Переваги використання Android Studio

- Інтегрованість з Android SDK: Android Studio постачається з вбудованою підтримкою Android SDK, що означає, що розробники можуть легко налаштовувати свої проекти під різні версії Android і різноманітні типи пристроїв.
- Emulator Integration: Вбудовані емулятори дозволяють розробникам швидко тестувати і відлагоджувати свої додатки в контрольованому середовищі, імітуючи різні пристрої та версії Android.
- Gradle-Based Build Support: Android Studio використовує Gradle для автоматизації процесів збірки, що забезпечує гнучкість і потужність управління залежностями і версіями проекту.
- Advanced Code Completion and Refactoring: Підтримка розумного завершення коду і рефакторингу значно підвищує продуктивність розробника, допомагаючи швидше писати більш чистий і ефективний код.
- Visual Layout Editor: Візуальний редактор макетів дозволяє легко створювати інтерфейси користувача, перетягуючи компоненти в редактор, що ідеально підходить для швидкого прототипування і розробки.

Android Studio також була чудово оновлена компанією Google із передовими інструментами. Деякі з останніх основних моментів включають:

- Посібник з візуального редагування в реальному часі для програмістів
- Наявність інструментів lint для контролю версій, продуктивності та

зручності використання

- Хороша можливість інтеграції з Google Cloud
- Можливість редагування на кількох екранах
- Візуалізація макета програми в режимі реального часу
- Варіанти забудови

Результати порівняння середовищ розробки в табл. 3.3.

Таблиця 3.2 – Порівняння середовищ розробки

Середовище розробки	Переваги	Недоліки
Android Studio	Повна інтеграція з Android SDK Потужний редактор коду з автодоповненням та рефакторингом Вбудований емулятор Android-пристроїв	Високі системні вимоги
IntelliJ IDEA	Універсальність Високоякісні інструменти автодоповнення та рефакторингу	Вартість повної версії Складність налаштування для новачків
Eclipse	Безкоштовний та з відкритим кодом Масштабованість за рахунок плагінів Гнучкість у підтримці багатьох мов програмування	Обмежена продуктивність Відсутність вбудованих засобів для обробки великих обсягів даних
NetBeans	Безкоштовний та з відкритим кодом Інтуїтивний і простий інтерфейс Відмінна підтримка для Java-розробки	Обмежена підтримка Android Менше доступних плагінів, ніж у Eclipse

Після розгляду різних середовищ розробки, я вибрав Android Studio для створення мобільного додатку. Основними причинами цього вибору є офіційна

підтримка Google, що гарантує актуальність та відповідність останнім стандартам та інструментам Android. Всі необхідні інструменти для розробки, тестування та розгортання Android-додатків вже включені, що зменшує необхідність додаткових налаштувань. Розширені функції автодоповнення, рефакторингу, а також потужні інструменти для відладки та аналізу продуктивності значно полегшують процес розробки. Вбудований емулятор дозволяє легко тестувати додатки на різних версіях Android та пристроях.

Таким чином, вибір Android Studio для розробки мобільного додатку на Java є обґрунтованим рішенням, що забезпечує високу продуктивність та ефективність розробки.

3.3 Проектування бази даних для зберігання інформації про харчові продукти та їх склад

Для створення бази даних для зберігання інформації про харчові продукти та їх склад, було використано два основні сервіси Firebase: Cloud Firestore та Cloud Storage. Ці сервіси забезпечують надійне та ефективне зберігання даних, а також зручний доступ до них.

Cloud Firestore [19] є гнучкою та масштабованою хмарною базою даних для зберігання та синхронізації даних в режимі реального часу (рис. 3.1). Основні переваги використання Cloud Firestore для нашого проекту включають:

1. Гнучка структура даних: Cloud Firestore підтримує зберігання даних у вигляді колекцій та документів, що дозволяє легко організувати інформацію про харчові продукти та їх склад.
2. Масштабованість: База даних автоматично масштабується в залежності від навантаження, забезпечуючи високу продуктивність навіть при великій кількості користувачів та запитів.
3. Реальний час: Cloud Firestore підтримує синхронізацію даних в режимі реального часу, що дозволяє користувачам отримувати оновлення миттєво після внесення змін.

4. Безпека: Використання правил безпеки Firebase дозволяє контролювати доступ до даних на основі аутентифікації користувачів та інших умов.

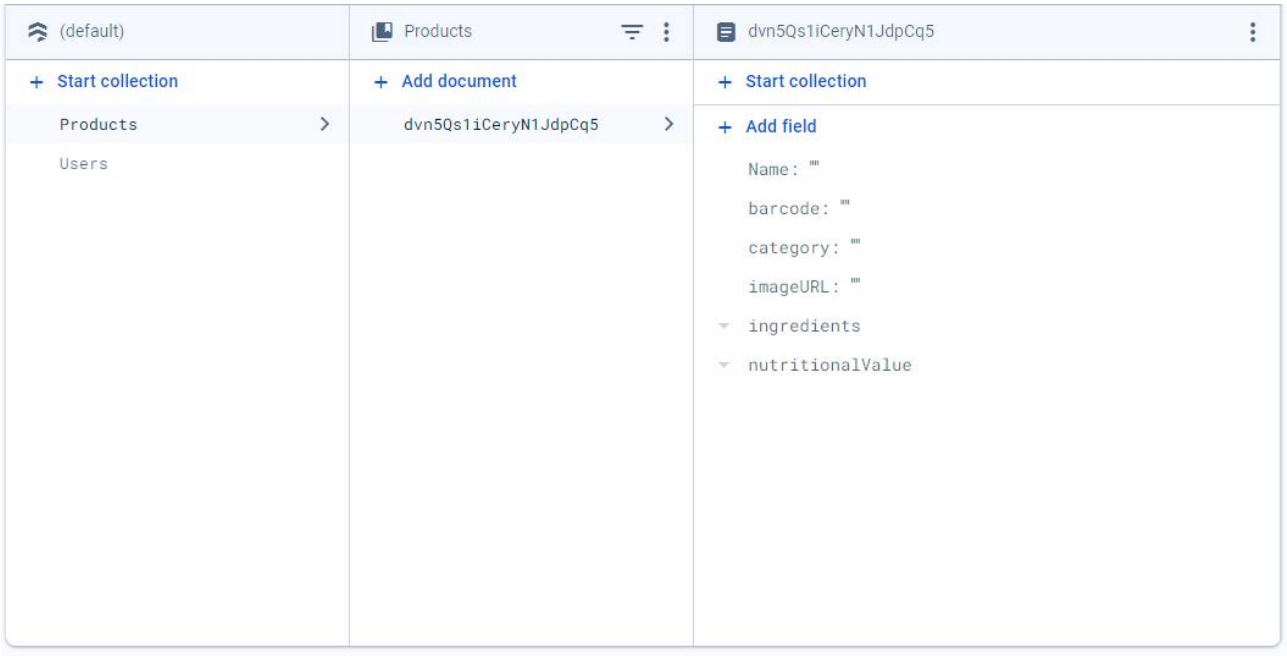

Рисунок 3.1 – Cloud Firestore

Основна структура даних у Cloud Firestore включає:

- Колекції: Наприклад, колекція Products, яка містить документи про окремі харчові продукти.
- Документи: Кожен документ у колекції Products містить інформацію про конкретний продукт, включаючи назву, склад, харчову цінність та інші атрибути.

Cloud Storage[20] від Firebase забезпечує зберігання великих файлів, таких як зображення, відео або інші медіафайли. У нашому проекті Cloud Storage використовується для зберігання зображень харчових продуктів та інших пов'язаних файлів. Основні переваги використання Cloud Storage включають:

1. Масштабованість та продуктивність: Cloud Storage забезпечує високу швидкість завантаження та вивантаження файлів, а також автоматичне масштабування для підтримки великих обсягів даних.

2. Інтеграція з іншими сервісами Firebase: Cloud Storage легко інтегрується з Cloud Firestore та іншими сервісами Firebase, що дозволяє створювати комплексні рішення.

3. Безпека: Правила безпеки Firebase для Cloud Storage дозволяють контролювати доступ до файлів на основі аутентифікації користувачів та інших умов.

Основні компоненти Cloud Storage включають:

- Buckets**:** Контейнери для зберігання файлів. У нашому проекті використовується один основний bucket для зберігання всіх зображень харчових продуктів.
- Файли (Objects)**:** Кожен файл в bucket представляє собою зображення або інший медіафайл, пов'язаний з конкретним харчовим продуктом.

У базі даних є дві основні колекції: колекція для продуктів та колекція для користувачів. Кожна з цих колекцій має свою структуру та призначення. Структура кожного документу в базі даних виглядає наступним чином:

Колекція Products містить інформацію про харчові продукти. Ця колекція відповідає за зберігання детальної інформації про його склад, харчову цінність, зображення та інші характеристии(рис 3.2). Основні поля в документах цієї колекції включають:

- Назва (name)**:** Назва продукту.
- Склад (ingredients): Список складових компонентів продукту.
- Харчова цінність (nutritionalValue)**:** Об'єкт, що містить інформацію про харчову цінність продукту, включаючи калорії, білки, жири та вуглеводи.
- Зображення (imageURL)**:** URL-адреса до зображення продукту.
- Штрих-код (barcode)**:** Унікальний ідентифікатор продукту, що дозволяє швидко знаходити інформацію за допомогою сканера штрих-коду.
- Категорія (category)**:** Категорія, до якої належить продукт

Products	
string	name
string[]	ingredients
object	nutritionalValue
string	imageURL
string	barcode
string	category

Рисунок 3.2 – Колекція Products

Колекція Users містить інформацію про користувачів застосунку. Ця колекція відповідає за зберігання особистих даних, налаштувань облікового запису та історії запитів(рис 3.3). Основні поля в документах цієї колекції включають:

- Ім'я користувача (username): Унікальне ім'я користувача.
- Електронна пошта (email): Адреса електронної пошти користувача.
- Пароль (password): Хешований пароль для забезпечення безпеки.
- Історія запитів (queryHistory): Список запитів, зроблених користувачем, включаючи дати та результати.
- Уподобання (preferences): Налаштування користувача, такі як мова інтерфейсу та налаштування сповіщень.
- Дата реєстрації (registrationDate): Дата та час реєстрації користувача.
- Роль (role): Роль користувача в системі.

Users	
string	username
string	email
string	password
string[]	queryHistory
object	preferences
date	registrationDate
string	role

Рисунок 3.3 – Колекція Users

Ця структура дозволяє Cloud Firestore ефективно зберігати та керувати даними згідно з визначеними потребами додатку.

Взаємодія між колекціями Products та Users забезпечує комплексне управління даними про харчові продукти та користувачів. Користувачі можуть здійснювати запити на аналіз продуктів, результати яких зберігаються в їх історії запитів. Це дозволяє відстежувати історію взаємодії користувачів із застосунком та забезпечує персоналізований досвід.

Використання Cloud Firestore та Cloud Storage від Firebase забезпечує надійне та ефективне зберігання даних про харчові продукти та користувачів. Гнучка структура даних, масштабованість, підтримка синхронізації в реальному часі та інтеграція з іншими сервісами Firebase роблять ці інструменти ідеальним вибором для реалізації даного проекту. Завдяки цим технологіям ми можемо забезпечити користувачам швидкий доступ до актуальної інформації та зручність у використанні застосунку.

3.4 Розробка модуля сканеру для зчитування штрих-кодів

Для успішного зчитування та розпізнавання штрих-кодів важливо визначити, які формати штрих-кодів будуть підтримуватися нашим модулем. У

харчовій промисловості найпоширенішим форматом є EAN-13, який містить 13 цифр і використовується для ідентифікації товарів [21] (рис. 3.4).

У нашому проекті ми використовуємо бібліотеку для зчитування штрих-кодів, яка надає широкий спектр можливостей для налаштування. Для цього ми використовуємо метод setBarcodeFormats, щоб вказати, що наш детектор має розпізнавати штрих-коди формату EAN-13.

```
BarcodeDetector detector = new BarcodeDetector.Builder(getContext())
        .setBarcodeFormats(Barcode.EAN_13)
        .build();

detector.setProcessor(new Detector.Processor<Barcode>() {
```

Рисунок 3.4 – Вибір штрих-коду

Для зчитування штрих-кодів використовується камера мобільного пристрою. Важливою частиною процесу інтеграції камери є забезпечення безперебійного відображення відеопотоку, що надходить з камери, та його обробка в режимі реального часу.

Камера була інтегрована в застосунок за допомогою компонента SurfaceView. Цей компонент дозволяє відображати відеопотік з камери без значних затримок, що є критичним для забезпечення точної та швидкої обробки зображення штрих-коду

Перед початком використання камери, додаток спочатку запитує у користувача дозвіл на її активацію. Це необхідно для забезпечення безпеки та конфіденційності користувача. Після отримання дозволу камера налаштовується на розпізнавання штрих-коду. Використовуючи компонент SurfaceView, забезпечується безперебійне відображення відеопотоку з камери.

Коли камера активна, вона постійно аналізує зображення в пошуках штрих-кодів. Як тільки штрих-код успішно розпізнано, камера вимикається для економії ресурсів та запобігання зайвого використання. Після цього викликається метод tryToVerifyProduct презентера, куди передається розпізнаний штрих-код як параметр. Цей метод займається подальшою

обробкою та верифікацією продукту на основі отриманого штрих-коду. (рис. 3.5).

```
public void receiveDetections(Detector.Detections<Barcode> detections) {
    final SparseArray<Barcode> barcodes = detections.getDetectedItems();
    if (barcodes != null && barcodes.size() > 0) {
        view.post(new Runnable() {
            @Override
            public void run() {
                cameraSource.stop();
                mPresenter.tryToVerifyProduct(barcodes.valueAt(0).displayValue);
            }
        });
    }
```

Рисунок 3.5 – Визначення штих-коду

Клас ScannerPresenter відповідає за обробку логіки сканування штрих-кодів та взаємодії з базою даних Firestore і системою аутентифікації Firebase. Нижче наведено фрагмент коду, який демонструє ініціалізацію цих компонентів.

FirebaseAuth використовується для отримання поточного екземпляру Firebase аутентифікації. Це важливо для того, щоб мати доступ до ідентифікатора поточного авторизованого користувача, який буде використаний у процесі підтвердження продукту, потім ініціалізується екземпляр Firestore, який буде використовуватись для взаємодії з базою даних.

Ініціалізація колекцій Products та Users є критичною, оскільки наступні операції вимагають доступу до цих даних для порівняння користувацьких уподобань з інгредієнтами продукту. Це забезпечує можливість верифікації продуктів на основі індивідуальних потреб користувача.

Застосування FirebaseAuth є важливим для отримання ідентифікатора поточного авторизованого користувача. Це дозволяє ідентифікувати користувача та використовувати його дані у процесі верифікації продуктів, що забезпечує персоналізований підхід до аналізу складу харчових продуктів.

Спочатку товар шукається в колекції Products за штрих-кодом, який був зчитаний під час сканування. Якщо товар знайдено, витягується його

ідентифікатор і об'єкт Products. Після цього, використовуючи FirebaseAuth, отримується поточний авторизований користувач і його дані шукаються в колекції Users. Якщо дані користувача знайдено, інформація про продукт додається до історії переглядів користувача. Після цього виконується перевірка, чи містить продукт інгредієнти, які користувач бажає уникати(рис. 3.6).

```
public void tryToVerifyProduct(String barcode) {
    mScannerView.showLoading();
    productRef.whereEqualTo("barcode", barcode)
        .limit(1)
        .get()
        .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
            @Override
            public void onComplete(@NonNull Task<QuerySnapshot> task) {
                if (task.isSuccessful()) {
                    if (task.getResult().getDocuments().size() < 1) {
                        mScannerView.onProductNotExist(barcode);
                    } else {
                        String productDocumentId = task.getResult().getDocuments().get(0).getId();
                        Product product = task.getResult().getDocuments().get(0).toObject(Product.class);
                        userRef.whereEqualTo("userId", mAuth.getCurrentUser().getUid())
                            .get()
                            .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
                                @Override
                                public void onComplete(@NonNull Task<QuerySnapshot> task) {
                                    if (task.isSuccessful()) {
                                        if (task.getResult().getDocuments().size() > 0) {
                                            String userDocumentId = task.getResult().getDocuments().get(0).getId();
                                            userRef.document(userDocumentId).collection("history")
                                                .document(productDocumentId).set(product, SetOptions.merge());
                                            boolean result = findMatch(product, task.getResult().getDocuments().get(0).toObject(User.class));
                                            if (result) {
                                                mScannerView.onProductContains(product);
                                                mScannerView.hideLoading();
                                            } else {
                                                mScannerView.onProductNotContains(product);
                                                mScannerView.hideLoading();
```

Рисунок 3.6 – Підтвердження продукту

Таким чином, цей код забезпечує персоналізований підхід до аналізу складу харчових продуктів, використовуючи дані про уподобання користувача для надання відповідних рекомендацій.

3.5 Розробка архітектури програмного забезпечення

Під час розробки архітектури програмного забезпечення особливу увагу було приділено забезпеченню максимальної дружності та простоти використання для кінцевих користувачів. Для підвищення візуальної привабливості та створення позитивного враження, як основний колір програми було обрано зелений. Зелений колір асоціюється з природою, гармонією та свіжістю, що додає користувачам відчуття спокою і довіри при використанні програми.

Підхід базувався на дотриманні основних принципів UX (User Experience) та UI (User Interface) дизайну, щоб забезпечити інтуїтивно зрозумілий, ефективний і приємний досвід користувача.

1. Принципи UX (User Experience) [22] дизайну:

- Корисність (Usability): Ми створили інтерфейс, який є зручним у використанні і дозволяє користувачам легко виконувати потрібні завдання. Всі функції програми добре структуровані, що знижує необхідність тривалого навчання.
- Зрозумілість (Clarity): Інтерфейс програми є логічним та зрозумілим, що мінімізує кількість можливих помилок користувача. Ми забезпечили чіткі вказівки та зворотний зв'язок для кожної дії користувача.
- Доступність (Accessibility): Ми врахували потреби різних груп користувачів, включаючи людей з обмеженими можливостями. Інтерфейс розроблений таким чином, щоб бути доступним для всіх користувачів, незалежно від їх фізичних можливостей або досвіду.
- Інтуїтивність (Intuitiveness): Ми дотримувалися принципу, що користувачі повинні інтуїтивно розуміти, як використовувати систему, без потреби в додаткових інструкціях. Використані елементи дизайну є знайомими для користувачів, що сприяє швидкому засвоєнню функціоналу.
- Задоволення (Satisfaction): Наша мета — забезпечити користувачам позитивний досвід використання програми. Ми прагнули зробити процес роботи з програмою приємним та зручним, що підвищує задоволеність користувачів.

2. Принципи UI (User Interface) [23] дизайну:

- Консистентність (Consistency): Всі елементи інтерфейсу, такі як кнопки, меню та іконки, мають однорідний стиль і поведінку. Це допомагає користувачам швидко орієнтуватися в програмі і зменшує когнітивне навантаження.
- Простота (Simplicity): Ми мінімізували кількість елементів інтерфейсу, залишивши лише найнеобхідніше. Простий і мінімалістичний

дизайн дозволяє користувачам зосередитися на виконанні своїх завдань без відволікань.

- Візуальна ієрархія (Visual Hierarchy): Ми організували інформацію таким чином, щоб користувачі могли швидко знайти потрібні елементи. Важливі елементи виділені розміром, кольором або розташуванням, що полегшує їх сприйняття.
- Відгук (Feedback): Кожна дія користувача супроводжується зворотним зв'язком. Наприклад, натискання кнопки супроводжується візуальними чи звуковими ефектами, що підтверджують успішне виконання дії.
- Естетичність (Aesthetics): Дизайн інтерфейсу привабливий та сучасний. Використання приємних кольорових схем, якісних іконок та графіки сприяє загальному позитивному враженню від програми.

Обидва напрями дизайну є критично важливими для розробки інтерфейсу, який не тільки виглядає добре, але й забезпечує високу якість взаємодії з користувачем.

Початковий екран програми NutriScope являє собою екран завантаження. Дизайн цього екрану мінімалістичний та зосереджений на центральному елементі, який представляє собою авокадо. На світлому фоні зображені ледь помітні контури різних продуктів харчування, що створює легкий і приємний візуальний ефект. У центрі екрану розміщено зображення половинки авокадо з кісточкою, під яким зеленим кольором написано назву програми - "NutriScope". Такий дизайн допомагає користувачам асоціювати програму з темою здорового харчування ще до початку використання додатку(рис. 3.7).

Рисунок 3.7 – Екран завантаження

Екран реєстрації в програмі NutriScope також має простий та зрозумілий дизайн. Центральним елементом знову є зображення половинки авокадо, яке розташоване під назвою програми, написаною зеленим кольором. (рис.3.8).

Нижче зображення авокадо розташовані три текстові поля для введення інформації:

1. Електронна пошта - поле для введення електронної адреси користувача.
2. Пароль - поле для введення паролю.
3. Введіть пароль ще раз - поле для підтвердження паролю.

Під текстовими полями розташована кнопка "Зареєструватись" зеленого кольору, яка служить для завершення процесу реєстрації.

Як працює цей екран:

1. Введення даних: Користувач вводить свою електронну адресу та пароль у відповідні текстові поля.
2. Перевірка пароля: Для підтвердження правильності введеного паролю, користувач повинен ввести його повторно в третьому текстовому полі.
3. Натискання кнопки: Після заповнення всіх полів, користувач натискає кнопку "Зареєструватись".

4. Обробка даних: Програма перевіряє правильність введених даних (наприклад, перевіряє, чи збігаються введені паролі).

5. Створення облікового запису: Якщо всі дані введено правильно, програма створює новий обліковий запис для користувача і, можливо, перенаправляє його на наступний екран, наприклад, екран вітання або головне меню.

Такий підхід забезпечує простий та інтуїтивно зрозумілий процес реєстрації для користувачів.

Рисунок 3.8 – Екран реєстрації користувача

Після того як реєстрація була успішно здійснена після першого запуску додатку, всі наступні рази користувач буде мати змогу авторизуватись де потрібно ввести «Електронну пошту» та «Пароль», (рис.3.9).

Рисунок 3.9 – Сторінка авторизації користувача

Після авторизації користувач потрапляє в головне меню програми NutriScope(рис. 3.10). Це меню має зручний та інтуїтивно зрозумілий інтерфейс, який допомагає швидко орієнтуватися у функціях програми.

Головне меню містить:

1. Рекомендовані продукти: У верхній частині екрану відображаються продукти, які рекомендовані для користувача. Це можуть бути популярні або корисні товари, які підходять до його дієти чи вподобань.

2. Список просканованих продуктів: Нижче розташований список продуктів, які були просканові користувачем. Кожен запис містить назву продукту та додаткову інформацію або опції для взаємодії.

Основні кнопки навігації:

- Сканувати: Кнопка у вигляді штрих-коду, розташована в центрі нижньої частини екрану. Вона дозволяє користувачеві швидко відкрити функцію сканування продуктів.

- Головна: Кнопка з іконкою будинку, розташована ліворуч від кнопки сканування. Вона повертає користувача до головного екрану з рекомендаціями та списком просканованих продуктів.
- Профіль: Кнопка з іконкою людини, розташована праворуч від кнопки сканування. Вона відкриває профіль користувача, де можна переглядати та редагувати особисту інформацію, налаштування та інші параметри.

Таке розташування елементів та навігаційні кнопки забезпечують зручність у користуванні та швидкий доступ до основних функцій програми.

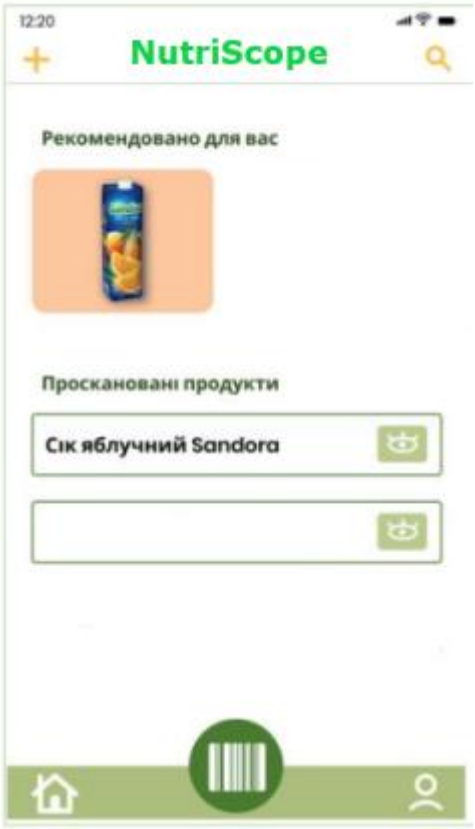

Рисунок 3.10 – Головне меню

Процес додавання нового продукту в додатку NutriScope складається з декількох ключових кроків (рис.3.11) . Кожен крок спрямований на забезпечення точного і повного введення інформації про продукт. Нижче описано деталі процесу:

1. Введення назви продукту:

- Користувач вводить назву продукту в спеціальне текстове поле, помічене як "Назва продукту". Це обов'язкове поле, без якого неможливо зберегти новий продукт.

2. Введення виробника:

- Виробник продукту вказується в текстовому полі "Виробник". Це також обов'язкове поле. Інформація про виробника є важливою для відстеження джерела продукту та його якості.

3. Введення інгредієнтів:

- Користувач вводить список інгредієнтів продукту в поле "Інгредієнти". Це поле обов'язкове, оскільки відомості про інгредієнти є критичними для оцінки харчової цінності та безпеки продукту для споживачів.

4. Фотографування продукту:

- Користувач має можливість зробити фотографію самого продукту, натиснувши на кнопку з іконкою камери, позначену як "Фото продукту *". Це дозволяє візуально ідентифікувати продукт і забезпечити додаткову інформацію про його вигляд та упаковку.

5. Фотографування списку інгредієнтів на упаковці:

- Для забезпечення точності введеної інформації, користувач також фотографує список інгредієнтів на упаковці продукту. Це робиться за допомогою іншої кнопки з іконкою камери, позначеної як "Фото інгредієнтів".

6. Збереження даних:

- Після введення всієї необхідної інформації та завантаження фотографій, користувач натискає кнопку "Зберегти", щоб зберегти новий продукт в базі даних додатку NutriScope.

Рисунок 3.11 – Екран додавання нового продукту в базу даних

Цей процес дозволяє забезпечити комплексний підхід до додавання нових продуктів в додатку, включаючи як текстову, так і візуальну інформацію, що сприяє підвищенню точності даних і покращенню користувацького досвіду.

Після того як користувач відсканує штрих-код продукту з'являється екран з результатами сканування в додатку(рис.3.12). Під назвою додатку розміщена велика іконка камери. Цей елемент інтерфейсу використовується для перегляду фотографії продукту, який було відскановано, під фотографією знаходиться назва продукту.

Під назвою продукту виводиться детальна інформація, що складається з трьох основних пунктів:

- Штрих-код: Відображається штрих-код продукту.
- Виробник: Інформація про виробника продукту.
- Інгредієнти: Перелік основних інгредієнтів продукту.

Рисунок 3.12 – Екран Продукту

Цей екран забезпечує користувачам миттєвий доступ до основної інформації про продукт, який було відскановано, дозволяючи їм легко оцінити виробника та склад продукту. Інтуїтивний дизайн допомагає швидко знайти потрібну інформацію та перейти до інших функцій додатку при необхідності.

Розробка графічного інтерфейсу була проведена у середовищі розробки Android Studio з дотриманням основних принципів UX і UI дизайну.

3.6 Висновки

Отже для створення мобільного додатку було обрано Java як мову програмування, а також Android Studio як середовище розробки.

Java разом із Android Studio відповідає ключовим вимогам проекту, забезпечуючи стабільність, високу продуктивність і широкі можливості для мобільного розроблення. Вибір цих технологій дозволяє ефективно використовувати багатий набір функцій та інструментів, які ідеально підходять для створення складних за логікою аналітичних додатків.

Обрані інструменти мають великі і активні спільноти, що забезпечує доступ до обширних ресурсів, включаючи бібліотеки, фреймворки,

документацію та форуми для розробників. Це сприяє не тільки полегшенню розробки, але й забезпечує швидке вирішення потенційних проблем та доступ до останніх технологічних нововведень.

Використання Android Studio забезпечує оптимальну інтеграцію з ОС Android, що є найпопулярнішою мобільною платформою у світі. Це гарантує, що додаток буде легко адаптуватись до різноманітних пристроїв та версій Android, забезпечуючи широке покриття аудиторії.

Вбудовані інструменти для тестування та емуляції в Android Studio дозволяють проводити глибоке тестування програми під різними умовами, що є критично важливим для забезпечення якості та надійності додатку перед його запуском.

Java та Android Studio разом формують міцну платформу, яка підтримує легке розширення та оновлення додатків, що важливо для тривалої підтримки проекту та його адаптації до змінних вимог користувачів і технологій.

Було реалізовано модуль сканеру для зчитування штрих-кодів, проведено проектування бази даних для зберігання інформації про харчові продукти та їх склад, а також розроблено архітектуру програмного забезпечення. Оці кроки становлять ключові етапи в реалізації програмного засобу для аналізу складу харчових продуктів і визначають його функціональність, ефективність та користувацьку зручність.

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Аналіз методів тестування програмного забезпечення

Тестування програмного забезпечення є критично важливим етапом розробки будь-якого додатка, оскільки воно забезпечує перевірку якості та надійності функціональності перед її впровадженням в експлуатацію. В даному розділі буде проведено аналіз основних методів тестування, які можна застосувати до мобільного додатка для аналізу складу харчових продуктів, написаного на Java у середовищі Android Studio.

1. Юніт-тестування (Unit Testing) [24]

Модульне тестування передбачає перевірку окремих модулів або компонентів програмного забезпечення. Модулі тестуються ізольовано один від одного, щоб гарантувати, що кожен з них працює правильно.

Переваги:

- Виявлення дефектів на ранніх стадіях розробки.
- Легкість у пошуку та виправленні помилок.
- Можливість автоматизації, що дозволяє часто виконувати тести.

Недоліки:

- Не виявляє проблем, пов'язаних з інтеграцією модулів.
- Потребує значних зусиль на написання тестів.

2. Інтеграційне тестування (Integration Testing) [25]

Після того, як юніт-тести підтвердять правильність роботи окремих модулів, наступним кроком є інтеграційне тестування. Цей метод перевіряє, наскільки коректно різні модулі програми працюють разом. Для мобільного додатку важливо перевірити інтеграцію між модулем сканера штрих-кодів та системою управління базами даних.

Переваги:

- Виявлення дефектів, які виникають при взаємодії модулів.
- Перевірка коректності інтерфейсів між модулями.

Недоліки:

- Складність в налаштуванні середовища тестування.
- Можливі труднощі у визначенні джерела помилки.

3. Системне тестування (System Testing) [26]

Системне тестування оцінює додаток в цілому, перевіряючи його поведінку в реальних умовах використання. Це включає тестування на різних пристроях з різними версіями Android, щоб забезпечити сумісність та відсутність помилок, пов'язаних із особливостями платформи.

Переваги:

- Комплексне тестування всієї системи.
- Виявлення дефектів, які можуть виникнути лише при повній інтеграції.

Недоліки:

- Високі витрати часу і ресурсів.
- Може бути важко відтворити конкретні сценарії помилок.

4. Тестування користувацького інтерфейсу (UI Testing) [27]

Тестування інтерфейсу користувача важливе для забезпечення інтуїтивно зрозумілої та зручної взаємодії з додатком. Використання таких інструментів як Espresso чи UI Automator дозволяє автоматизувати тестування інтерфейсів, включаючи тестування взаємодій із компонентами і переходів між екранами.

Переваги:

- Забезпечує стабільність програмного забезпечення після змін.
- Можливість автоматизації, що дозволяє швидко виконувати тести після кожної зміни.

Недоліки:

- Потребує значних зусиль на підтримку тестових сценаріїв.
- Може пропустити нові дефекти, якщо не охоплює всі можливі сценарії.

5. Приймальне тестування (Acceptance Testing) [28]

Приймальне тестування проводиться для того, щоб переконатися, що програмне забезпечення відповідає всім вимогам та готове до впровадження.

Цей етап часто включає бета-тестування з реальними користувачами для оцінки функціональності та зручності програми в реальних умовах.

Переваги:

- Забезпечує відповідність продукту очікуванням замовника.
- Виявлення дефектів, які можуть бути важливими з точки зору кінцевих користувачів.

Недоліки:

- Залежність від чітких та повних вимог.
- Може бути складним для автоматизації.

Тестування програмного забезпечення є багатогранним процесом, що включає різні методи, кожен з яких відіграє важливу роль у забезпеченні якості продукту. Модульне тестування дозволяє виявити дефекти на ранніх стадіях, інтеграційне тестування забезпечує коректну взаємодію між модулями, системне тестування гарантує комплексну перевірку всієї системи, регресійне тестування зберігає стабільність програмного забезпечення після змін, а прийомне тестування забезпечує відповідність вимогам користувачів. Використання комбінованого підходу, що включає всі ці методи, дозволяє досягти високої якості програмного забезпечення та задоволення потреб кінцевих користувачів.

4.2 Тестування розробленого програмного продукту

Програмний продукт було ретельно протестовано з використанням усіх основних методів тестування програмного забезпечення, щоб забезпечити його високу якість, стабільність та відповідність вимогам користувачів. Ось детальний опис того, як було проведено тестування:

1. Модульне тестування (Unit Testing)

На першому етапі було проведено модульне тестування для перевірки окремих компонентів програми. Кожен модуль (окрема функція або клас) був ізольовано протестований за допомогою спеціальних тестових сценаріїв, що дозволило виявити та виправити дефекти на ранніх стадіях розробки. Було

використано різні тестові фреймворки, такі як JUnit, що дозволило автоматизувати процес тестування та швидко ідентифікувати помилки в коді. Завдяки цьому, було забезпечено високу якість базових елементів програмного продукту.

2. Інтеграційне тестування (Integration Testing)

Після завершення модульного тестування було проведено інтеграційне тестування для перевірки взаємодії між різними модулями програми. Всі модулі були об'єднані у групи, і їх спільна робота була протестована на предмет коректної взаємодії та передачі даних. Це дозволило виявити та виправити проблеми, які виникають при об'єднанні окремих компонентів у єдину систему. Для інтеграційного тестування використовувалися такі інструменти, як TestNG, що забезпечили ефективне проведення тестів.

3. Системне тестування (System Testing)

Наступним етапом було системне тестування, яке передбачає перевірку всієї інтегрованої системи в цілому. Продукт був протестований з точки зору функціональних та нефункціональних вимог, включаючи продуктивність, безпеку, сумісність та користувацький інтерфейс. Системне тестування забезпечило перевірку всіх аспектів роботи програми, гарантувавши її повну відповідність вимогам специфікації та очікуванням користувачів. Для цього етапу було використано різні інструменти автоматизованого тестування, такі як Selenium для тестування користувацького інтерфейсу та JMeter для тестування продуктивності.

4. Регресійне тестування (Regression Testing)

Після внесення змін до програмного продукту було проведено регресійне тестування для перевірки стабільності програмного забезпечення. Регресійне тестування включало повторне виконання всіх попередніх тестових сценаріїв, щоб переконатися, що нові зміни або виправлення не порушили існуючу функціональність. Автоматизація цього процесу за допомогою таких інструментів, як Jenkins, дозволила швидко виконувати тести після кожної зміни, забезпечуючи постійну стабільність і надійність продукту.

5. Прийомне тестування (Acceptance Testing)

Фінальним етапом було прийомне тестування, яке проводилося для перевірки відповідності програмного продукту вимогам користувачів та замовників. Це тестування проводилося з використанням реальних сценаріїв використання програми, які відображають очікувані дії кінцевих користувачів. Прийомне тестування підтвердило, що продукт відповідає всім заданим вимогам, готовий до впровадження та використання. Користувачі та замовники взяли участь у цьому етапі, що дозволило отримати їх зворотний зв'язок та внести необхідні корективи перед остаточним релізом.

Наступний етап полягає у тестуванні екрану реєстрації, де ми спробуємо зареєструвати користувача, не вказуючи його ім'я. Якщо користувач намагатиметься зареєструватися без вказівки імені, система видасть повідомлення про помилку та запросить внести необхідні дані. Це є коректною відповіддю системи, оскільки ім'я є критичною інформацією для подальшої роботи з акаунтом. Результати цього тесту представлені на рисунку 4.1.

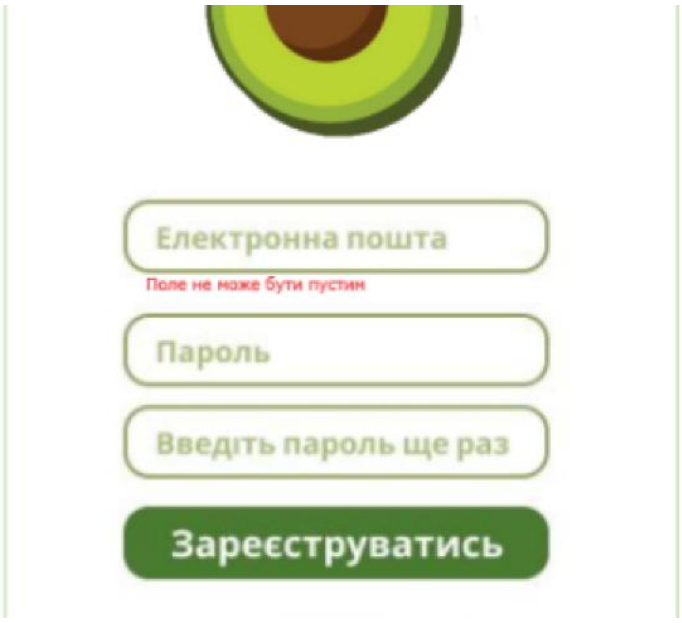

Рисунок 4.1 – Реєстрація користувача без імені

Переконавшись, що система не дозволяє реєструвати користувача без необхідних полів, спробуємо ввести неправильне значення в поле електронної

адреси. У результаті тестування отримаємо сповіщення про те, що було введено невірну електронну адресу.

Перевіривши введення не дійсних даних, спробуємо заповнити всі поля вірно і перевірити чи зареєструє система користувача. У результаті перевірки система перенаправила користувача на сторінку авторизації, отже користувача було успішно зареєстровано(див. рис. 4.2).

Рисунок 4.2 – Результат успішної реєстрації користувача

4.3 Розробка інструкції користувача

Інструкція користувача передбачає визначення технічних вимог для запуску програмного продукту. Деталі щодо мінімальної та рекомендованої конфігурації серверного комп’ютера можна знайти в таблицях 4.1 та 4.2.

Таблиця 4.1 – Мінімальна конфігурація:

Тип процесора	Exynos 9825 2,7 ГГц
Об’єм оперативної пам’яті	2 ГБ для 64-разрядної системи
Місце на жорсткому диску	1 ГБ
Операційна система	Android 9

Таблиця 4.2 – Рекомендована конфігурація:

Тип процесора	Dimensity 9000 Plus 3,2 ГГц
Об’єм оперативної пам’яті	4 ГБ для 64-разрядної системи
Місце на жорсткому диску	2 ГБ
Операційна система	Android 11

Після запуску платформи користувач повинен авторизуватися в системі та перейти на головний екран. В результаті він отримує список просканованих продуктів. Далі, йому надається можливість обрати один із продуктів. Перед переглядом складу продукту відбувається перевірка статусу користувача в системі. Якщо користувач не увійшов до системи - йому пропонується виконати вхід. Користувач також може добавити новий харчовий продукт, для чого потрібно сфотографувати сам продукт та список його інгредієнтів. Продукти, які були просканові, добавляються в головне меню, а також в список продуктів.

4.4 Висновки

У розділі було проведено всебічний аналіз методів тестування програмного забезпечення, тестування розробленого програмного продукту, а також розроблено інструкцію користувача.

Тестування розробленого програмного продукту включало застосування описаних методів для забезпечення його функціональної та нефункціональної якості. Було виконано модульне тестування для перевірки окремих компонентів,

інтеграційне тестування для перевірки їх взаємодії, системне тестування для оцінки всієї системи, а також регресійне тестування для перевірки стабільності після внесення змін. Це забезпечило виявлення та виправлення дефектів на різних етапах розробки, гарантуючи, що програмний продукт відповідає заданим вимогам.

Успішне тестування програмного продукту завершилося розробкою інструкції користувача, яка містить детальні вказівки щодо використання програмного забезпечення. Інструкція спрямована на забезпечення користувачів всім необхідним для ефективного використання продукту, включаючи опис функціональності та приклади використання. Це дозволяє користувачам швидко освоїти програму і використовувати її з максимальною ефективністю.

ВИСНОВКИ

В результаті виконання бакалаврської дипломної роботи, було розроблено програмний засіб для аналізу складу харчових продуктів. Він дозволяє користувачам отримувати детальну інформацію про харчові продукти, включаючи їх склад та поживну цінність, шляхом сканування штрих-кодів або вручного пошуку. Розроблений програмний засіб використовує передові технології обробки зображень для точного розпізнавання продуктів за їх штрих-кодами, а також має доступ до бази даних, де зберігається інформація про склад та поживну цінність продуктів.

Цей програмний засіб розроблено з метою сприяти здоровому способу життя та допомогти користувачам приймати обгрунтовані харчові рішення. Він надає можливість швидко та зручно дізнаватися про склад продуктів, допомагаючи уникнути потенційно шкідливих інгредієнтів та підтримуючи балансоване харчування.

Було проведено аналіз стану мобільних систем з використанням засобів аналізу складу харчових, порівняльний аналіз аналогів продуктів та визначено їх плюси та мінуси, аналіз методів розв’язання поставленої задачі. На їх основі було доведено доцільність та актуальність власної розробки, поставлено задачі дослідження, було визначено основні функціональні вимоги до застосунку , включаючи необхідність точного визначення складу продуктів.

Проведені етапи розробки моделі системи, алгоритмів роботи та структури проєкту заклали міцну основу для створення програмного продукту, який відповідає вимогам та забезпечує точний і надійний аналіз складу харчових продуктів.

Також було проведено варіантний аналіз та вибір мови програмування і середовища розробки. Була розроблена база даних, яка містить інформацію про харчові продукти, їх поживну цінність, склад, можливі алергени та інші важливі характеристики. Це дозволило забезпечити користувачів доступом до детальної та достовірної інформації про продукти, що сприяє усвідомленому

вибору продуктів та підтримці здорового способу життя. Був розроблений модуль сканера штрих-коду, який дозволяє користувачам отримувати інформацію про харчові продукти шляхом сканування штрих-кодів. Цей модуль забезпечує зручний та швидкий доступ до даних про продукти, що дозволяє ефективно використовувати програмний засіб та заощаджує час користувачів. Був створений інтерфейс користувача, який забезпечує легкий доступ до функціоналу програмного засобу та його зручне використання. Інтуїтивно зрозумілі елементи управління дозволяють користувачам швидко орієнтуватися та виконувати необхідні дії без зайвих зусиль.

Проведений аналіз методів тестування та тестування розробленого застосунка довели його працездатність та коректність роботи.

В рамках роботи було успішно розроблено та реалізовано програмний засіб, який надає користувачам можливість отримати детальну інформацію про харчові продукти шляхом сканування штрих-кодів або вручного пошуку. Використання передових технологій обробки зображень дозволило забезпечити точне розпізнавання продуктів, а база даних зі збереженням інформації про склад та поживну цінність продуктів забезпечує достовірність та актуальність отриманих даних.

Розроблений програмний засіб сприятиме усвідомленному вибору продуктів та підтримці здорового способу життя. Він забезпечує швидкий та зручний доступ до інформації про склад продуктів, допомагаючи уникнути потенційно шкідливих інгредієнтів та підтримуючи балансоване харчування.

Застосування мови програмування Java та середовища розробки Android Studio в процесі розробки дозволило створити ефективний та функціональний програмний продукт.

Отже, отримані результати відображають важливий внесок у сферу харчової індустрії та сприяють покращенню якості та свідомості вибору харчових продуктів у сучасному суспільстві.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. HOW DOES EATING HEALTHY AFFECT YOUR LIFE? URL: https://lifelinechiropractic.org/eating-healthy-affect-life/ (дата звернення: 13.03.2024).

2. Quality Control in Food Manufacturing: What You Need to Know. URL: https://www.deskera.com/blog/quality-control-in-food-manufacturing-what-you-need-to-know (дата звернення: 13.03.2024).

3. Щербань К.О. Актуальність розробки програмного засобу для аналізу складу харчових продуктів. *Комп'ютерна та програмна інженерія* : зб. матеріалів доп. учасн. VI Міжнародна студентська наукова конференції, м.Біла Церква, 7 червня, 2024 р. С. 223–224.

4. MyFitnessPal. URL: https://www.myfitnesspal.com/ (дата звернення: 15.03.2024).

5. Yuka. URL: http://surl.li/uipip (дата звернення: 15.03.2024).

6. Fooducate. URL: https://www.fooducate.com/ (дата звернення: 15.03.2024).

7. AllergyEats. URL: https://www.allergyeats.com/ (дата звернення: 15.03.2024).

8. ShopWell**.** URL: http://surl.li/uipja (дата звернення: 15.03.2024).

9. Introduction to web API. URL: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Introduction (дата звернення: 20.03.2024).

10. UML activity diagram. URL: https://evergreens.com.ua/ua/articles/uml-diagrams.html (дата звернення: 20.03.2017).

11. What is Python: URL: https://www.teradata.com/glossary/what-is-python (дата звернення: 04.04.2024).

12. C++ Programming Language. URL: https://www.geeksforgeeks.org/c-plus-plus/ (дата звернення: 04.04.2024).

13. JavaScript. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript (дата звернення: 04.04.2024).

14. Шилдт Г. Java 8. Керівництво для початківців: пер. з англ. М. Вільямс / Г. Шилдт – Вільямс, 2015. – 712 с.

15. IntelliJ IDEA. URL: https://www.jetbrains.com/help/idea/discover-intellij-idea.html (дата звернення: 15.04.2024).

16. What Is Eclipse? URL: https://www.eclipse.org/home/whatis/ (дата звернення: 15.04.2024).

17. Android Studio URL: https://developer.android.com/studio (дата звернення: 15.04.2024).

18. Cloud Firestore URL: https://firebase.google.com/docs/firestore (дата звернення: 22.03.2024).

19. Cloud Storage URL: https://cloud.google.com/storage?hl (дата звернення: 22.03.2024).

20. Breaking it down - EAN-13 Barcode. URL: https://www.matthews.com.au/ean-13-barcode (дата звернення: 15.04.2024).

21. UX (User Experience) URL: https://www.interaction-design.org/literature/topics/ux-design (дата звернення: 10.05.2024).

22. UI (User Interface) URL: https://www.interaction-design.org/literature/article/user-interface-design-guidelines-10-rules-of-thumb (дата звернення: 10.05.2024).

23. Unit Testing URL: https://blog.ithillel.ua/articles/unit-testy-u-java.-korotkyi-posibnyk (дата звернення: 20.05.2024).

24. Integration Testing URL: https://www.geeksforgeeks.org/software-engineering-integration-testing/ (дата звернення: 20.05.2024).

25. System Testing URL: https://www.javatpoint.com/system-testing (дата звернення: 20.05.2024).

26. UI Testing URL: https://www.hotjar.com/ui-design/testing/ (дата звернення: 20.05.2024).

27. Acceptance Testing URL: https://testsigma.com/guides/acceptance-testing/ (дата звернення: 20.05.2024).

ДОДАТКИ

64

Додаток А – Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
д.т.н., професор
_____ О. Н. Романюк
«12» березня 2024 р.

Технічне завдання
на бакалаврську дипломну роботу «Розробка програмного засобу для аналізу складу харчових продуктів» за спеціальністю 121 – Інженерія програмного забезпечення

Керівник бакалаврської дипломної роботи:
_____ д.т.н., професор Л.Б. Ліщинська
«12» березня 2024 р.
Виконав:
_____ студент гр. 3ПІ-20б К.О. Щербань
«12» березня 2024 р.

Вінниця – 2024 року

1. **Найменування та галузь застосування**

Бакалаврська дипломна робота: «Розробка програмного засобу для аналізу складу харчових продуктів».

Галузь застосування – здорове харчування.

2. Підстава для розробки.

Підставою для виконання бакалаврської дипломної роботи (БДР) є індивідуальне завдання на БДР та наказ №80 від «11» березня 2024 р. ректора ВНТУ про закріплення тем БДР.

3. Мета та призначення розробки.

Метою роботи є покращення швидкості та точності процесу аналізу складу харчових продуктів шляхом розробки мобільного застосунку.

4. Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись БДР.

1. HOW DOES EATING HEALTHY AFFECT YOUR LIFE? URL: https://lifelinechiropractic.org/eating-healthy-affect-life/ (дата звернення: 13.03.2024).

2. 9. Introduction to web API. URL: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Introduction (дата звернення: 20.03.2024).

3. Шилдт Г. Java 8. Керівництво для початківців: пер. з англ. М. Вільямс / Г. Шилдт – Вільямс, 2015. – 712 с.

4. 20. Breaking it down - EAN-13 Barcode. URL: https://www.matthews.com.au/ean-13-barcode (дата звернення: 15.04.2024).

5. Технічні вимоги

Вхідні дані — користувацькі дані, штрих-код продукту, інформація про харчові продукти.

Вихідні дані — аналіз складу продуктів, персоналізовані рекомендації,

інформація про харчову цінність

.

6. Конструктивні вимоги.

Конструкція пристрою повинна відповідати естетичним та ергономічним вимогам, повинна бути зручною в обслуговуванні та керуванні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

– пояснювальна записка до бакалаврської дипломної роботи;

– технічне завдання;

– лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки:

№ з п/п	Назва етапів бакалаврської кваліфікаційної Роботи	Строк виконання етапів роботи
1	Обґрунтування вибору методу розробки та постановка задачі дослідження	14.03.24 –22.03.24
2	Проектування бази даних	14.03.24 –22.03.24
3	Розробка модуля сканеру штрих-кодів	25.03.24 –19.04.24
4	Розробка інтерфейсу користувача	22.04.24 –10.05.24
5	Тестування застосунку	13.05.24 – 24.05.24
6	Оформлення матеріалів до захисту БДР	27.05.24 – 30.05.24

10. Порядок контролю та прийняття.

Виконання етапів бакалаврської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи. Захист бакалаврської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком

67

Додаток Б – Протокол перевірки на плагіат

ПРОТОКОЛ
ПЕРЕВІРКИ БАКАЛАВРСЬКОЇ ДИПЛОМНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Розробка програмного засобу для аналізу складу харчових продуктів

Тип роботи: БДР

Підрозділ: кафедра програмного забезпечення, ФІТКІ

Науковий керівник: Ліщинська Л.Б.

Оригінальність	94,5%
Схожість	5,5%

Аналіз звіту подібності

- **Запозичення, виявлені у роботі, оформленні коректно і не містять ознак плагіату.**

☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цілісності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховання недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Черноволик Г.О.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи _____ Щербань К.О.

Керівник роботи _____ Ліщинська Л.Б.

Додаток В – Лістинг програми

Лістинг файлу MainActivity.java

```
package com.foodscan.Activity;

import android.app.Dialog;
import android.content.Context;
import android.content.Intent;
import android.graphics.drawable.ColorDrawable;
import android.os.Bundle;
import android.support.design.widget.TabLayout;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentPagerAdapter;
import android.support.v4.app.FragmentTransaction;
import android.support.v4.view.ViewPager;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.Window;
import android.widget.FrameLayout;
import android.widget.ImageView;
import android.widget.RelativeLayout;
import android.widget.TextView;

import com.foodscan.Barcode.BarcodeGraphicTracker;
import com.foodscan.Fragment.HistoryFragment;
import com.foodscan.Fragment.LifyzerFragment;
import com.foodscan.Fragment.ProfileFragment;
import com.foodscan.R;
import com.foodscan.Utility.TinyDB;
import com.foodscan.Utility.UserDefaults;
import com.foodscan.Utility.Utility;
import com.foodscan.WsHelper.model.DTOProduct;
import com.foodscan.WsHelper.model.DTOUser;
import com.google.android.gms.vision.barcode.Barcode;
import com.rey.material.app.ThemeManager;

import java.util.ArrayList;
import java.util.List;

import io.realm.Realm;

public class MainActivity extends AppCompatActivity implements View.OnClickListener,
ViewPager.OnPageChangeListener, BarcodeGraphicTracker.BarcodeUpdateListener {

    private static final String TAG = MainActivity.class.getSimpleName();
    public FrameLayout frame_main;
    public boolean needDialog = true;
    public DTOUser dtoUser;
    public ViewPager viewPager;
    public ViewPagerAdapter adapter;
    public int offset = 0;
    public String noOfRecords = UserDefaults.REQ_NO_OF_RECORD;
    public boolean mIsLoading = false;
    public boolean isMoreData = false;
```

```
public boolean isFavLoaded = false;
public ArrayList<DTOProduct> favArrayList = new ArrayList<>();
public ArrayList<DTOProduct> historyArrayList = new ArrayList<>();
private Context mContext;
private RelativeLayout rl_history, rl_scan_food, rl_profile;
private FrameLayout frame_history, frame_scan, frame_profile;
private ImageView img_history, img_scan, img_profile;
private TextView txt_history, txt_scan_food, txt_profile;
private TinyDB tinyDB;
private Realm realm;

private TabLayout tabLayout;
private int LOGIN_REQ_CODE = 100;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    overridePendingTransition(R.anim.slide_right, R.anim.translate);

    mContext = MainActivity.this;
    tinyDB = new TinyDB(mContext);
    realm = Realm.getDefaultInstance();
    dtoUser = realm.where(DTOUser.class).findFirst();

    initView();
    initGlobals();
}

private void initView() {


    frame_main = findViewById(R.id.frame_main);

    viewPager = (ViewPager) findViewById(R.id.viewpager);
    createViewPager(viewPager);

    tabLayout = (TabLayout) findViewById(R.id.tabs);
    tabLayout.setupWithViewPager(viewPager);
    createTabIcons();

    tabLayout.setSmoothScrollingEnabled(true);
    viewPager.addOnPageChangeListener(MainActivity.this);

    int pos = 1;
    TabLayout.Tab tab = tabLayout.getTabAt(pos);
    tab.select();
    this.viewPager.setCurrentItem(pos);


}

private void initGlobals() {

    Bundle bundle = getIntent().getExtras();
    if (bundle != null) {
        needDialog = bundle.getBoolean("needDialog", false);
        if (needDialog) {
            showRegisterSuccessDialog();
```

```
            }
        }

//

    }

    @Override
    protected void onResume() {
        super.onResume();

        if (viewPager != null && adapter != null) {

            if (viewPager.getCurrentItem() == 2 &&
tinyDB.getBoolean(UserDefaults.NEED_REFRESH_USER)) {

                dtoUser = realm.where(DTOUser.class).findFirst();

                Fragment fragment = adapter.getItem(2);
                if (fragment instanceof ProfileFragment) {
                    ProfileFragment profileFragment = (ProfileFragment) fragment;
                    profileFragment.userDataChanges();
                    tinyDB.putBoolean(UserDefaults.NEED_REFRESH_USER, false);
                }
            }
        }
    }

    private void createViewPager(ViewPager viewPager) {
        adapter = new ViewPagerAdapter(getSupportFragmentManager());
        adapter.addFrag(new HistoryFragment(), getString(R.string.History));
        //adapter.addFrag(new ScanFragment(), getString(R.string.Scan_Food));
        adapter.addFrag(new LifyzerFragment(), getString(R.string.Scan_Food));
        adapter.addFrag(new ProfileFragment(), getString(R.string.Profile));

        viewPager.setAdapter(adapter);
    }

    private void createTabIcons() {

        TextView tabOne = (TextView)
LayoutInflater.from(this).inflate(R.layout.custom_tab, null);
        tabOne.setText(getString(R.string.History));
        tabOne.setCompoundDrawablesWithIntrinsicBounds(0,
R.drawable.img_history_not_selected, 0, 0);
        tabLayout.getTabAt(0).setCustomView(tabOne);

        TextView tabTwo = (TextView)
LayoutInflater.from(this).inflate(R.layout.custom_tab, null);
        tabTwo.setText(getString(R.string.Scan_Food));
        tabTwo.setCompoundDrawablesWithIntrinsicBounds(0,
R.drawable.img_scan_not_selected, 0, 0);
        tabLayout.getTabAt(1).setCustomView(tabTwo);

        TextView tabThree = (TextView)
LayoutInflater.from(this).inflate(R.layout.custom_tab, null);
        tabThree.setText(getString(R.string.Profile));
        tabThree.setCompoundDrawablesWithIntrinsicBounds(0,
R.drawable.img_profile_not_selected, 0, 0);
```

```
        tabLayout.getTabAt(2).setCustomView(tabThree);

    }

    @Override
    public void onClick(View v) {

        switch (v.getId()) {

//
        }
    }


    public void replaceFragment(Fragment fragment, int containerView) {

        try {
            FragmentTransaction transaction =
getSupportFragmentManager().beginTransaction();

            transaction.replace(containerView, fragment);
            transaction.addToBackStack("");

            transaction.commit();

        } catch (Exception e) {

            e.printStackTrace();
            Log.e(TAG, e.getMessage());
        }

    }

    private void showRegisterSuccessDialog() {

        final Dialog d = new Dialog(mContext);
        d.requestWindowFeature(Window.FEATURE_NO_TITLE);
```

Лістинг файлу CameraSource.java

```
package com.foodscan.Barcode;

import android.Manifest;
import android.annotation.SuppressLint;
import android.annotation.TargetApi;
import android.content.Context;
import android.graphics.ImageFormat;
import android.graphics.SurfaceTexture;
import android.hardware.Camera;
import android.hardware.Camera.CameraInfo;
import android.os.Build;
import android.os.SystemClock;
import android.support.annotation.Nullable;
import android.support.annotation.RequiresPermission;
import android.support.annotation.StringDef;
import android.util.Log;
```

```
import android.view.Surface;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.view.WindowManager;

import com.google.android.gms.common.images.Size;
import com.google.android.gms.vision.Detector;
import com.google.android.gms.vision.Frame;

import java.io.IOException;
import java.lang.Thread.State;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.nio.ByteBuffer;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;



@SuppressWarnings("deprecation")
public class CameraSource {
    @SuppressLint("InlinedApi")
    public static final int CAMERA_FACING_BACK = CameraInfo.CAMERA_FACING_BACK;
    @SuppressLint("InlinedApi")
    public static final int CAMERA_FACING_FRONT = CameraInfo.CAMERA_FACING_FRONT;

    private static final String TAG = "OpenCameraSource";


    private static final int DUMMY_TEXTURE_NAME = 100;


    private static final float ASPECT_RATIO_TOLERANCE = 0.01f;

    @StringDef({
        Camera.Parameters.FOCUS_MODE_CONTINUOUS_PICTURE,
        Camera.Parameters.FOCUS_MODE_CONTINUOUS_VIDEO,
        Camera.Parameters.FOCUS_MODE_AUTO,
        Camera.Parameters.FOCUS_MODE_EDOF,
        Camera.Parameters.FOCUS_MODE_FIXED,
        Camera.Parameters.FOCUS_MODE_INFINITY,
        Camera.Parameters.FOCUS_MODE_MACRO
    })
    @Retention(RetentionPolicy.SOURCE)
    private @interface FocusMode {}

    @StringDef({
        Camera.Parameters.FLASH_MODE_ON,
        Camera.Parameters.FLASH_MODE_OFF,
```

```
    Camera.Parameters.FLASH_MODE_AUTO,
    Camera.Parameters.FLASH_MODE_RED_EYE,
    Camera.Parameters.FLASH_MODE_TORCH
})
@Retention(RetentionPolicy.SOURCE)
private @interface FlashMode {}

private Context mContext;

private final Object mCameraLock = new Object();


private Camera mCamera;

private int mFacing = CAMERA_FACING_BACK;


private int mRotation;

private Size mPreviewSize;


private float mRequestedFps = 30.0f;
private int mRequestedPreviewWidth = 1024;
private int mRequestedPreviewHeight = 768;


private String mFocusMode = null;
private String mFlashMode = null;

private SurfaceView mDummySurfaceView;
private SurfaceTexture mDummySurfaceTexture;


private Thread mProcessingThread;
private FrameProcessingRunnable mFrameProcessor;


private Map<byte[], ByteBuffer> mBytesToByteBuffer = new HashMap<>();


public static class Builder {
    private final Detector<?> mDetector;
    private CameraSource mCameraSource = new CameraSource();


    public Builder(Context context, Detector<?> detector) {
        if (context == null) {
            throw new IllegalArgumentException("No context supplied.");
        }
        if (detector == null) {
```

```
                throw new IllegalArgumentException("No detector supplied.");
            }

            mDetector = detector;
            mCameraSource.mContext = context;
        }


        public Builder setRequestedPreviewSize(int width, int height) {

            if ((width <= 0) || (width > MAX) || (height <= 0) || (height > MAX)) {
                throw new IllegalArgumentException("Invalid preview size: " + width
+ "x" + height);
            }
            mCameraSource.mRequestedPreviewWidth = width;
            mCameraSource.mRequestedPreviewHeight = height;
            return this;
        }


        public Builder setFacing(int facing) {
            if ((facing != CAMERA_FACING_BACK) && (facing != CAMERA_FACING_FRONT))
{
                throw new IllegalArgumentException("Invalid camera: " + facing);
            }
            mCameraSource.mFacing = facing;
            return this;
        }


        public CameraSource build() {
            mCameraSource.mFrameProcessor = mCameraSource.new
FrameProcessingRunnable(mDetector);
            return mCameraSource;
        }
    }


    public interface ShutterCallback {

        void onShutter();
    }

        public interface PictureCallback {

        void onPictureTaken(byte[] data);
    }


    public interface AutoFocusCallback {
```

```
    void onAutoFocus(boolean success);
}

/
public interface AutoFocusMoveCallback {

    void onAutoFocusMoving(boolean start);
}


public void release() {
    synchronized (mCameraLock) {
        stop();
        mFrameProcessor.release();
    }
}


@RequiresPermission(Manifest.permission.CAMERA)
public CameraSource start() throws IOException {
    synchronized (mCameraLock) {
        if (mCamera != null) {
            return this;
        }

        mCamera = createCamera();

        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.HONEYCOMB) {
            mDummySurfaceTexture = new SurfaceTexture(DUMMY_TEXTURE_NAME);
            mCamera.setPreviewTexture(mDummySurfaceTexture);
        } else {
            mDummySurfaceView = new SurfaceView(mContext);
            mCamera.setPreviewDisplay(mDummySurfaceView.getHolder());
        }
        mCamera.startPreview();

        mProcessingThread = new Thread(mFrameProcessor);
        mFrameProcessor.setActive(true);
        mProcessingThread.start();
    }
    return this;
}

@RequiresPermission(Manifest.permission.CAMERA)
public CameraSource start(SurfaceHolder surfaceHolder) throws IOException {
    synchronized (mCameraLock) {
        if (mCamera != null) {
            return this;
        }

        mCamera = createCamera();
```

```
        mCamera.setPreviewDisplay(surfaceHolder);
        mCamera.startPreview();

        mProcessingThread = new Thread(mFrameProcessor);
        mFrameProcessor.setActive(true);
        mProcessingThread.start();
    }
    return this;
}

public void stop() {
    synchronized (mCameraLock) {
        mFrameProcessor.setActive(false);
        if (mProcessingThread != null) {
            try {

                mProcessingThread.join();
            } catch (InterruptedException e) {
                Log.d(TAG, "Frame processing thread interrupted on release.");
            }
            mProcessingThread = null;
        }


        mBytesToByteBuffer.clear();

        if (mCamera != null) {
            mCamera.stopPreview();
            mCamera.setPreviewCallbackWithBuffer(null);
            try {

                if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.HONEYCOMB) {
                    mCamera.setPreviewTexture(null);

                } else {
                    mCamera.setPreviewDisplay(null);
                }
            } catch (Exception e) {
                Log.e(TAG, "Failed to clear camera preview: " + e);
            }
            mCamera.release();
            mCamera = null;
        }
```

Лістинг файлу ScanFragment.java

```
package com.foodscan.Fragment;


import android.Manifest;
import android.annotation.SuppressLint;
```

```
import android.app.AlertDialog;
import android.app.Dialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.pm.PackageManager;
import android.graphics.drawable.ColorDrawable;
import android.hardware.Camera;
import android.net.Uri;
import android.os.Bundle;
import android.provider.Settings;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.v4.app.ActivityCompat;
import android.support.v4.app.Fragment;
import android.util.Log;
import android.view.GestureDetector;
import android.view.LayoutInflater;
import android.view.MotionEvent;
import android.view.ScaleGestureDetector;
import android.view.View;
import android.view.ViewGroup;
import android.view.Window;
import android.widget.EditText;
import android.widget.RelativeLayout;
import android.widget.TextView;
import android.widget.Toast;

import com.foodscan.Activity.MainActivity;
import com.foodscan.Activity.ProductDetailsActivity;
import com.foodscan.Interfaces.FlashLightChangeListner;
import com.foodscan.OCR.CameraSource;
import com.foodscan.OCR.CameraSourcePreview;
import com.foodscan.OCR.GraphicOverlay;
import com.foodscan.OCR.OcrDetectorProcessor;
import com.foodscan.OCR.OcrGraphic;
import com.foodscan.R;
import com.foodscan.Utility.TinyDB;
import com.foodscan.Utility.UserDefaults;
import com.foodscan.Utility.Utility;
```

```
import com.foodscan.WsHelper.helper.Attribute;
import com.foodscan.WsHelper.helper.WebserviceWrapper;
import com.foodscan.WsHelper.model.DTOProductDetailsData;
import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.GoogleApiAvailability;
import com.google.android.gms.vision.text.TextBlock;
import com.google.android.gms.vision.text.TextRecognizer;
import com.rey.material.app.ThemeManager;

import java.io.IOException;
import java.text.Normalizer;

import io.realm.Realm;

public class ScanFragment extends Fragment implements
WebserviceWrapper.WebserviceResponse, FlashLightChangeListner {


    public static final String AutoFocus = "AutoFocus";
    public static final String UseFlash = "UseFlash";
    public static final String TextBlockObject = "String";
    private static final String TAG = ScanFragment.class.getSimpleName();

    private static final int RC_HANDLE_GMS = 9001;

    private static final int RC_HANDLE_CAMERA_PERM = 2;
    private Context mContext;
    private Realm realm;

    private TinyDB tinyDB;
    private View viewFragment;
    private RelativeLayout ll_parent;
    private CameraSource mCameraSource;
    private CameraSourcePreview mPreview, preview_barcode;
    private GraphicOverlay<OcrGraphic> mGraphicOverlay;


    private ScaleGestureDetector scaleGestureDetector;
    private GestureDetector gestureDetector;
    private String productName = "";
```

```
            private TextView txt_product, txt_barcode;

            @Override
            public void onActivityResult(int requestCode, int resultCode, Intent data)
{
                super.onActivityResult(requestCode, resultCode, data);
                Log.e("!_@", "activity called");
                manageCamera();
            }

            @Override
            public void onViewCreated(View view, @Nullable Bundle savedInstanceState)
{
                super.onViewCreated(view, savedInstanceState);
            }


            @Nullable
            @Override
            public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup
container, Bundle savedInstanceState) {

                viewFragment = inflater.inflate(R.layout.scan_fragment, container,
false);
                mContext = getActivity();
                //  isOnCreateCalled = true;
                mContext = getActivity();
                realm = Realm.getDefaultInstance();
                tinyDB = new TinyDB(mContext);
                initView(viewFragment);
                initGlobals();

                return viewFragment;
            }


            private void initView(View viewFragment) {

                ll_parent = viewFragment.findViewById(R.id.ll_parent);
                mPreview                    =                    (CameraSourcePreview)
```

```
viewFragment.findViewById(R.id.preview);
            mGraphicOverlay             =             (GraphicOverlay<OcrGraphic>)
viewFragment.findViewById(R.id.graphicOverlay);

            gestureDetector      =      new      GestureDetector(mContext,      new
CaptureGestureListener());
            scaleGestureDetector    =    new    ScaleGestureDetector(mContext,    new
ScaleListener());



            txt_product = viewFragment.findViewById(R.id.txt_product);
            txt_barcode = viewFragment.findViewById(R.id.txt_barcode);

            preview_barcode = viewFragment.findViewById(R.id.preview_barcode);

        }

        private void initGlobals() {

            txt_product.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {

                    mPreview.setVisibility(View.VISIBLE);
                    preview_barcode.setVisibility(View.GONE);
                }
            });

            txt_barcode.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {

                    mPreview.setVisibility(View.GONE);
                    preview_barcode.setVisibility(View.VISIBLE);

                }
            });

            viewFragment.setOnTouchListener(new View.OnTouchListener() {
                public boolean onTouch(View v, MotionEvent e) {
```

```
                    boolean b = scaleGestureDetector.onTouchEvent(e);
                    boolean c = gestureDetector.onTouchEvent(e);
                    return b || c;

                }
            });
        }


        @SuppressLint("InlinedApi")
        private void createCameraSource(boolean autoFocus, boolean useFlash) {
            Context context = mContext;


            TextRecognizer textRecognizer = new
TextRecognizer.Builder(context).build();
            textRecognizer.setProcessor(new OcrDetectorProcessor(mGraphicOverlay));




            if (!textRecognizer.isOperational())

            {

                Log.w(TAG, "Detector dependencies are not yet available.");


                IntentFilter lowstorageFilter = new
IntentFilter(Intent.ACTION_DEVICE_STORAGE_LOW);
                boolean hasLowStorage = mContext.registerReceiver(null,
lowstorageFilter) != null;

                if (hasLowStorage) {
                    //low storage
                    Toast.makeText(mContext, R.string.low_storage_error,
Toast.LENGTH_LONG).show();
                    Log.w(TAG, getString(R.string.low_storage_error));

                }
```

```
            }


            mCameraSource =
                    new CameraSource.Builder(getActivity().getApplicationContext(),
textRecognizer)
                            .setFacing(CameraSource.CAMERA_FACING_BACK)
                            .setRequestedPreviewSize(1280, 1024)
                            .setRequestedFps(2.0f)
                            .setFlashMode(UserDefaults.isFLashLightOn             ?
Camera.Parameters.FLASH_MODE_TORCH : null)
                            .setFocusMode(autoFocus                               ?
Camera.Parameters.FOCUS_MODE_CONTINUOUS_PICTURE : null)
                            .build();
        }


        private void requestCameraPermission(boolean showSettingScreen) {
            if (showSettingScreen) {
                showErrorDialog(true);
            } else {
                final          String[]          permissions          =          new
String[]{Manifest.permission.CAMERA};
                requestPermissions(permissions, RC_HANDLE_CAMERA_PERM);
            }
        }


        private boolean onTap(float rawX, float rawY) {
            OcrGraphic graphic = mGraphicOverlay.getGraphicAtLocation(rawX, rawY);
            TextBlock text = null;
            if (graphic != null) {
                text = graphic.getTextBlock();
                if (text != null && text.getValue() != null) {


                    showTextDialog(text.getValue());


                } else {
                    Log.d(TAG, "text data is null");
```

```
                }
            } else {
                Log.d(TAG, "no text detected");
            }

            return text != null;
        }

        private void showTextDialog(String detectedText) {

            detectedText = detectedText.replaceAll("[\\t\\n\\r]+", " ");

            final Dialog d = new Dialog(mContext);
            d.requestWindowFeature(Window.FEATURE_NO_TITLE);
            d.getWindow().setBackgroundDrawable(new
ColorDrawable(android.graphics.Color.TRANSPARENT));
            d.setContentView(R.layout.dialog_detected_text);
            d.setCancelable(true);
            d.setCanceledOnTouchOutside(true);

            boolean isLightTheme = ThemeManager.getInstance().getCurrentTheme() ==
0;

            d.getWindow().getAttributes().windowAnimations = (isLightTheme ?
R.style.SimpleDialogLight : R.style.SimpleDialog);//R.style.DialogAnimation

            final EditText edt_detected_text =
d.findViewById(R.id.edt_detected_text);
            edt_detected_text.setText(detectedText);

            edt_detected_text.setSelection(edt_detected_text.getText().length());

            edt_detected_text.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    edt_detected_text.setCursorVisible(true);
                }
            });

            TextView txt_viewproduct = d.findViewById(R.id.txt_viewproduct);
            txt_viewproduct.setOnClickListener(new View.OnClickListener() {
```

```
                @Override
                public void onClick(View v) {

                    if (Utility.validateStringPresence(edt_detected_text)) {

                        productName = edt_detected_text.getText().toString();


                        productName       =       Normalizer.normalize(productName,
Normalizer.Form.NFD);
                        productName = productName.replaceAll("[^\\p{ASCII}]", "");
                        /);

                        if (((MainActivity) mContext).dtoUser != null) {
                            wsCallProductDetails();
                        } else {


                        }

                    }

                    d.dismiss();
                }
            });

            d.show();
        }

        private void displayNoResultDialog() {

            final Dialog d = new Dialog(mContext);
            d.requestWindowFeature(Window.FEATURE_NO_TITLE);
            d.getWindow().setBackgroundDrawable(new
ColorDrawable(android.graphics.Color.TRANSPARENT));
            d.setContentView(R.layout.dialog_no_result_found);
            d.setCancelable(true);
            d.setCanceledOnTouchOutside(true);

            boolean isLightTheme = ThemeManager.getInstance().getCurrentTheme() ==
0;
```

```
            d.getWindow().getAttributes().windowAnimations = (isLightTheme ?
R.style.SimpleDialogLight : R.style.SimpleDialog);//R.style.DialogAnimation

            TextView txt_ok = d.findViewById(R.id.txt_ok);
            txt_ok.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    d.dismiss();
                }
            });

            d.show();
        }


        @Override
        public void onResume() {
            super.onResume();
            Log.e("!_@_", "Onresume call");
            try {

                {
                    manageCamera();

                }
                startCameraSource();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }

        private void manageCamera() {

            boolean                          autoFocus                          =
getActivity().getIntent().getBooleanExtra(AutoFocus, true);
            boolean useFlash = getActivity().getIntent().getBooleanExtra(UseFlash,
false);
```

```
            int       rc       =       ActivityCompat.checkSelfPermission(mContext,
Manifest.permission.CAMERA);
            if (rc == PackageManager.PERMISSION_GRANTED) {
                createCameraSource(autoFocus, useFlash);
            } else {
                if                      (Utility.getPermissionStatus(getActivity(),
Manifest.permission.CAMERA) == Utility.DENIED) {
                    Log.e("!_@_", "----<<<< DENIED >>>>>-------------");
                    requestCameraPermission(true);
                } else {
                    requestCameraPermission(false);
                }
            }
        }

        /**
         * Stops the camera.
         */
        @Override
        public void onPause() {
            super.onPause();
            if (mPreview != null) {
                mPreview.stop();
            }
        }


        @Override
        public void onDestroy() {
            super.onDestroy();
            if (mPreview != null) {
                mPreview.release();
            }
        }


        @Override
        public void onRequestPermissionsResult(int requestCode,
                                               @NonNull String[] permissions,
                                               @NonNull int[] grantResults) {
```

```
            if (requestCode != RC_HANDLE_CAMERA_PERM) {
                Log.d(TAG, "Got unexpected permission result: " + requestCode);
                super.onRequestPermissionsResult(requestCode,          permissions,
grantResults);
                return;
            }

            if    (grantResults.length    !=    0    &&    grantResults[0]    ==
PackageManager.PERMISSION_GRANTED) {
                Log.d(TAG,  "Camera  permission  granted  -  initialize  the  camera
source");

                boolean                          autoFocus                          =
getActivity().getIntent().getBooleanExtra(AutoFocus, true);
                boolean                          useFlash                           =
getActivity().getIntent().getBooleanExtra(UseFlash, false);
                createCameraSource(autoFocus, useFlash);
                return;
            }
            Log.e(TAG,   "Permission   not   granted:   results   len   =   "   +
grantResults.length +
                    "   Result   code   =   "   +   (grantResults.length   >   0   ?
grantResults[0] : "(empty)"));


            showErrorDialog(false);

        }

        private void showErrorDialog(final boolean showSettingScreen) {
            DialogInterface.OnClickListener        listenerNeg        =        new
DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int id) {
    //                getActivity().finish();
                    dialog.dismiss();
                }
            };
            DialogInterface.OnClickListener        listenerPos        =        new
DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int id) {
                    Intent               intent               =               new
```

```
Intent(Settings.ACTION_APPLICATION_DETAILS_SETTINGS);
                        Uri           uri           =           Uri.fromParts("package",
getActivity().getPackageName(), null);
                        intent.setData(uri);
                        startActivityForResult(intent, 123);
                        dialog.dismiss();
                    }
                };

                if (showSettingScreen) {
                    AlertDialog.Builder builder = new AlertDialog.Builder(mContext);
                    builder.setTitle(R.string.app_name)
                            .setMessage(R.string.no_camera_permission_showSetting)
                            .setPositiveButton(R.string.ok, listenerPos)
                            .setNegativeButton(R.string.cancel, listenerNeg)
                            .show();
                } else {
                    AlertDialog.Builder builder = new AlertDialog.Builder(mContext);
                    builder.setTitle(R.string.app_name)
                            .setMessage(R.string.no_camera_permission)
                            .setNegativeButton(R.string.ok, listenerNeg)
                            .show();
                }
            }


            private void startCameraSource() throws SecurityException {
                // Check that the device has play services available.
                int                                  code                                    =
GoogleApiAvailability.getInstance().isGooglePlayServicesAvailable(
                        mContext);
                if (code != ConnectionResult.SUCCESS) {
                    Dialog dlg =
```