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

«До захисту допущено»
В.о. завідувача кафедри

_____ Світлана ВАЩЕНКО

____ _____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня магістр

зі спеціальності 122 «Комп'ютерні науки»,

освітньо-професійної програми «Інформаційні технології проектування»

на тему: Мобільний додаток підбору рецептів з використанням штучного інтелекту

Здобувача (ки) групи ІТ.м-22 Матвієнка Владислава Олександровича
(шифр групи) (прізвище, ім'я, по батькові)

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

_____ Владислав Матвієнко
(підпис) (Ім'я та ПРІЗВИЩЕ здобувача)

Керівник старший викладач кафедри ІТ к.т.н. Ольга БОЙКО _____
(посада, науковий ступінь, вчене звання, Ім'я та ПРІЗВИЩЕ) (підпис)

Суми – 2023

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

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри ІТ

Світлана ВАЩЕНКО
«_____» _____ 2023 р.

ЗАВДАННЯ

на кваліфікаційну роботу магістра студентові

Матвієнко Владислав Олександрович
(прізвище, ім'я, по батькові)

1 Тема кваліфікаційної роботи Мобільний додаток підбору рецептів з використанням штучного інтелекту

затверджена наказом по університету від «08» листопада 2023 р. № 1249-VI

2 Термін здачі студентом кваліфікаційної роботи «11» ___грудня___ 2023 р.

3 Вхідні дані до кваліфікаційної роботи інформація про рецепти

4 Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) аналіз предметної області, постановка задачі та методи дослідження, моделювання та проектування, практична реалізація мобільного додатку

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових слайдів презентації) актуальність роботи, постановка задачі, аналіз програмних продуктів – аналогів, функціональні вимоги до додатку, інструменти реалізації, IDEF0-діаграма та її декомпозиція, діаграма варіантів використання, алгоритм пошуку, практична реалізація, демонстрація додатку, висновок

6. Консультанти випускної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

Дата видачі завдання _____.

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Визначення мети проекту	07.10.2023- 07.10.2023	
2	Аналіз предметної області	08.10.2023- 08.10.2023	
3	Аналіз конкурентів	09.10.2023- 09.10.2023	
4	Формулювання цілей	10.10.2023- 10.10.2023	
5	Визначення інструментів	11.10.2023- 13.10.2023	
6	Планування WBS, OBS	14.10.2023- 17.10.2023	
7	Планування ризиків	18.10.2023- 18.10.2023	
8	Аналіз документації	19.10.2023- 21.10.2023	
9	Проектування архітектури	22.10.2023- 26.10.2023	
10	Проектування дизайну інтерфейсу користувача	27.10.2023- 31.10.2023	
11	Розробка функціоналу додатка	01.11.2023- 19.11.2023	
12	Розробка інтерфейсу користувача	20.11.2023- 30.11.2023	
13	Тестування	01.12.2023- 02.12.2023	
14	Завершення	03.12.2023- 04.12.2023	

Магістрант _____ Владислав Матвієнко

Керівник роботи _____ к.т.н. Ольга БОЙКО

АНОТАЦІЯ

Тема кваліфікаційної роботи магістра «Мобільний додаток підбору рецептів з використанням штучного інтелекту».

Пояснювальна записка складається зі вступу, 4 розділів, висновків, списку використаних джерел із 40 найменувань та 2 додатків. Загальний обсяг роботи − 80 сторінок, у тому числі 41 сторінки основного тексту, 4 сторінки списку використаних джерел, 29 сторінок додатків, 4 таблиці, 36 рисунків, 1 формула.

Актуальність роботи полягає в тому, що щодня виникають нові ідеї та рецепти, а мобільні додатки можуть відіграти важливу роль у структуруванні цього гастрономічного безладу та забезпеченні швидкого доступу до необхідних знань, витрачаючи на це набагато менше часу. Це не тільки спрощує процес пошуку потрібних рецептів, але й дозволяє користувачам експериментувати з новими стравами, наприклад, використовуючи продукти, які вже є на їхній кухні. Таким чином, даний мобільний додаток для пошуку рецептів можуть стати незамінним помічником для любителів кулінарії.

Мета роботи: розробка мобільного додатка підбору рецептів з використанням штучного інтелекту. Результатом роботи є повноцінно функціонуючий мобільний додаток.

Ключові слова: мобільний додаток, Android, штучний інтелект, TF-IDF Flutter, Python.

ЗМІСТ

ВСТУП

У сучасному світі, де швидкість і зручність є ключовими факторами, мобільні пристрої, зокрема смартфони, стали не тільки невід'ємною частиною нашого повсякденного життя, але й ключовим інструментом пошуку інформації. Завдяки їм ми не лише спілкуємося та взаємодіємо один з одним, але й отримуємо доступ до різноманітних сервісів.

Актуальність. Важливість створення мобільних додатків продовжує зростати щороку. Кожен рік виходить сотні, а то й тисячі нових додатків. Згідно даних App Annie [1] , кількість нових мобільних додатків, опублікованих в App Store і Google Play Store, збільшилася з 600 тис у 2011 році до 150 млн у 2023 році. Це також стосується мобільних додатків, пов’язаних з гастрономією та кулінарією.

У сфері технологічного прогресу інтеграція штучного інтелекту (ШІ) стала ключовим фактором у формуванні інноваційних рішень для вирішення різних задач нашого життя [2]. Світ, де час є дорогоцінним ресурсом, а кулінарне розмаїття розширюється, потреба в інтуїтивному, керованому штучним інтелектом рішенні для спрощення процесу вибору рецептів стає все більш нагальною.

Щодня виникають нові ідеї та рецепти, а мобільні додатки можуть відіграти важливу роль у структуруванні цього гастрономічного безладу та забезпеченні швидкого доступу до необхідних знань, витрачаючи на це набагато менше часу. Це не тільки спрощує процес пошуку потрібних рецептів, але й дозволяє користувачам експериментувати з новими стравами, наприклад, використовуючи продукти, які вже є на їхній кухні. Таким чином, мобільні додатки для пошуку рецептів можуть стати незамінним помічником для любителів кулінарії [3].

Об’єкт дослідження: процес пошуку рецептів по запропонованим інгредієнтам.

Предмет дослідження: алгоритм пошуку рецептів з використанням методів штучного інтелекту.

Мета даної роботи – це розробка мобільного додатка підбору рецептів з використанням штучного інтелекту.

Практичне цінність: можливість отримати рекомендовані рецепти по наявним інгредієнтам на кухні використовуючи алгоритм пошуку, це зменшить час для знаходження рецептів.

Для досягнення заданої мети необхідно вирішити такі задачі:

– проаналізувати предметну область, виконати огляд останніх публікацій і досліджень в цьому питанні, провести огляд програмних продуктів-аналогів;

– запропонувати алгоритм підбору рецептів з використанням методів штучного інтелекту;

– спроектувати архітектуру мобільного додатка та створити структурно-функціональну модель;

– виконати програмну реалізацію мобільного додатку та розробити інтерфейс користувача.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд останніх досліджень і публікацій

У світі, де сучасний темп життя вимагає швидких рішень та ефективного використання часу, приготування їжі стає справжнім викликом. З огляду на зростаючий інтерес до кулінарії та різноманітність страв, користувачі часто стикаються з проблемою вибору та пошуку рецептів, особливо з урахуванням індивідуальних уподобань та наявності певних інгредієнтів на кухні [4].

Аналіз літератури в галузі кулінарії та штучного інтелекту свідчить про прагнення вирішити подібні проблеми та запропонувати користувачам інноваційні рішення в пошуках потрібних рецептів. В статті [5] авторами обговорюються основні методології та переважні техніки в рекомендаційних системах та те, як штучний інтелект може ефективно покращити технологічний розвиток та застосування рекомендаційних систем.

В іншій статті детальніше розповідається про модель Convolution Neural Network (CNN), яка застосовується для ідентифікації інгредієнтів їжі, а для рекомендацій рецептів використовується машинне навчання [6].

Ряд деяких простих та складних алгоритмів, таких як, колаборативна фільтрація, контент-орієнтована фільтрація, гібридна система, асоціативні правила, машинне навчання та глибоке навчання, можуть використовуються для створення рекомендацій [7].

У створенні рекомендацій часто використовують метод TF-IDF, наприклад, для рекомендацій новин. TF-IDF надає ваги кожному слову в заголовку новини, а потім шукає схожість між історіями, використовуючи косинусну схожість [8].

Ці джерела надають важливий контекст та інформацію про використання штучного інтелекту для рекомендацій рецептів, особливо з урахуванням інгредієнтів, які має користувач. Вони можуть слугувати хорошою відправною точкою для додаткового дослідження та розробки мобільного додатку.

1.2 Аналіз існуючих аналогів

Наразі існує доволі багато додатків і платформ, які використовують штучний інтелект для підбору рецептів і дозволяють користувачам знаходити рецепти за наявними інгредієнтами.

Перший аналог “SuperCook”, який є популярним на просторах інтернету [9]. SuperCook - це платформа, яка революціонізує спосіб приготування їжі, надаючи користувачам ефективні та креативні варіанти, засновані на інгредієнтах, які вони мають під рукою. Користувач вказує інгредієнти, які він має в себе дома, SuperCook за допомогою алгоритмів штучного інтелекту аналізує ваші інгредієнти та виводять оптимальні рецепти із доступних сайтів.

На рисунку 1.1 зображено інтерфейс та приклад використання платформи.

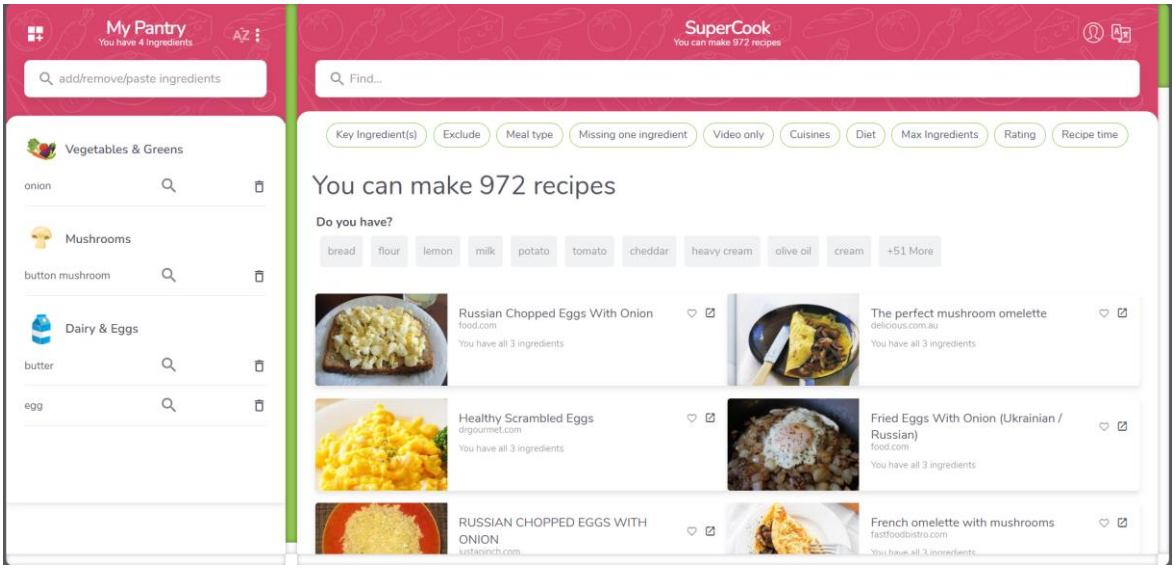

Рисунок 1.1 – Платформа “SuperCook“

Джерело: побудовано автором (знімок з екрану)

Наступний аналог веб-додаток “Recipe Radar” [10]. За функціональністю він дуже схожий на попередній, але також можна вказати, які інгредієнти не слід використовувати у страві. Користувацький інтерфейс дуже відстає від попереднього розглянутого аналога.

На рисунках 1.2 і 1.3 зображено інтерфейс та приклад використання веб-додатку.

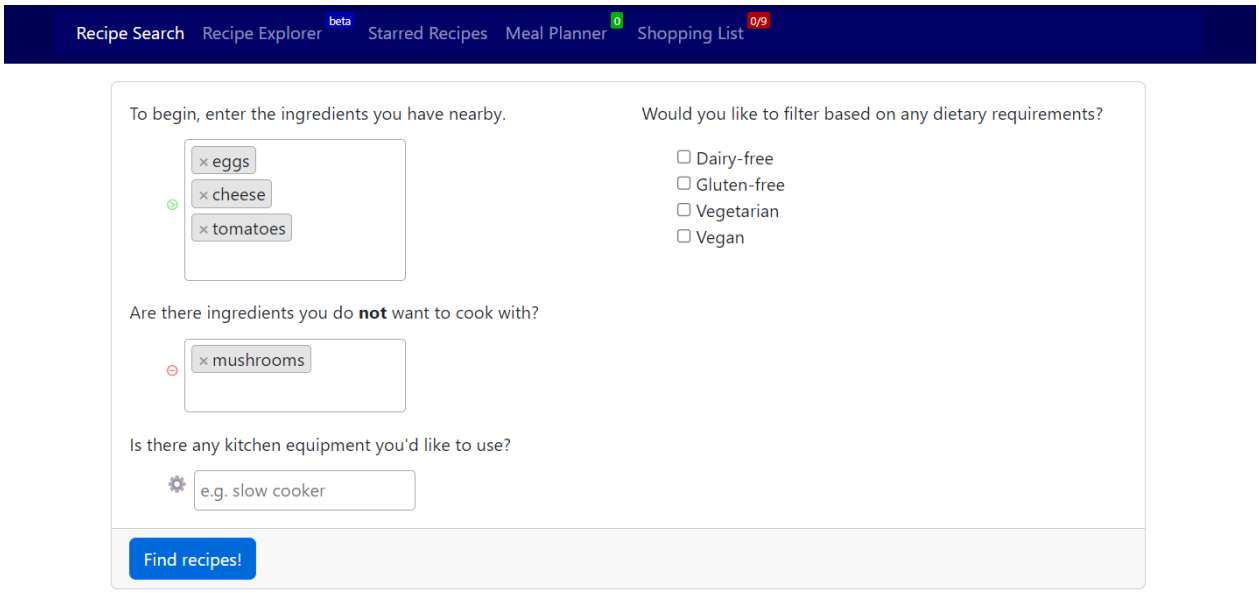

Рисунок 1.2 – Веб-додаток “Recipe Radar“

Джерело: побудовано автором (знімок з екрану)

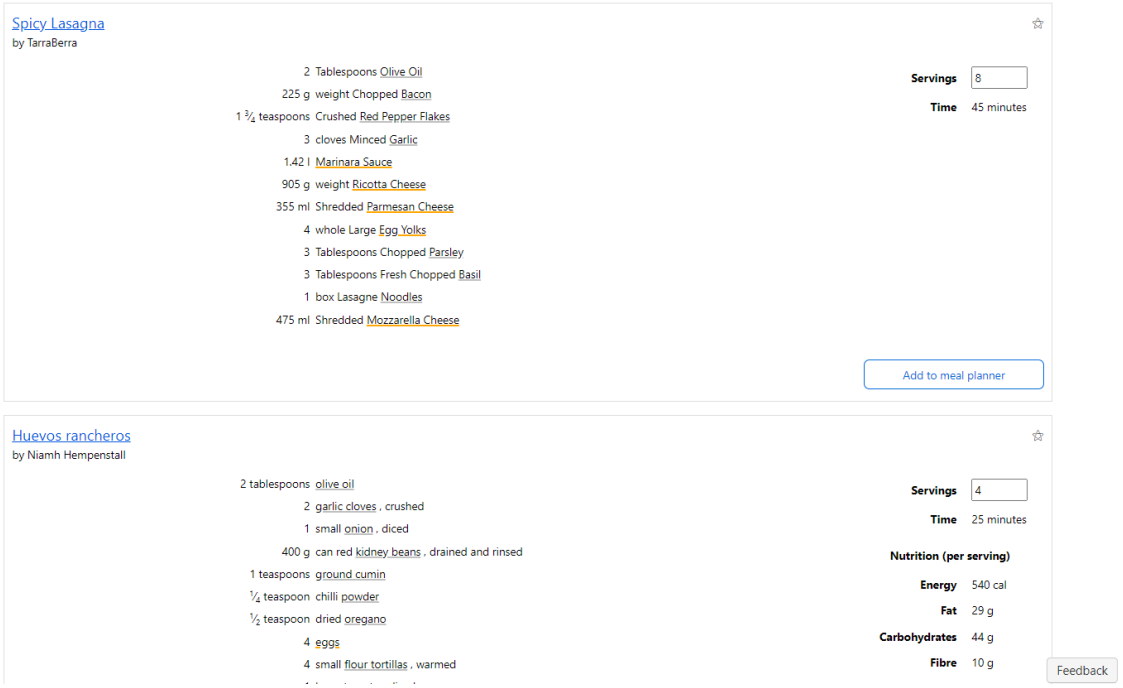

Рисунок 1.3 – Веб-додаток “Recipe Radar“

Джерело: побудовано автором (знімок з екрану)

Ще один веб-сайт на який можна звернути увагу - “Tesco Real Food“ [11]. Tesco Real Food - це веб-сайт, що пропонує різноманітні рецепти для користувачів. Він містить рецепти на будь-яку нагоду, від сніданків до вечері, від простих страв до святкових блюд.

На сайті “Tesco Real Food“ можна знайти рецепти, які можна приготувати за 30 хвилин, рецепти для повільного кулінарії, а також рецепти для приготування страв з продуктів, які потрібно використати.

Відгуки користувачів про “Tesco Real Food“ варіюються. Деякі користувачі високо оцінюють якість рецептів та великий вибір, тоді як інші вказують на проблеми з обслуговуванням клієнтів.

Додаток здійснює пошук тільки на своєму веб-сайті серед рецептів, які були додані користувачами.

На рисунку 1.4 зображено інтерфейс та приклад використання веб-додатку.

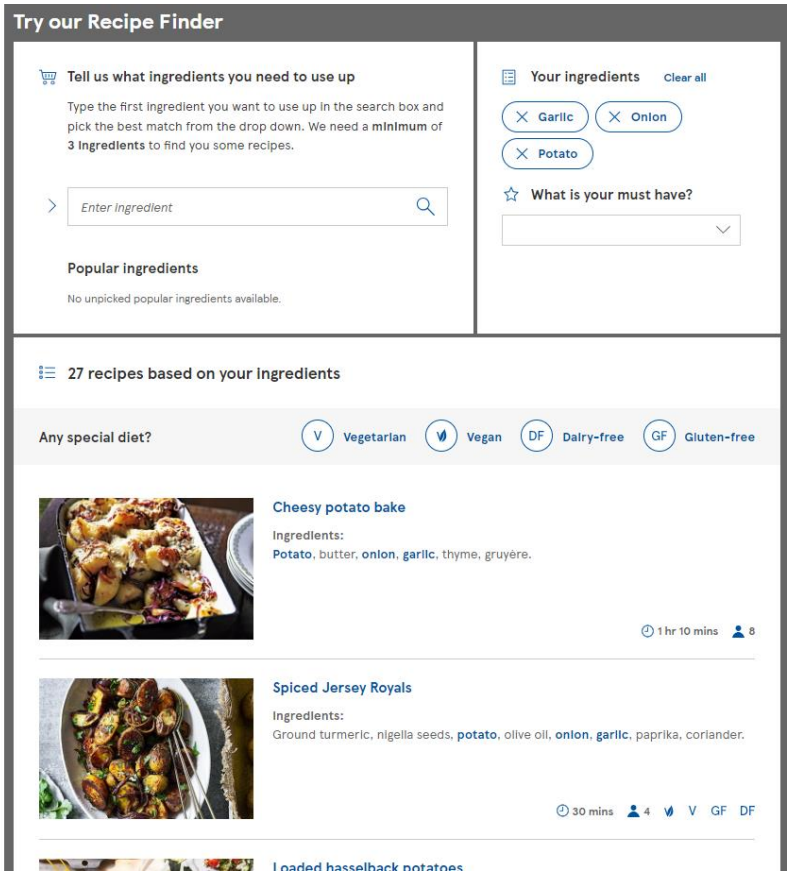

Рисунок 1.4 – Веб-додаток “Tesco Real Food“

Джерело: побудовано автором (знімок з екрану)

З мобільних додатків можна привести “Pantry Chef” [12]. Доволі простий додаток, який, як і попередні аналоги, здійснює пошук рецептів по заданим інгредієнтам. Більш серйозної функціональності не має. Рецепт можна переглянути, але функції зберегти його на потім не має.

На рисунках 1.5 та 1.6 зображено інтерфейс та приклад використання мобільного додатка.

Рисунок 1.5 – Мобільний додаток “Pantry Chef”

Джерело: побудовано автором (знімок з екрану)

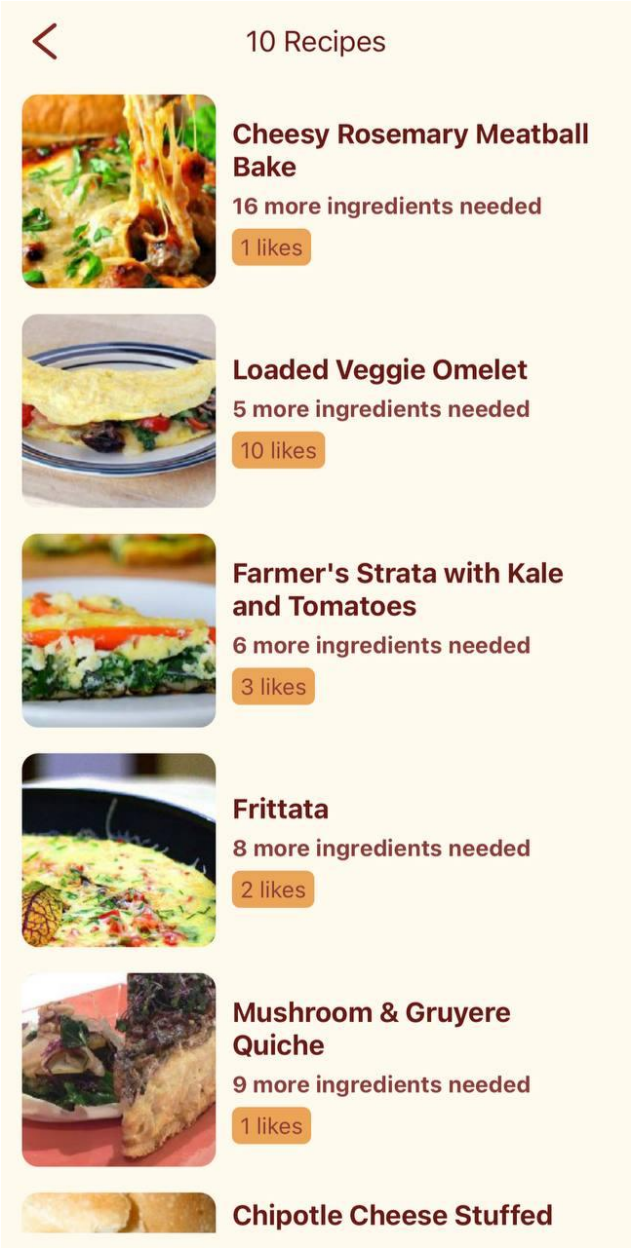

Рисунок 1.6 – Мобільний додаток “Pantry Chef”

Джерело: побудовано автором (знімок з екрану)

Також до аналогів можна привести чат-бот “ChatGPT”, який може допомогти з пошуком рецептів по інгредієнтах, надаючи конкретні поради та рецепти на основі ваших уподобань та доступних інгредієнтів [13].

На рисунках 1.7 і 1.8 зображено приклад застосування чат-боту.

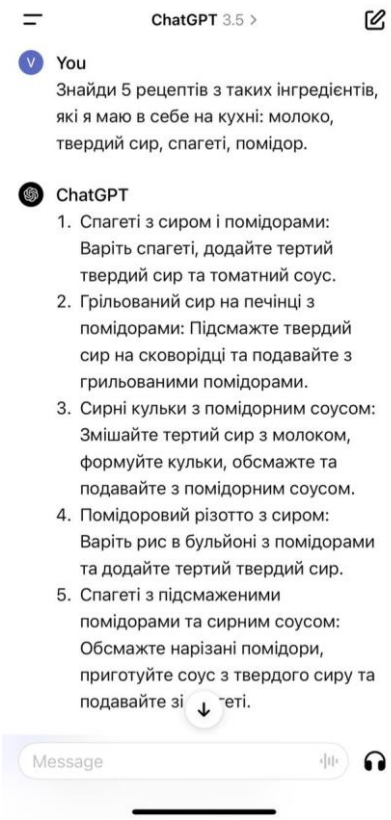

Рисунок 1.7 – Чат-бот “ChatGPT”

Джерело: побудовано автором (знімок з екрану)

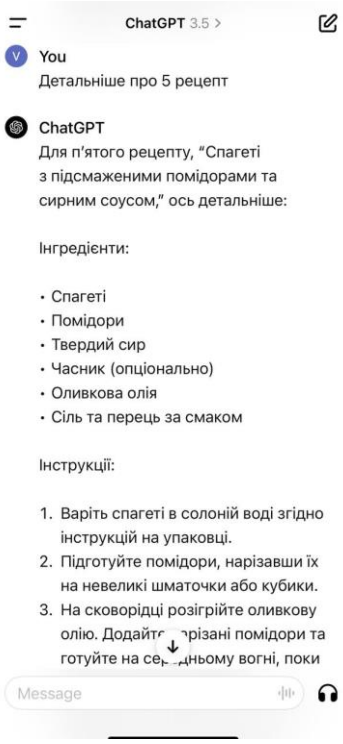

Рисунок 1.8 – Чат-бот “ChatGPT”

Джерело: побудовано автором (знімок з екрану)

У версії 4 було додано можливість додавання файлів та зображень [14]. Це дозволяє чат-боту аналізувати зміст зображення, наприклад, на основі фотографії інгредієнтів, він може пропонувати відповідні рецепти. Однак, час від часу чат-бот може допускати помилки, надавати некоректну інформацію або створювати власні рецепти, які не були перевірені на практиці. Використання чат-бота великою кількістю користувачів може створювати значне навантаження на сервери.

Порівняння функціоналу сервісів для пошуку рецептів між собою та у порівнянні з додатком котрий створюється під час роботи над магістерською роботою, зображено у таблиці 1.1

Таблиця 1.1 – Порівняння функціоналу аналогів з власною розробкою

Джерело: побудовано автором

Параметри	Назва сервісу					
	SuperCook	Recipe Radar	Tesco Real Food	Pantry Chef	ChatGPT	Власний мобільний додаток
Дизайн	+	-	-	+	+	+
Зручність використання	-	-	-	+	+	+
Ціна	Безкоштовно	Безкоштовно	Безкоштовно	Безкоштовно	Безкоштовно/ $20/місяць	Безкоштовно
Велика кількість підібраних рецептів	+	+	-	-	-	+
Використання інструментів штучного інтелекту (машинного навчання) для пошуку	+	+	-	-	+	+
Використання камери для розпізнавання інгредієнтів	-	-	-	-	+	+

В результаті аналізу предметної області та аналогів можна зробити висновок про актуальність розробки мобільного додатку підбору рецептів з використанням штучного інтелекту. Розроблюваний додаток повинен мати переваги, які не мають аналоги, для розпізнавання інгредієнтів через камеру пристрою використовувати бібліотеку для розпізнавання, без необхідності писати всі інгредієнти самому, здійснюватиме пошук використовуючи машинне навчання для знаходження рецептів від більш точних до менш.

2 ПОСТАНОВКА ЗАДАЧІ ТА МЕТОДИ ДОСЛІДЖЕННЯ

2.1 Мета та задачі дослідження

Об’єктом дослідження є процес пошуку рецептів по запропонованим інгредієнтам.

Предметом даного дослідження є алгоритм пошуку рецептів з використанням методів штучного інтелекту.

Метою цієї роботи є створення мобільного додатку для підбору рецептів з використанням штучного інтелекту. Головною функцією стане можливість сканування інгредієнтів за допомогою камери з подальшим пошуком рецептів.

Для досягнення заданої мети необхідно вирішити такі задачі:

– проаналізувати предметну область, виконати огляд останніх публікацій і досліджень в цьому питанні, провести огляд програмних продуктів-аналогів;

– запропонувати алгоритм підбору рецептів з використанням методів штучного інтелекту;

– спроектувати архітектуру мобільного додатка та створити структурно-функціональну модель;

– виконати програмну реалізацію мобільного додатку та розробити інтерфейс користувача.

Додаток повинен обробляти запит користувача і мати можливість видавати рекомендації з подальшим виведенням результату у вигляді списку рецептів.

У результаті було окреслено такі вимоги до мобільного додатку:

− пошук рецептів по наявним інгредієнтам;

− сканування продуктів для подальшого пошуку рецептів;

− можливість вибору альтернативних інгредієнтів із доступного списку у випадку їх відсутності у основній виборці;

− безпосередній перегляд рецептів;

− додавання рецепту до списку улюблених;

– оновлення списку рецептів, якщо вони не задовільнять вимогам користувача.

Практичне значення роботи є можливість отримати рекомендовані рецепти по наявним інгредієнтам на кухні використовуючи алгоритм пошуку, це зменшить час для знаходження рецептів.

2.2 Методи дослідження

Після того, як було проведено мету та задачі дослідження, потрібно визначити технології для реалізації мобільного додатка.

Сканування інгредієнтів повинно здійснюватися за допомогою платформи Vision AI [15] та VisionAPI [16], яка зосереджена на наданні машинам здатності “бачити” та розуміти вміст зображень або відео. Це дозволить без зайвих зусиль отримати дані про інгредієнти на зображенні з подальшим їх обробленням.

Аналіз дата сетів відбувається за допомогою бібліотеки Pandas [16] на окремому мікросервісі, який побудований на фреймворку Flask [18], а пошук по потрібним інгредієнтам здійснюється за допомогою TF-IDF, який є інструментом бібліотеки scikit-learn, вона має прості та ефективні інструменти для прогнозного аналізу даних [19].

Інтерфейс мобільного додатку буде створено за допомогою фреймворку Flutter від компанії Google, який призначений для створення крос-платформенних мобільних застосунків. Основною метою Flutter є розробка красивих та високопродуктивних додатків, які можуть працювати як на iOS, так і на Android, а також на веб-платформі [20].

Існує багато способів отримання даних, і їх використання залежить від галузі дослідження, типу та типу необхідної інформації, доступного часу та інших ресурсів. У більшості випадків цей процес супроводжується величезними

витратами ресурсів, особливо часових і фінансових. Отже, у цьому випадку у якості даних, будуть використані дані із відкритих джерел [21].

Набір даних містить CSV-файл і заархівовану папку, що складається з 13 582 зображень і рядків відповідно.

CSV-файл містить 5 стовпців, а саме:

- назва, є назвою страви;
- інгредієнти, містить інгредієнти в тому вигляді, в якому вони були взяті з веб-сайту;
- інструкції, містить покрокову інструкцію з рецепту, яких слід дотримуватися, щоб відтворити страву;
- назва картинки, містить ім'я зображення, яке зберігається в архівованій папці;
- очищені інредієнти: Містить інгредієнти після обробки та очищення.

Перед використання набір даних було детальніше перевірено, пусті дані було очищено.

Якщо детальніше про здійснення пошуку, то для реалізації такої системи пошуку рецептів можна застосувати різні алгоритми та статистичні методи, такі як: GloVe[22], Word2Vec[23], Doc2Vec[24], TF-IDF та інші [25]. Для даної ж роботи був обраний саме статистичний метод TF-IDF, який використовується для визначення важливості слова в документі в контексті корпусу текстів [26]. Цей метод широко використовується в обробці природної мови та серед інформаційного пошуку[27]. Метод TF-IDF є досить ефективним і популярним методом тому він і був обраний для цього завдання.

2.2.1 Алгоритм TF-IDF

Алгоритм TF-IDF працює шляхом визначення відносної частоти слів у конкретному документі порівняно зі зворотною пропорцією цього слова у всьому корпусі документів [28]. Зрозуміло, що цей розрахунок повинен визначати, наскільки релевантним є дане слово в конкретному документі. Слова, які часто

зустрічаються в одному або невеликій групі документів, як правило, мають вищі значення TF-IDF, ніж такі поширені слова, як артиклі та прийменники.

Формальна процедура застосування TF-IDF має деякі незначні відмінності у всіх його застосуваннях, але загальний підхід працює наступним чином, як зображено на формулі 2.1:

$$W_{t,d} = tf_{t,d} * \log(\frac{N}{df_t}) \quad (2.1)$$

де частота терміну моделює частоту терміну *t* у документі *d*. Обернена частота документа вимірює, скільки інформації надає термін у документі, і обчислюється як логарифм кількості документів у корпусі *N*, поділений на загальну кількість документів, що містять *t* [29].

У ході даної роботи використовувалась бібліотека scikit-learn, яка надає доступ до математичних функцій, включаючи метод пошуку TF-IDF. Для виконання цього методу на вхід подавався набір даних із рецептами та перелік наявних інгредієнтів. Результат роботи методу включає в себе рецепти, які найбільш відповідають вказаним інгредієнтів.

3 МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ

3.1 Структурно-функціональне моделювання процесу рекомендацій рецептів

Наступним етапом є виконання структурно-функціонального моделювання та проектування мобільного додатку. Для цього були створені діаграми в нотації IDEF0 з описом послідовностей процесу пошуку рецептів.

3.1.1 Діаграми нотації IDEF0

Діаграма IDEF0 - це графічний метод моделювання бізнес-процесів, який використовується для аналізу і проектування систем та їх функцій [30].

Основна мета діаграм IDEF0 - надати структуроване уявлення про функції системи або організації, а також визначити їхні зв'язки та інтерфейси. Це практичний інструмент для визначення обов'язків, функцій, входів і виходів у процесі, а також для визначення послідовності операцій.

Діаграма IDEF0 включає в себе ряд наступних компонентів:

– Функції: це основні дії або процеси, які відбуваються в системі. Вони зображуються у вигляді прямокутників та містять назву функції та унікальний номер.

– Стрілки: це лінії, які з’єднують функції та представляють потоки даних або матеріалів. Вони можуть бути вхідними, вихідними, механізмами або контролюючими.

– Вхідні дані: це дані або ресурси, які використовуються функцією.

– Вихідні дані: це результати або продукти функції.

– Механізми: це ресурси, які використовуються для виконання функції, але які не змінюються в процесі.

– Контролюючі: це елементи, які визначають, коли та як виконується функція [31].

Для мобільного додатку контекстна діаграма IDEF0 виглядає як на рисунку 3.1.

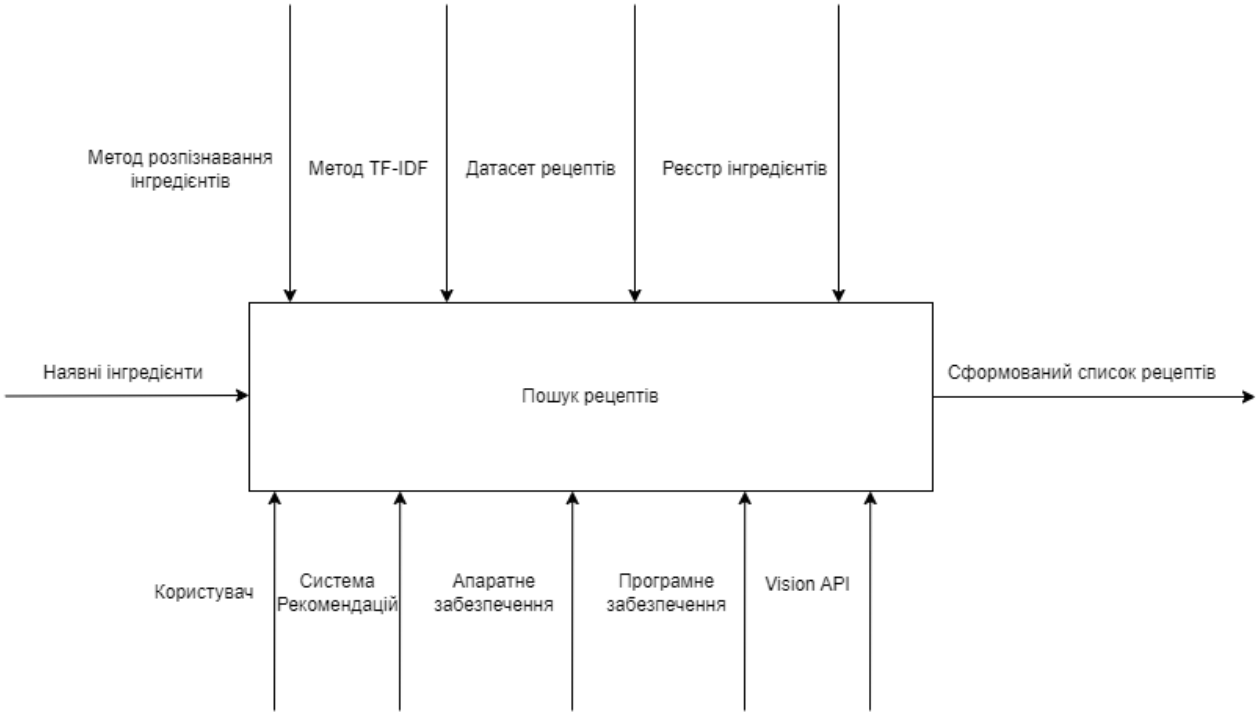

Рисунок 3.1 - Контекстна діаграма IDEF0

Джерело: побудовано автором

Тобто процес «Формування рекомендації рецептів» містить такі дані:

– вхідні дані – наявні інгредієнти;

– вихідні дані – сформований рекомендаційний список рецептів.

– механізми – сам користувач, апаратне та програмне забезпечення, система рекомендацій, платформа Vision API;

– контролюючі дані – метод пошуку TF-IDF, метод розпізнавання інгредієнтів, датасет рецептів та реєстр інгредієнтів.

Аналізуючи дану діаграму, ми можемо визначити лише взаємозв’язок процесів у загальному контексті, відкидаючи непотрібні подробиці. Щоб

детальніше описати логіку та послідовність роботи, модель варто розкласти на складові, тобто декомпозувати [32].

Діаграма декомпозиції показана на рисунку 3.2.

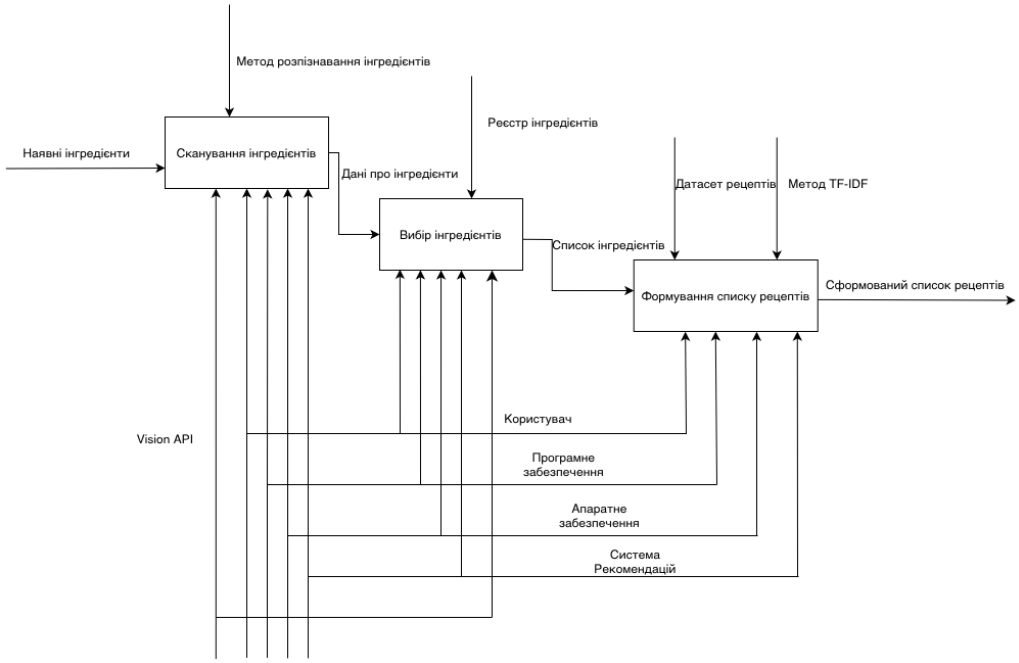

Рисунок 3.2 – Діаграма декомпозиції

Джерело: побудовано автором

Як видно з рисунку декомпозиція системи відбувається на три головних процеси:

- сканування інгредієнтів;
- вибір інгредієнтів;
- формування списку рецептів.

3.2 Моделювання варіантів використання

Діаграма Use Case - це графічне зображення можливих взаємодій користувача з системою. Вона показує різні випадки використання та різні типи користувачів системи, і часто супроводжується іншими типами діаграм.

Вона є основною формою системних/програмних вимог для нової програми, що розробляється. Вона визначає очікувану поведінку (що), а не точний метод її реалізації (як). Вона показує взаємозв'язки між випадками використання, акторами та системами [33].

Нижче відображено список сценаріїв використання ним системи:

- сканувати інгредієнти;
- переглядати рекомендації рецептів на основі інгредієнтів;
- переглядати окремий рецепт;
- зберігати рецепт до улюблених;
- переглядати улюблені рецепти.

Розроблена діаграма Use Case на основі списку сценаріїв представлена на рисунку 3.3.

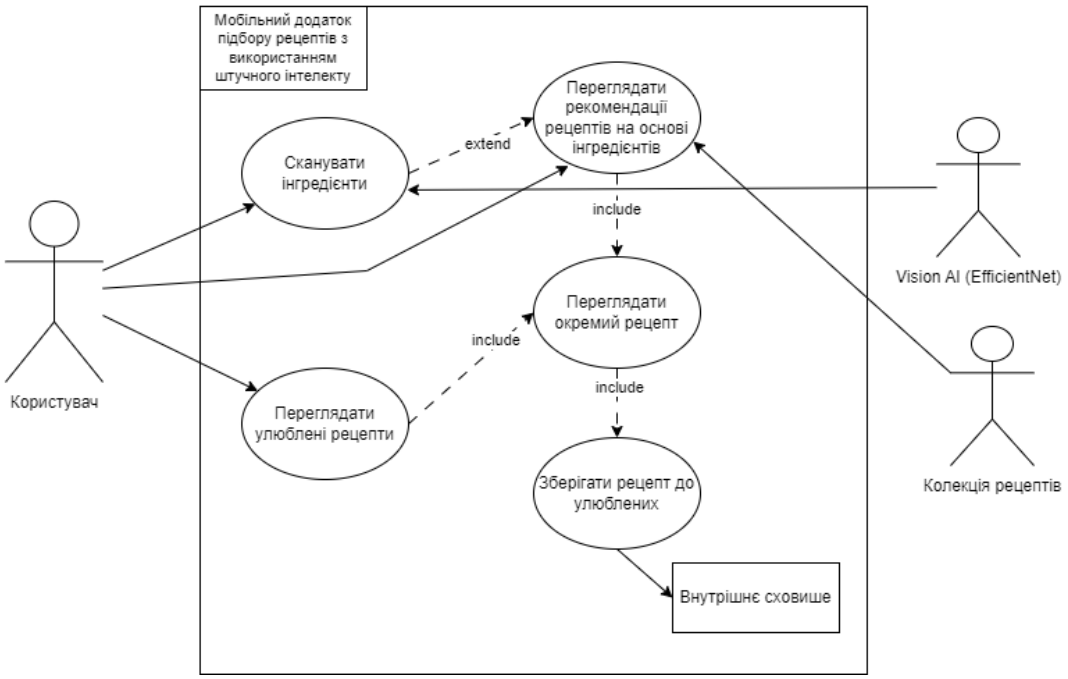

Рисунок 3.3 – Діаграма Use Case

Джерело: побудовано автором

Це допомагає проектувати систему з точки зору кінцевого користувача. Діаграма не показує деталі випадків використання: вона лише узагальнює деякі з відносин між випадками використання, акторами та системами [34].

3.3 Архітектура додатку

На рисунку 3.4 зображено архітектуру додатку підбору рецептів з використанням штучного інтелекту.

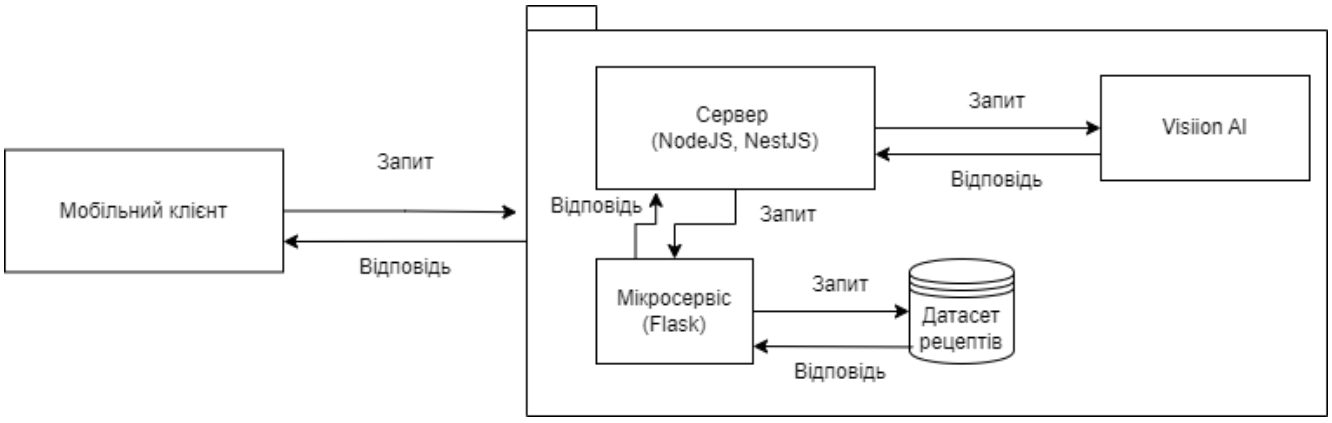

Рисунок 3.4 – Архітектура додатку

Джерело: побудовано автором

Додаток має мобільний інтерфейс, який за допомогою Rest запитів звертається до основного серверу, а той у свою чергу до мікросервісу, який виконує алгоритм пошуку в датасеті рецептів і передає підібрані рецепти вище та до Vision AI для сканування інгредієнтів.

4 РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ

4.1 Програмна реалізація

Додаток реалізований з використанням Flutter, що є фреймворком для розробки крос-платформених мобільних додатків мови програмування Dart. Flutter надає зручні інструменти для створення естетичного та ефективного інтерфейсу, а також дозволяє ефективно працювати серверними запитами. Додаток має наступні ключові функції:

- інтерфейс користувача: Інтуїтивно зрозумілий інтерфейс для користувача;
- сканування інгредієнтів: Використання камери для сканування інгредієнтів за допомогою машинного зору та обробки зображень, щоб швидко ідентифікувати та додавати їх до списку;
- пошук рецептів за інгредієнтами: Взаємодія з серверною частиною для отримання персоналізованих рецептів з урахуванням обраних інгредієнтів та вподобань користувача.

Для мобільного додатку була реалізована допоміжна серверна частина, розроблена з використанням Node.js, який є ефективним та масштабованим середовищем для розробки серверних додатків. Вона буде посередником між мобільний додатком та мікросервісом з алгоритмом пошуку. Основні функції серверної частини детальніше:

- обробка запитів від мобільного додатку: Прийняття та обробка запитів від клієнтського додатку з використанням RESTful API [35];
- комунікація з Vision API: Отримання результатів сканування продуктів;
- комунікація з мікросервісом для алгоритму пошуку рецептів: Відправка запитів до мікросервісу для отримання рекомендованих рецептів.

Також створено невеликий мікросервіс для реалізації алгоритму пошуку рецептів, щоб відгородити його від основного сервера, написаний на Flask, легкому

та гнучкому фреймворку для розробки веб-додатків на мові програмування Python. Основні функції мікросервісу включають:

– алгоритм пошуку рецептів: Реалізація методу TF-IDF для пошуку рецептів на основі інгредієнтів;

– відповіді на запити від серверної частини: Обробка запитів для пошуку та повернення персоналізованих рецептів;

– оптимізація швидкості та відповіді: Забезпечення оптимальної швидкості відповіді на запити для поліпшення досвіду користувача.

На рисунках 4.1 показано набір файлів, використаних при розробці мобільного додатку.

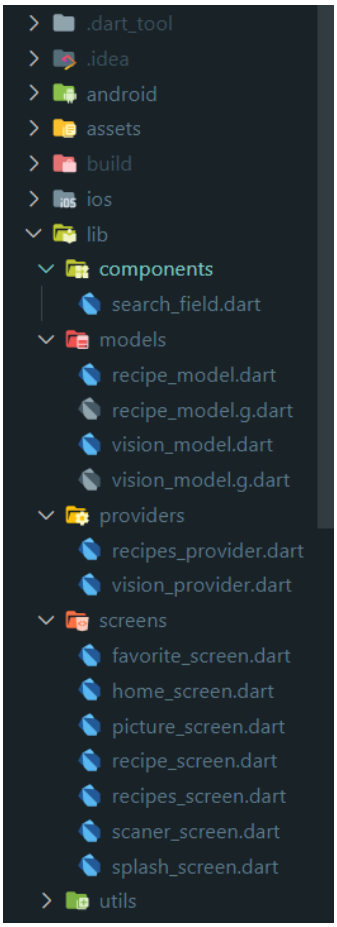

Рисунок 4.1 – Файли мобільного додатку

Джерело: побудовано автором (знімок з екрану)

Папка android містить усі специфічні для Android файли та конфігурації, включаючи також AndroidManifest.xml, файли збірки Gradle та інші необхідні ресурси.

Каталог ios містить усі файли, що стосуються iOS, у тому числі файл Info.plist, проект Xcode та інші ресурси.

У каталозі lib міститься код Dart, який є основною функціональністю додатку. Головна точка входу є main.dart. У цьому каталозі створюються власні віджети, екрани та логіку.

Каталог models містить моделі об’єктів рецепту та інформації розпізнаного зображення.

Каталог components містить окремі компоненти додатку, наприклад, поле пошуку.

Каталог screens містить екрани додатку, загальну картину яку бачить користувач під час використання додатку.

У каталозі utils містяться основні налаштування додатку.

Каталог assets використовується для зберігання статичних файлів, таких як зображення, шрифти та дані JSON, які можуть знадобитися додатку.

Розробка інтерфейсу на Flutter включає створення UI з використанням віджетів. Все у Flutter є віджетом. Віджети описують, яким повинен бути їх вигляд, враховуючи їх поточну конфігурацію та стан. Ось декілька основних віджетів, які часто використовуються в додатку:

– Text: Потрібен для створювання послідовності стилізованого тексту в межах додатку.

– Row, Column: Ці гнучкі віджети дозволяють створювати гнучкі макети як в горизонтальному (Row), так і в вертикальному (Column) напрямках.

– Stack: Замість лінійної орієнтації (або горизонтальної, або вертикальної), віджет Stack дозволяє розміщувати віджети один поверх одного в порядку розміщення.

– Container: Віджет Container дозволяє створювати прямокутний візуальний елемент, є елементарною одиницею.

На рисунку 4.2 показані створені власні віджети для додатку, які містяться в каталозі widgets.

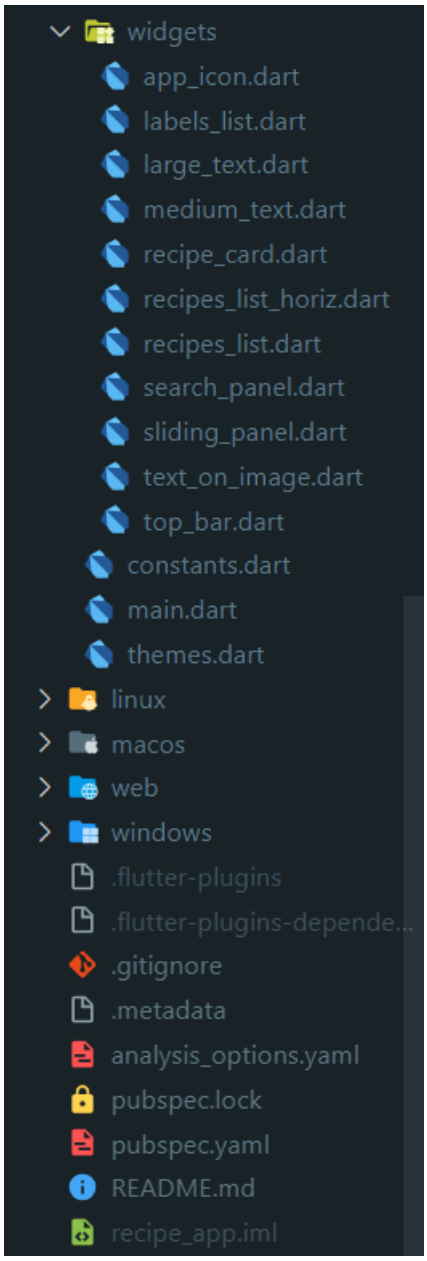

Рисунок 4.2 – Файли мобільного додатку

Джерело: побудовано автором (знімок з екрану)

Каталоги linux, macos, web та windows потрібні для детальнішого налаштування додатку для різних пристроїв.

Файл pubspec.yaml є життєво важливим компонентом додатку.. Він розташований у кореневому каталозі та використовується для керування залежностями проекту, визначення метаданих про вашу програму та визначення активів.

На рисунку 4.3 зображено структуру серверної частини.

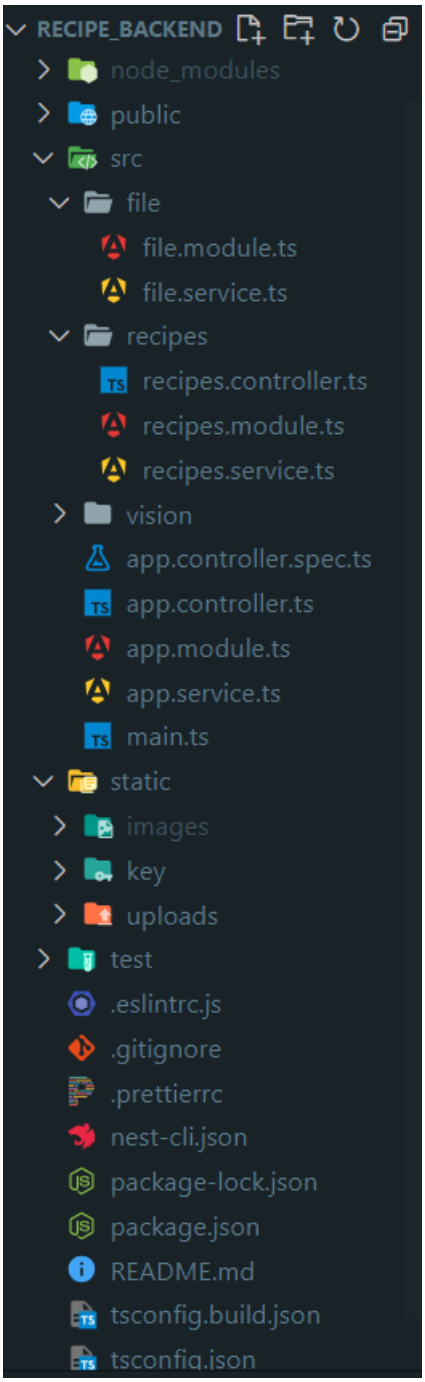

Рисунок 4.3 – Файли серверу

Джерело: побудовано автором (знімок з екрану)

Серверна частина містить каталог public, який потрібен для загально доступних файлів.

Каталог src містить основний функціонал, тут знаходиться точка входу додатку, також модулі, контролери та сервіси, які потрібні для створення власного API сервісу.

Каталог static містить статичні файли та картинки рецептів [36].

Файл tsconfig.json містить основні команди та залежності сервісу.

На рисунку 4.4 показано структуру мікросервісу.

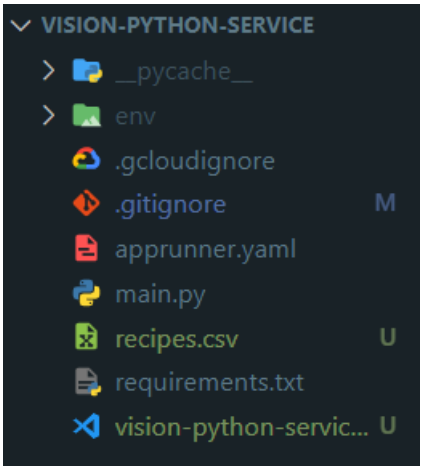

Рисунок 4.4 – Файли мікросервісу

Джерело: побудовано автором (знімок з екрану)

Сервіс містить основний файл main.py, який є точкою входу в додаток, та де описано весь функціонал.

Файл recipes.csv містить інформацію про рецепти в яких буде здійснюватися пошук.

Бібліотеки які використовувались для розробки мобільного додатка показано на рисунку 4.5.

```
dependencies:
  camera: ^0.10.5+4
  camera_avfoundation: ^0.9.13+7
  dio: ^5.3.3
  flutter:
    sdk: flutter
  flutter_screenutil: ^5.7.0
  flutter_svg: ^2.0.5
  google_fonts: ^4.0.4
  http: ^0.13.6
  http_parser: ^4.0.2
  image_picker: ^1.0.4
  json_annotation: ^4.8.1
  json_serializable: ^6.6.2
  shared_preferences: ^2.2.2
  sliding_up_panel: ^2.0.0+1
```

Рисунок 4.5 – Залежності мобільного додатка

Джерело: побудовано автором (знімок з екрану)

Нижче наданий короткий опис основних залежностей:

– camera: Ця бібліотека дозволяє використовувати камеру пристрою для зйомки фотографій і відео в додатках Flutter;

– shared_preferences: Потрібна для постійного сховища для простих даних;

– dio: Dio - це потужний HTTP-клієнт для Dart, який має підтримку перехоплювачів, глобальної конфігурації, FormData, скасування запитів, завантаження файлів та таймауту;

– flutter_screenutil: Ця бібліотека дозволяє легко адаптувати розмір екрану та шрифтів у додатках Flutter;

– sliding_up_panel: Ця бібліотека дозволяє створювати віджети, які можуть бути перетягнуті вгору, щоб показати або приховати вміст;

– flutter_svg: Ця бібліотека дозволяє відображати SVG-файли в додатках Flutter;

– google_fonts: Ця бібліотека дозволяє використовувати шрифти Google в додатках Flutter;

– http: Ця бібліотека дозволяє виконувати HTTP-запити в додатках Flutter;

– http_parser: Ця бібліотека дозволяє обробляти HTTP-запити в додатках Flutter;

– image_picker: Ця бібліотека дозволяє вибирати зображення з бібліотеки зображень пристрою або робити нові фотографії за допомогою камери;

– json_annotation: Ця бібліотека дозволяє створювати анотації для класів, що дозволяє їх серіалізувати в JSON

Допоміжний сервер та мікросервіс на Flask розміщені на платформі Amazon Lighsail [37] на базі операційної системи Linux Debian [38]. На рисунку 4.6 можна побачити характеристики серверів.

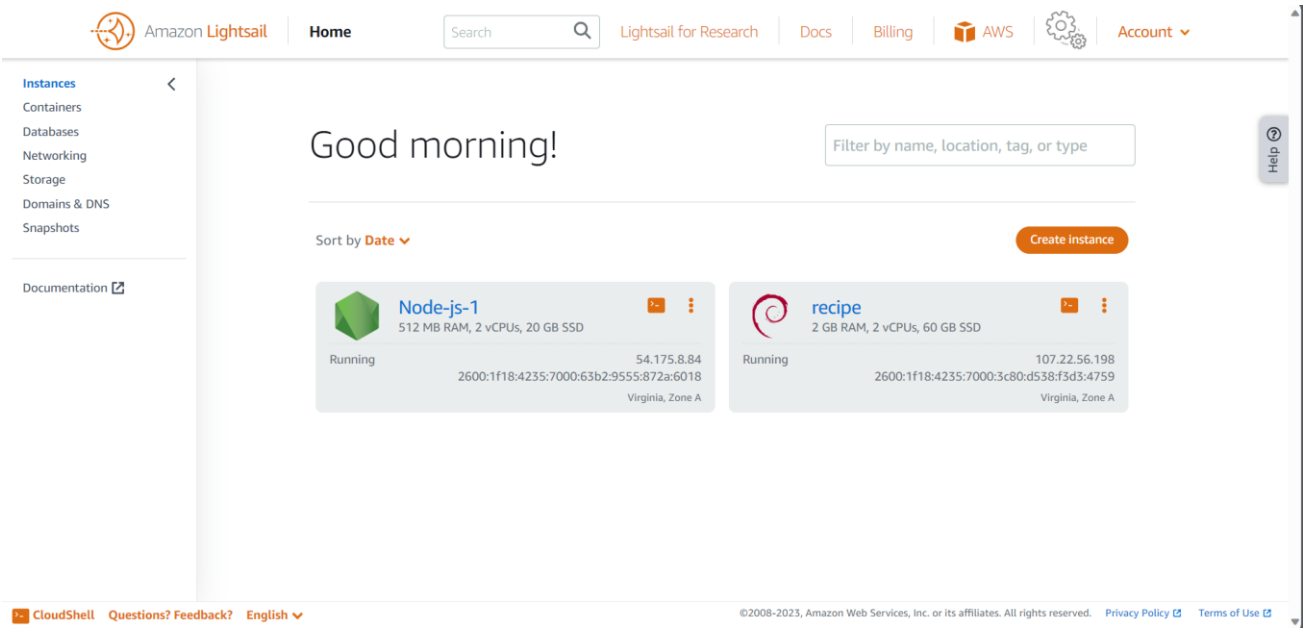

Рисунок 4.6 – Сервери мобільного додатку

Джерело: побудовано автором (знімок з екрану)

На рисунку 4.7 представлений лістинг функції для рекомендації рецептів на основі введених інгредієнтів користувача.

```python
def recommend_recipes(data, user_input_ingredients, offset, num_recommendations=5):

    tfidf_vectorizer = TfidfVectorizer()
    tfidf_matrix = tfidf_vectorizer.fit_transform(data['Ingredients'])

    user_input_vector = tfidf_vectorizer.transform([' '.join(user_input_ingredients)])
    similarities = cosine_similarity(user_input_vector, tfidf_matrix)

    # Визначення найбільш схожих рецептів
    top_recipe_indices = similarities.argsort()[0][::-1][int(num_recommendations):int(offset)+int(num_recommendations)]
    top_recipes = data.loc[top_recipe_indices, ['Title', 'Ingredients', 'Instructions', 'Image_Name']]

    json_data = top_recipes.to_dict(orient="records")

    return json_data
```

Рисунок 4.7 – Функція пошуку рецептів

Джерело: побудовано автором (знімок з екрану)

TfidfVectorizer використовується для перетворення текстових даних інгредієнтів рецептів в матрицю TF-IDF (Term Frequency-Inverse Document Frequency) [39].

Метод fit_transform застосовується до стовпця 'Ingredients' у вихідному наборі даних, створюючи TF-IDF матрицю.

Введені користувачем інгредієнти також трансформуються в матрицю за допомогою TfidfVectorizer і обчислюється їхній вектор TF-IDF.

Далі обчислюється косинусна схожість між всіма рецептами і вектором користувача в TF-IDF матриці. cosine_similarity генерує матрицю схожості [40].

Метод similarities.argsort сортує рецепти в порядку спадання за схожістю, top_recipe_indices обирає індекси рецептів, які будуть рекомендовані користувачу, враховуючи offset та num_recommendations. З допомогою цих індексів обирається певна кількість найбільш схожих рецептів з вихідного набору даних.

Потім створюється DataFrame, який містить відібрані рецепти з вихідного набору даних. Результат конвертується в формат JSON, де він передається іншій функції, яка відправляє вже по http запиту користувачеві.

На рисунку 4.8 показано частина лістингу отримання розпізнаних інгредієнтів з використанням Vision API.

```
@Injectable()
export class VisionService {
  async get_vision_image(file: string) {
    const client = new vision.ImageAnnotatorClient({
      keyFilename: './static/key/visionapi-399608-c8886036e1cc.json',
    });

    try {
      const [results] = await client.objectLocalization(file);
      const objects = results.localizedObjectAnnotations;

      const [labelsResults] = await client.labelDetection(file);
      const labels = labelsResults.labelAnnotations;
      const labelDescriptions = labels.map(
        (label: { description: string }) => ({
          description: label.description,
        }),
      );
```

Рисунок 4.8 – Отримання даних про інгредієнти

Джерело: побудовано автором (знімок з екрану)

Ця функція використовує об'єкт vision.ImageAnnotatorClient для взаємодії з Google Cloud Vision API. Вона отримує об'єкти та мітки на зображенні інгредієнтів і повертає об'єкт, що містить ці дані. Файл ключа автентифікації використовується для автентифікації до веб-сервісу Google Cloud Vision API.

Далі проводиться підготовка даних до відправлення користувачу: вибірка релевантних інгредієнтів, виключення непотрібних слів, приведення координат до однієї точки. Лістинг коду зображено на рисунку 4.9.

```
const objectData = objects
  .filter((object: any) => {
    const name = object.name.toLowerCase();
    return (
      object.score > 0.8 &&
      !name.includes('food') &&
      !name.includes('vegetable')
    );
  })
  .map((object: any) => {
    return {
      name: object.name,
      similarity: Math.round(object.score * 100),
      coordinates: {
        x:
          (object.boundingPoly.normalizedVertices[0].x +
            object.boundingPoly.normalizedVertices[1].x) /
          2,
        y: object.boundingPoly.normalizedVertices[0].y,
      },
    };
  });
```

Рисунок 4.9 – Підготовка даних

Джерело: побудовано автором (знімок з екрану)

Лістинг основних програмних файлів мобільного додатку наведено у додатку Б.

4.2 Використання додатку

Використання мобільним додатком розпочинається з головного екрану, який показано на рисунку 4.10, де можна побачити назву додатка, кнопку для переходу в розділ улюблених рецептів, віджет для переходу до сканування інгредієнтів, список рецептів та текстове поле для пошуку рецептів.

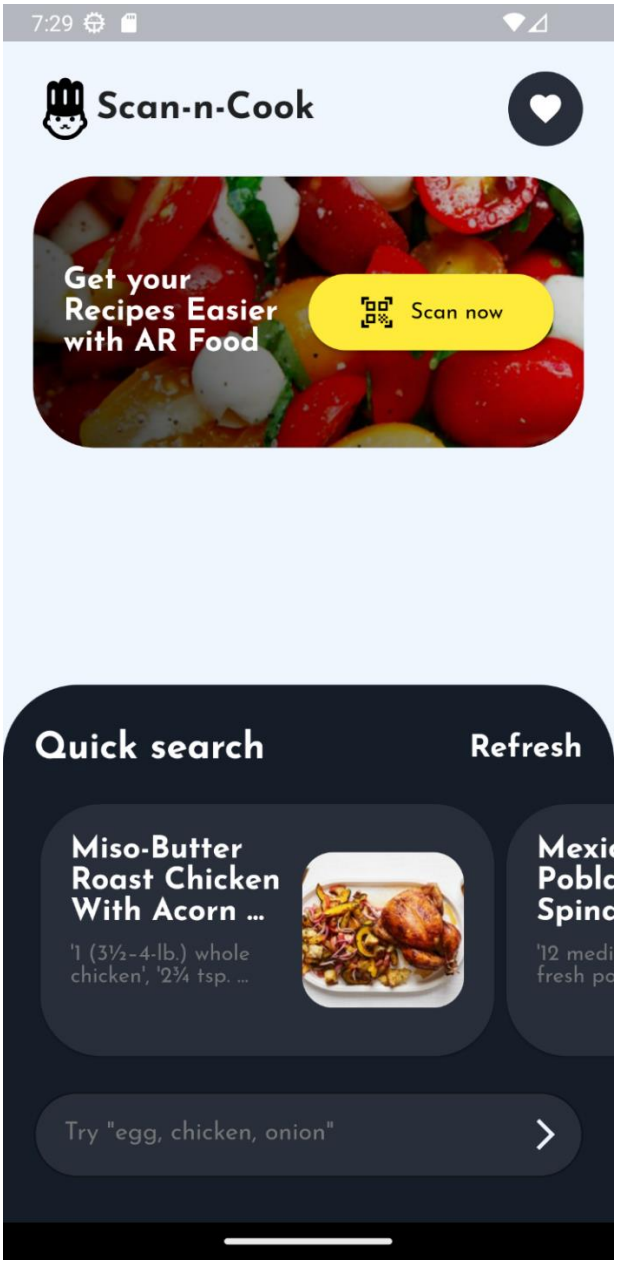

Рисунок 4.10 – Головний екран

Джерело: побудовано автором (знімок з екрану)

На рисунку 4.11 зображено пошук інгредієнтів через текстове поле внизу екрана.

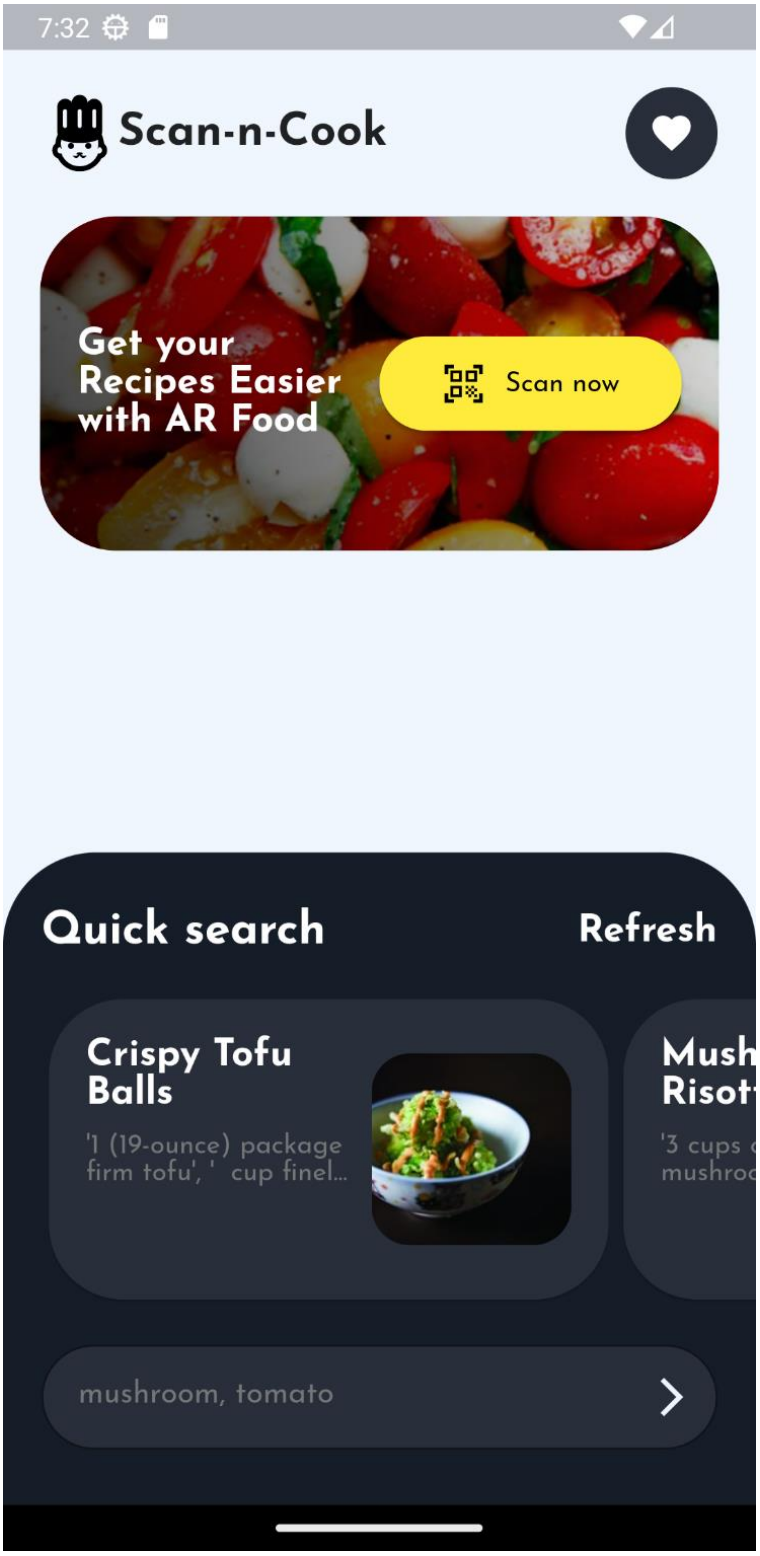

Рисунок 4.11 – Пошук рецептів через текстове поле

Джерело: побудовано автором (знімок з екрану)

При натисканні кнопки «Scan now», користувач переходить до вікна сканування інгредієнтів, де він фотографує наявні в нього інгредієнти, для кращого

фотографування можна ввімкнути спалах, зображено на рисунку 4.12. Також можна вибрати наявне зображення продуктів зі свого сховища, зображено на рисунку 4.13.

Рисунок 4.12 – Вікно для сканування інгредієнтів

Джерело: побудовано автором (знімок з екрану)

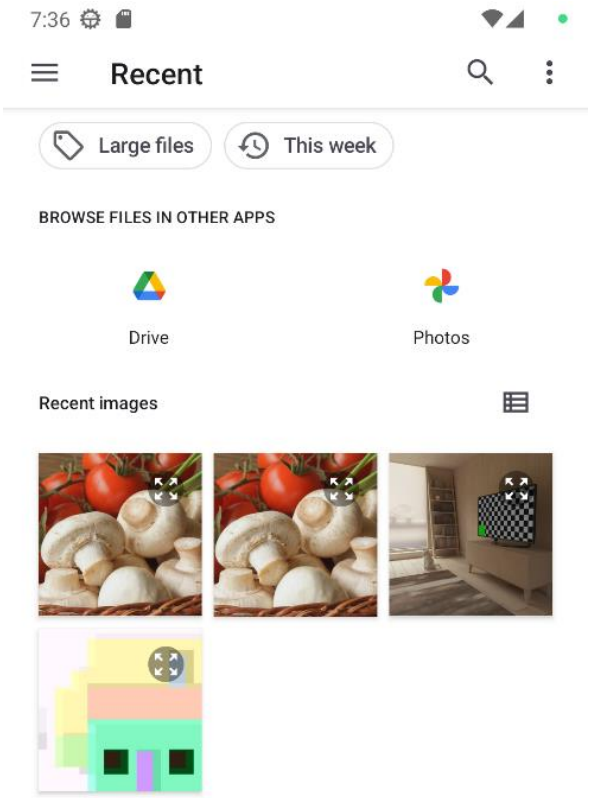

Рисунок 4.13 – Вибір зображення зі сховища

Джерело: побудовано автором (знімок з екрану)

Після сканування користувач переходить до вікна з визначеними інгредієнтами, показано на рисунку 4.14.

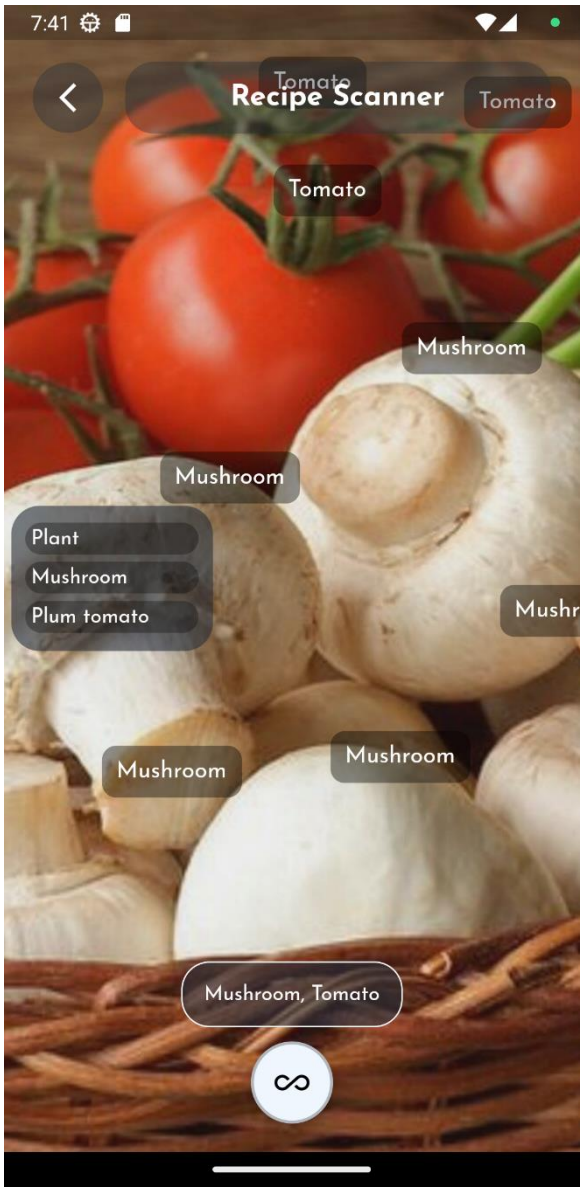

Рисунок 4.14 – Вікно з інформацією про інгредієнти

Джерело: побудовано автором (знімок з екрану)

Якщо інгредієнти визначені неналежним чином, то користувач також може додати можливий інгредієнт зі списку більш ймовірних продуктів, який знаходиться зліва, зображено на рисунку 4.15.

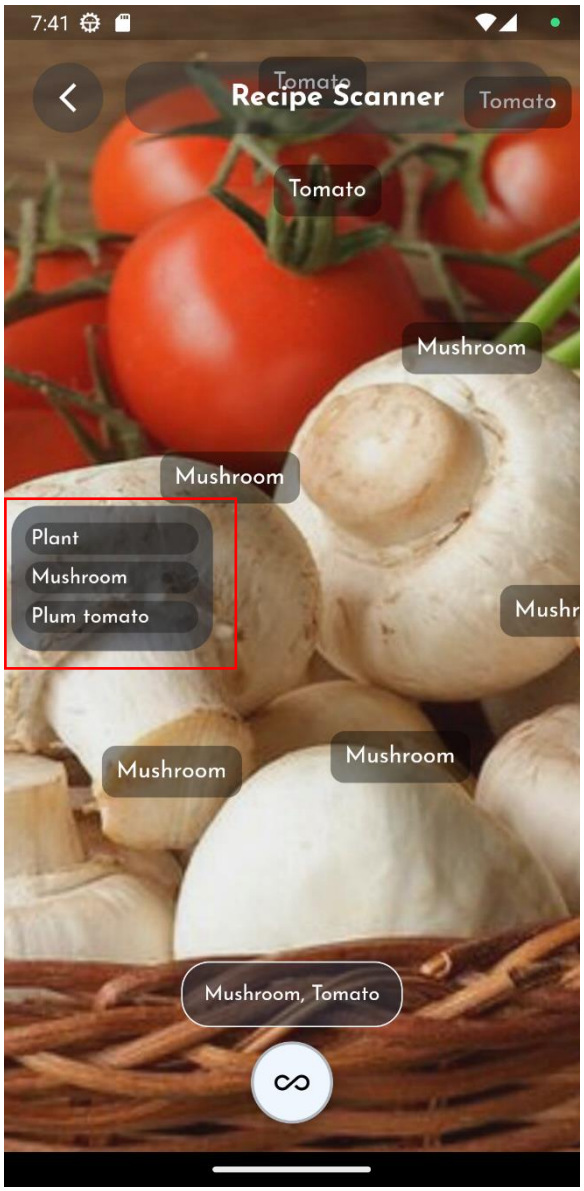

Рисунок 4.15 – Вікно з інформацією про інгредієнти

Джерело: побудовано автором (знімок з екрану)

Після натискання кнопки пошуку користувач переходить до вікна зі списком підібраних рецептів, де можна перейти до підходящого рецепта, щоб детальніше ознайомитися з ним, зображено на рисунку 4.16.

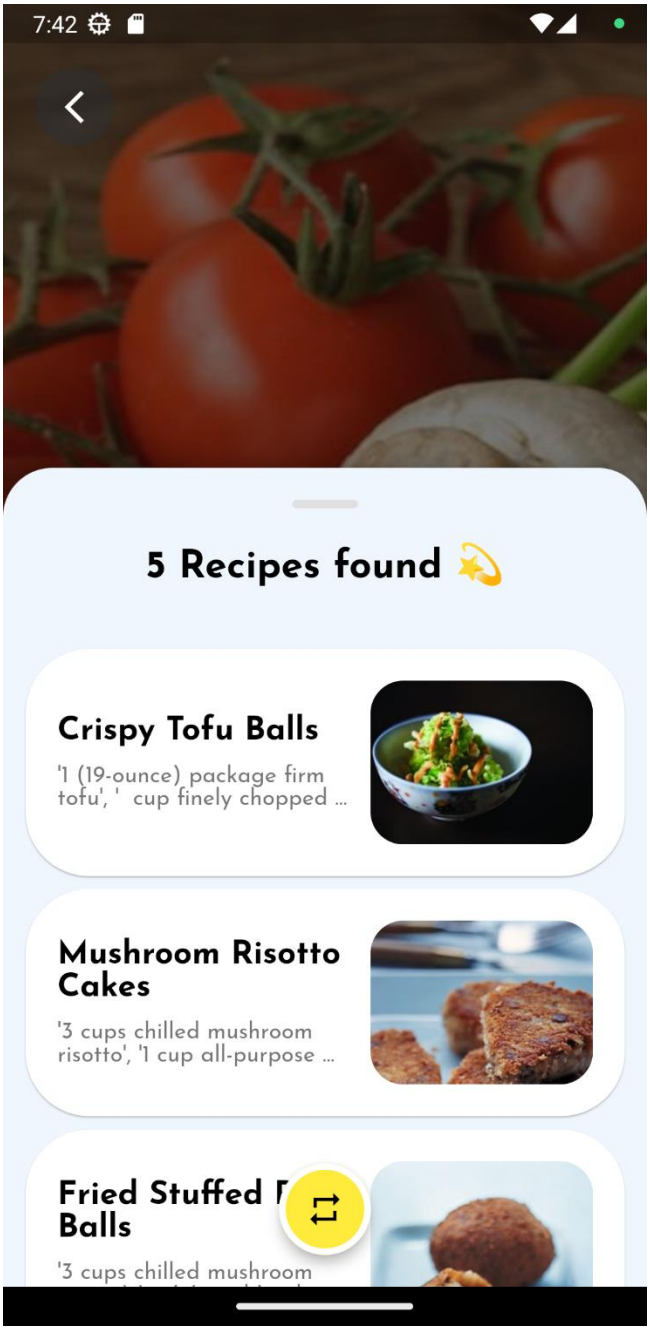

Рисунок 4.16 – Вікно зі знайденими рецептами

Джерело: побудовано автором (знімок з екрану)

Також користувач може натиснути кнопку внизу екрану для оновлення списку новими рецептами, зображено на рисунку 4.17.

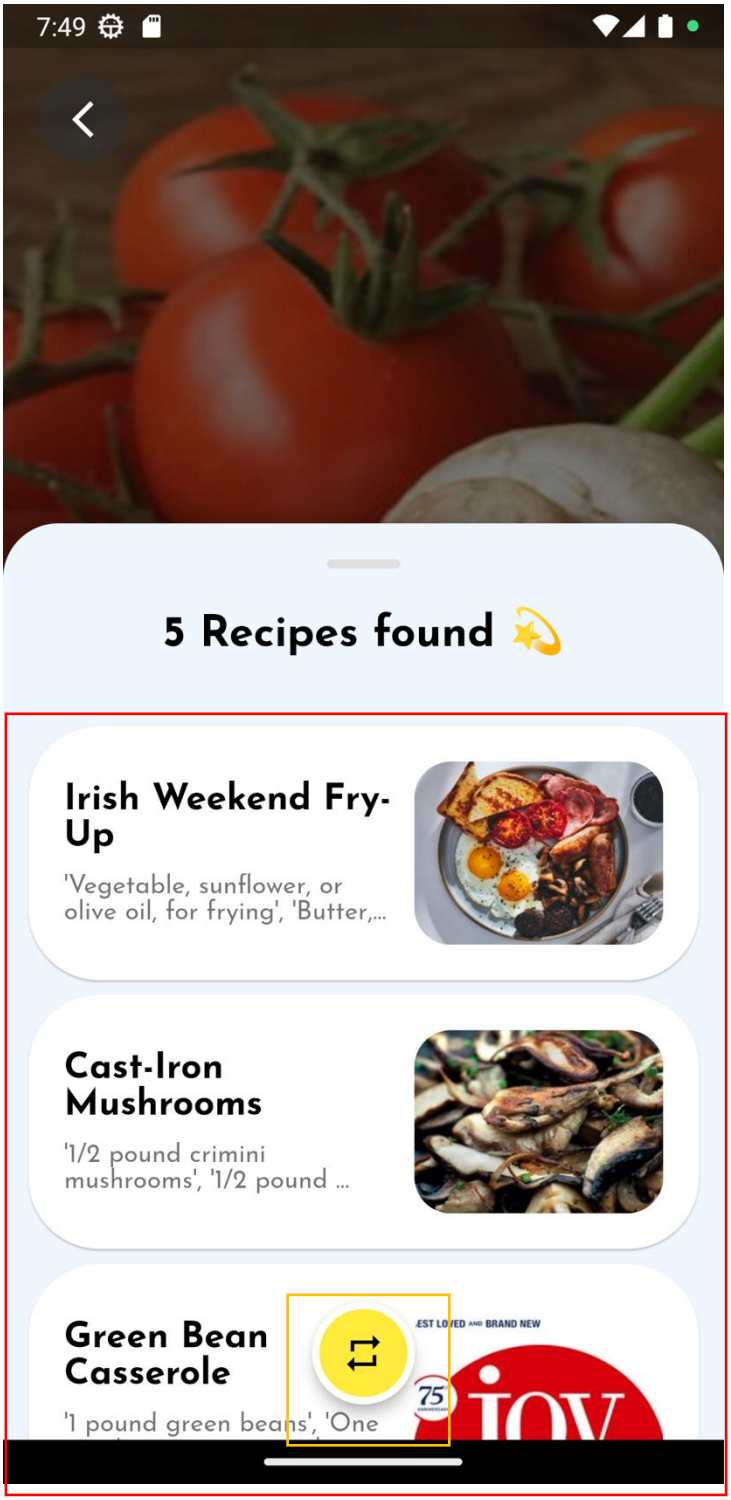

Рисунок 4.17 – Оновлений список рецептів

Джерело: побудовано автором (знімок з екрану)

Якщо натиснути на потрібний рецепт, то користувач попадає до вікна з детальнішим ознайомленням з рецептом, зображено на рисунку 4.18. Де також можна додати рецепт до улюблених, щоб повернутися до нього знову.

Рисунок 4.18 – Вікно з описом рецепту

Джерело: побудовано автором (знімок з екрану)

Якщо користувач повернеться до головного вікна і натисне на кнопку з іконкою «Серце», то він перейде до вікна з улюбленими рецептами, зображено на рисунку 4.19.

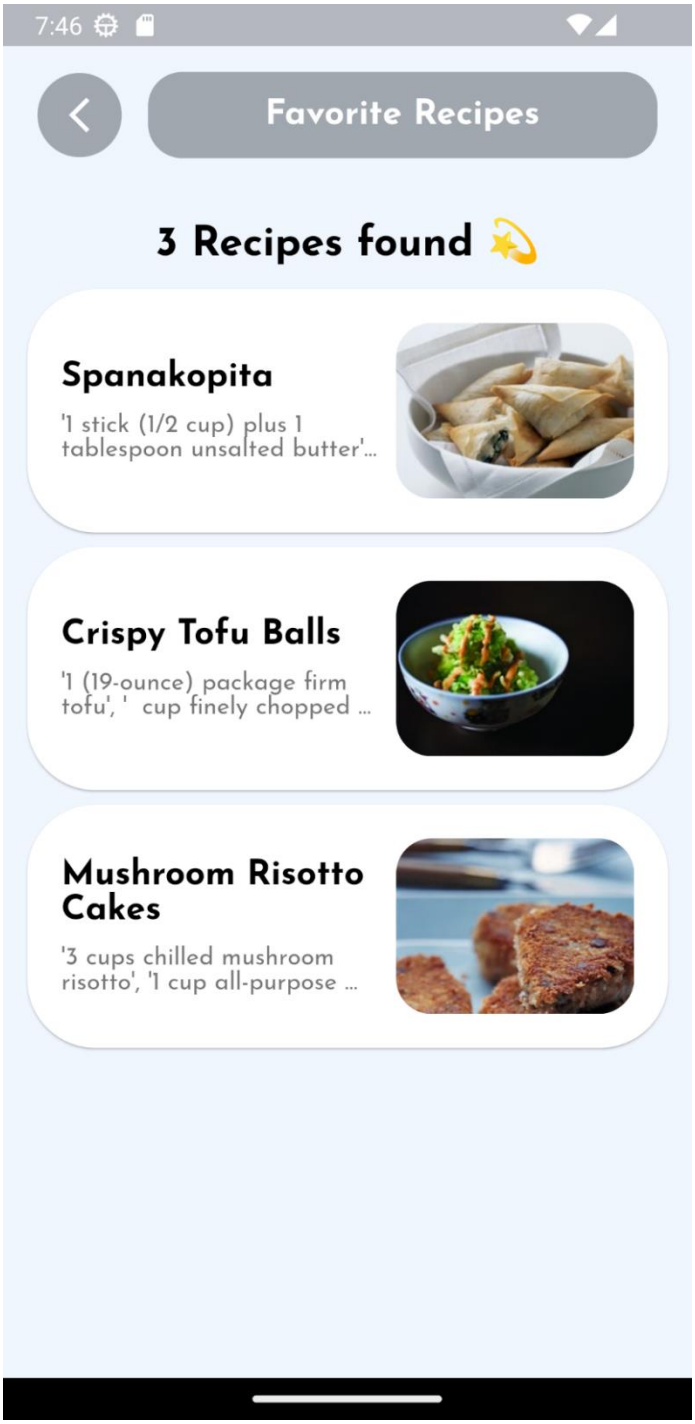

Рисунок 4.19 – Вікно улюблених рецептів

Джерело: побудовано автором (знімок з екрану)

ВИСНОВКИ

Під час проходження практики було вивчено та проаналізовано ключові елементи для розробки програмного забезпечення. Було проведено аналіз проектної документації, збір, систематизація й узагальнення матеріалу для реалізації мобільного додатку.

Були вибрані методи дослідження та встановлено необхідне програмне забезпечення, а саме фреймворк Flutter для розробки інтерфейса додатка, платформу Vision API для розпізнавання інгредієнтів на зображенні, Flask з бібліотекою pandas для реалізації мікросервісу, який буде обробляти датасет та метод пошуку TF-IDF з бібліотеки scikit-learn, який буде здійснювати безпосередній пошук рекомендованих рецептів, налаштоване відповідне середовище розробки для мобільного додатку.

Виконано планування робіт з розроблення мобільного додатку та спроектовано, змодельовано мобільний додаток підбору рецептів з використанням штучного інтелекту, за допомогою стандарту IDEF0 була розроблена контекстна діаграма та діаграма першого рівня декомпозиції. Різні сценарії використання мобільного додатку були представлені у вигляді Use Case діаграми, яку було створено. Встановлені часові рамки відображаються у вигляді таймлайну та діаграми Ганта, а також проведено аналіз потенційних ризиків та способів їх нейтралізації.

Розробка представленого мобільного продукту була здійснена за допомогою фреймворка Flutter. Створений мобільний додаток в повному обсязі відповідає всім встановленим вимогам. Його тестування на користувачах пройшло успішно.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. App annie. [електронний ресурс] режим доступу до ресурсу: https://www.data.ai/en/go/state-of-mobile-2023/

2. Schmidt, «The end of serendipity: Will artificial intelligence remove chance and choice in everyday life?», в ACM International Conference Proceeding Series, 2021. doi: 10.1145/3464385.3464763.

3. The Problem with Recipes. [Електронний ресурс] Режим доступу до ресурсу: https://comfortdujour.com/2022/04/05/the-problem-with-recipes/

4. J. L. Krenn, «Developing an “App-titude” for Cooking», 2019. doi: 10.4018/978-1-5225-9351-5.ch002.

5. Q. Zhang, J. Lu, і Y. Jin, «Artificial intelligence in recommender systems», Complex and Intelligent Systems, вип. 7, вип. 1, 2021, doi: 10.1007/s40747-020-00212-w.

6. M. K. Morol, M. S. J. Rokon, I. B. Hasan, A. M. Saif, R. H. Khan, і S. S. Das, «Food Recipe Recommendation Based on Ingredients Detection Using Deep Learning», в ACM International Conference Proceeding Series, 2022. doi: 10.1145/3542954.3542983.

7. K. Bhareti, S. Perera, S. Jamal, M. H. Pallege, V. Akash, і S. Wiieweera, «A Literature Review of Recommendation Systems», в 2020 IEEE International Conference for Innovation in Technology, INOCON 2020, 2020. doi: 10.1109/INOCON50539.2020.9298450.

8. G. Yunanda, D. Nurjanah, і S. Meliana, «Recommendation System from Microsoft News Data using TF-IDF and Cosine Similarity Methods», Building of Informatics, Technology and Science (BITS), вип. 4, вип. 1, 2022, doi: 10.47065/bits.v4i1.1670.

9. SuperCook. [Електронний ресурс] Режим доступу до ресурсу: https://www.supercook.com/#/desktop

10. Recipe Search. [Електронний ресурс] Режим доступу до ресурсу: https://www.reciperadar.com/

11. Tesco Real Food. [Електронний ресурс] Режим доступу до ресурсу: https://realfood.tesco.com/what-can-i-make-with.html

12. Pantry Chef. [Мобільний додаток] Режим доступу до ресурсу: https://apps.apple.com/ua/app/pantry-chef-recipe-finder/id6472689381?l=uk

13. ChatGPT. [Електронний ресурс] Режим доступу до ресурсу: https://chat.openai.com/

14. Petter Törnberg, «ChatGPT-4 Outperforms Experts and Crowd Workers in Annotating Political Twitter Messages with Zero-Shot Learning», 2023. doi: 10.48550/arXiv.2304.06588.

15. Vision AI. [Електронний ресурс] Режим доступу до ресурсу: https://cloud.google.com/vision?hl=uk

16. VisionAPI. [Електронний ресурс] Режим доступу до ресурсу: https://cloud.google.com/vision/docs

17. W. McKinney і P. D. Team, «Pandas - Powerful Python Data Analysis Toolkit», Pandas - Powerful Python Data Analysis Toolkit, 2015.

18. D. Fan, F. Wang, і J. He, «EAST Experimental Data Gateway Based on RESTful and Microservice Architecture», в 2023 4th International Conference on Computer Engineering and Application, ICCEA 2023, 2023. doi: 10.1109/ICCEA58433.2023.10135363.

19. J. Hao і T. K. Ho, «Machine Learning Made Easy: A Review of Scikit-learn Package in Python Programming Language», Journal of Educational and Behavioral Statistics, вип. 44, вип. 3. 2019. doi: 10.3102/1076998619832248.

20. R. Payne, Beginning App Development with Flutter: Create Cross-Platform Mobile Apps. 2019. doi: 10.1007/978-1-4842-5181-2.

21. Food Ingredients and Recipes Dataset with Images. [Електронний ресурс] Режим доступу до ресурсу: https://www.kaggle.com/datasets/pes12017000148/food-ingredients-and-recipe-dataset-with-images

22. Feng Wen, Zixuan Zhang, Tianyiyi He, Chengkuo Lee, «AI enabled sign language recognition and VR space bidirectional communication using triboelectric smart glove», 2021, doi: 10.1038/s41467-021-25637-w.

23. Putra Fissabil Muhammad, R. Kusumaningrum, A. Wibowo « Sentiment Analysis Using Word2vec And Long Short-Term Memory (LSTM) For Indonesian Hotel Reviews», 2022. doi: 10.1016/J.PROCS.2021.01.061.

24. Tirta Hema Jaya Hidayat, Y. Ruldeviyani, A. R. Aditama, Gusti Raditia Madya, Ade Wija Nugraha, Muhammad Wijaya Adisaputra, « Sentiment analysis of twitter data related to Rinca Island development using Doc2Vec and SVM and logistic regression as classifier», 2022. doi: 10.1016/j.procs.2021.12.187.

25. T. H. Huong, K. Tran-Trung, D. T. C. Lai, і V. T. Hoang, «Sentiment Analysis based on word vector representation for short comments in Vietnamese language», в Proceedings - 2022 9th NAFOSTED Conference on Information and Computer Science, NICS 2022, 2022. doi: 10.1109/NICS56915.2022.10013426.

26. D. E. Cahyani і I. Patasik, «Performance comparison of tf-idf and word2vec models for emotion text classification», Bulletin of Electrical Engineering and Informatics, вип. 10, вип. 5, 2021, doi: 10.11591/eei.v10i5.3157.

27. H. Zhou, « Research of Text Classification Based on TF-IDF and CNN-LSTM», 2022. doi: 10.1088/1742-6596/2171/1/012021.

28. H. Vranken, Hassan Alizadeh, «Detection of DGA-Generated Domain Names with TF-IDF», 2022. doi: 10.3390/electronics11030414.

29. Maarten Grootendorst «BERTopic: Neural topic modeling with a class-based TF-IDF procedure», 2022. doi: 10.48550/arXiv.2203.05794.

30. G. Manenti, M. Ebrahimiarjestan, L. Yang, і M. Yu, «Functional modelling and IDEF0 to enhance and support process tailoring in systems engineering», в ISSE 2019 - 5th IEEE International Symposium on Systems Engineering, Proceedings, 2019. doi: 10.1109/ISSE46696.2019.8984539.

31. P.-M. Spanidis, F. Pavloudakis, і C. Roumpos, «Introducing the IDEF0 Methodology in the Strategic Planning of Projects for Reclamation and Repurposing of Surface Mines», 2021. doi: 10.3390/materproc2021005026.

32. O. Sylkin, M. Kryshtanovych, A. Zachepa, S. Bilous, і A. Krasko, «Modeling the process of applying anti-crisis management in the system of ensuring

financial security of the enterprise», Business: Theory and Practice, вип. 20, 2019, doi: 10.3846/btp.2019.41.

33. What is Use Case Diagram? [Електронний ресурс] Режим доступу до ресурсу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/

34. M. N. Arifin, D. Siahaan, «Structural and Semantic Similarity Measurement of UML Use Case Diagram», 2020. doi: 10.24843/lkjiti.2020.v11.i02.p03.

35. Alfansyah Nori Pratama, Farida Ardiani, « Optimization of Model-View-ViewModel (MVVM) Architecture Pattern and RESTfull API on Android-based E-Learning Application», 2023. doi: 10.5120/ijca2023923261.

36. Pham, Anh Duc,« DEVELOPING BACK-END OF A WEB APPLICATION WITH NESTJS FRAMEWORK Case: Integrify Oy's student management system», 2020.

37. Amazon Lightsail. [Електронний ресурс] Режим доступу до ресурсу: https://lightsail.aws.amazon.com/ls/webapp/home/instances

38. Debian. [Електронний ресурс] Режим доступу до ресурсу: https://www.debian.org/

39. Vipin Kumar, Basant Subba, « A TfidfVectorizer and SVM based sentiment analysis framework for text data corpus», 2020. doi: 10.1109/NCC48643.2020.9056085.

40. R. Verma, A. Mittal , « Multiple attribute group decision-making based on novel probabilistic ordered weighted cosine similarity operators with Pythagorean fuzzy information», 2022. doi: 10.1007/s41066-022-00318-1.

ДОДАТОК А. ПЛАНУВАННЯ РОБІТ

А.1 Ідентифікація мети ІТ-проекту

Метою проекту є розробка мобільного додатку підбору рецептів з використанням штучного інтелекту.

Деталізація мети методом SMART:

S – Метою проекту є розробка мобільного додатку підбору рецептів з використанням штучного інтелекту.

M – додаток, який зможе надати користувачам рецепти, які відповідають їхнім вимогам у 90% випадків.

A – Використовуючи технології штучного інтелекту та машинного навчання, ми можемо навчити систему підбирати відповідні рецепти.

R – Цей проект важливий, оскільки він спрощує процес вибору рецептів, дозволяючи користувачам використовувати продукти, які вже є у них на кухні та витрачати на це менше свого вільного часу.

T – Мета - розробити та запустити додаток протягом наступних 2 місяців, при роботі 4-6 годин на день з вихідними.

А.2 Планування змісту структури робіт інформаційної системи

Основний інструмент для організації та планування структури робіт - це WBS (Work Break Structure). Це представлення проекту у вигляді ієрархічної структури робіт, яка створюється за допомогою послідовної декомпозиції. Цей інструмент спрямований на детальне планування, оцінку вартості, визначення та розподіл персональної відповідальності виконавців. Він фокусується на основних роботах і результатах, які визначають зміст проекту.

На рисунку А.1 приведена WBS структура проекту.

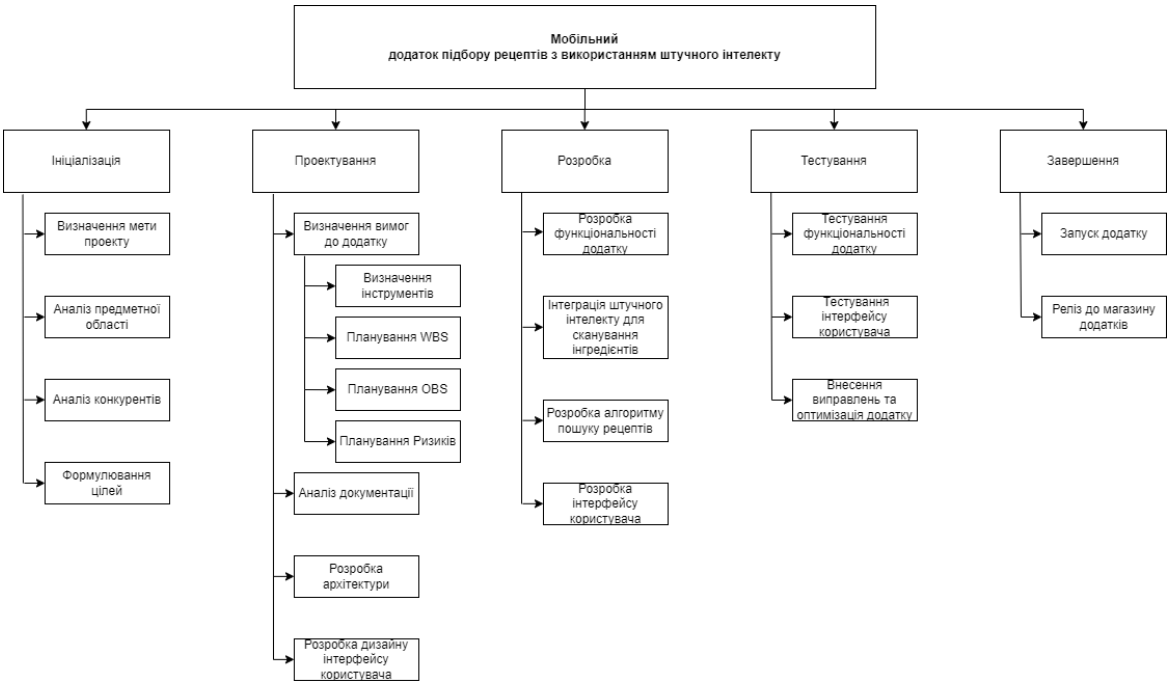

Рисунок А.1 – WBS-структура проекту

Джерело: побудовано автором

Після створення структури WBS проекту, наступним кроком є розробка OBS (Organization Break structure). Вона включає в себе склад, ієрархію, взаємодію та розподіл робіт між підрозділами та органами управління. Встановлюються певні відносини щодо реалізації владних повноважень, потоків команд та інформації.

Організаційна структура проекту стосується лише внутрішньої організаційної структури проекту.

Організаційна структура проекту зображена на рисунку А.2.

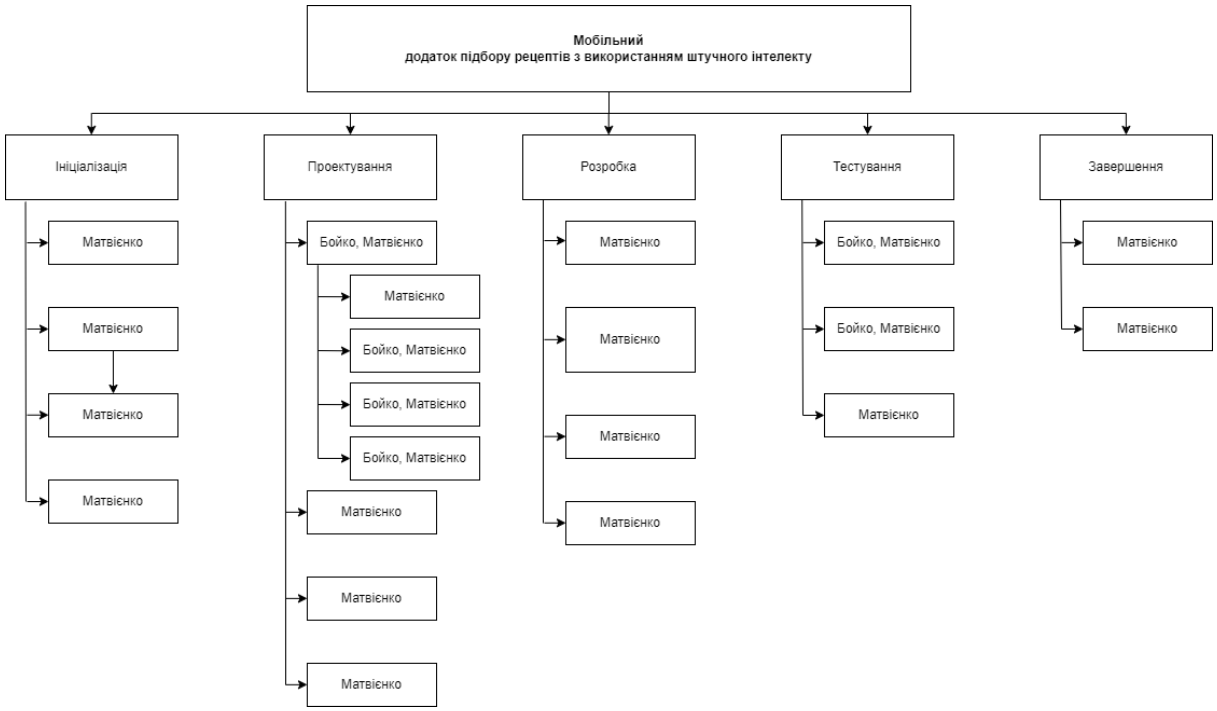

Рисунок А.2 – OBS-структура проекту

Джерело: побудовано автором

Список виконавців, що функціонують в проекті представлений в таблиці А.1.

Таблиця А.1 – Виконавці проекту

Джерело: побудовано автором

Ім’я	Роль	Задачі
Матвієнко В.О.	Розробник	Займається розробкою функціоналу проекту
Бойко О. В.	Менеджер проекту, тестувальник	Відповідає за виконання термінів та за тестування функціоналу.

А.3 Побудова календарного графіку виконання ІТ - проекту

Діаграма Ганта - це графічний інструмент для представлення часового плану проекту. Вона названа на честь американського інженера і консультанта з управління Генрі Ганта, який розробив цей метод у 1910-х роках.

Діаграма Ганта використовується для візуалізації графіку проекту, де горизонтальні бари представляють кожну задачу проекту. Довжина кожного бару відображає тривалість задачі, а положення бару на часовій шкалі відображає початок та кінець задачі.

Календарний план виконання робіт для побудови діаграми Ганта представлений на рисунку А.3. Сама діаграма Ганта представлена на рисунку А.4.

Задача	Дата початку	Дата закінчення	Кількість днів
Ініціалізація			4
Визначення мети проекту	07.10.2023	07.10.2023	1
Аналіз предметної області	08.10.2023	08.10.2023	1
Аналіз конкурентів	09.10.2023	09.10.2023	1
Формулювання цілей	10.10.2023	10.10.2023	1
Проектування			21
Визначення вимог до додатку			8
Визначення інструментів	11.10.2023	13.10.2023	3
Планування WBS	14.10.2023	16.10.2023	3
Планування OBS	17.10.2023	17.10.2023	1
Планування Ризиків	18.10.2023	18.10.2023	1
Аналіз документації	19.10.2023	21.10.2023	3
Проектування архітектури	22.10.2023	26.10.2023	5
Проектування дизайну інтерфейсу користувача	27.10.2023	31.10.2023	5
Розробка			30
Розробка функціональності додатку	01.11.2023	12.11.2023	12
Інтеграція штучного інтелекту для сканування інгредієнт	13.11.2023	15.11.2023	3
Розробка алгоритму пошуку рецептів	16.11.2023	19.11.2023	4
Розробка інтерфейсу користувача	20.11.2023	30.11.2023	11
Тестування			3
Тестування функціональності додатку	01.12.2023	01.12.2023	1
Тестування інтерфейсу користувача	01.12.2023	01.12.2023	1
Внесення виправлень та оптимізація додатку	02.12.2023	02.12.2023	1
Завершення			2
Запуск додатку	03.12.2023	03.12.2023	1
Реліз до магазину додатків	04.12.2023	04.12.2023	1
			60

Рисунок А.3 – Календарний план

Джерело: побудовано автором (знімок з екрану)

Рисунок А.4 – Діаграма Ганта

Джерело: побудовано автором (знімок з екрану)

Як можна зрозуміти розробка додатка проводилась протягом двох місяців, при роботі 4-6 годин на день з вихідними.

А.4 Планування ризиків проекту

Планування ризиків проекту - це процес ідентифікації, аналізу та відповіді на потенційні проблеми, які можуть вплинути на успішне виконання проекту. Це включає в себе визначення ризиків, оцінку їх впливу та ймовірності, розробку стратегій реагування та моніторинг та контроль ризиків протягом життєвого циклу проекту.

Класифікація ризиків для даного проєкта є наступна:

1. Ймовірність виникнення:

– низька (1);

– середня (2);

– висока (3).

2. Величина впливу:

– низька (1);

– середня (2);

– висока (3).

На основі даних проведено класифікацію. ризиків для даного проекту, яка наведена в таблиці А.2 та збудована матриця ризиків на рисунку А.5.

Таблиця А.2 – Класифікація ризиків

Джерело: побудовано автором

Ідентифікатор	Назва ризику	Ймовірність виникнення	Величина впливу
R1	Проблеми зі створенням інтуїтивно зрозумілого інтерфейсу користувача	2	2

Продовження таблиці А.2

R2	Збої в роботі алгоритму пошуку рецептів	1	3
R3	Проблеми з якістю даних рецептів	1	2
R4	Проблеми з обробкою великих обсягів даних	1	2
R5	Недотримання календарного плану	2	2
R6	Недоліки при плануванні робіт	2	1
R7	Відсутність електрики	2	3
R8	Поломка серверів	1	3
R9	Виявлення нових багів під час користування додатком	3	1
R10	Труднощі із інтеграцією сторонніх бібліотек і модулів	1	1
R11	Зміни вимог або обсягу робіт під час розробки	3	2

Продовження таблиці А.2

R12	Недооцінка часу і ресурсів для деяких етапів проекту	3	1
R13	Ризик не правильного функціонування додатку	3	3

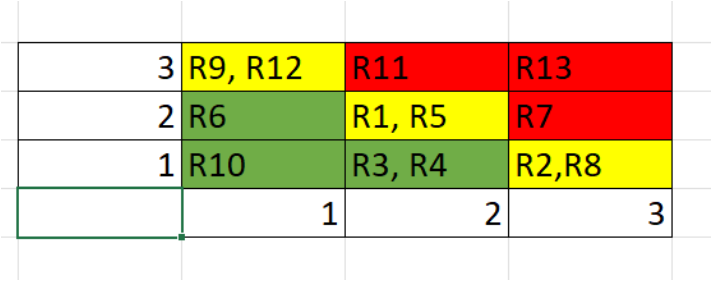

Рисунок А.5 – Матриця ризиків

Джерело: побудовано автором (знімок з екрану)

В таблиці А.3 вказано ранги ризиків та варіанти їх запобігання.

Таблиця А.3 – Ранг ризику та варіанти запобігання

Джерело: побудовано автором

Ідентифікатор	Ранг ризику	Варіанти запобігання
R1	4	Зробити інтерфейс більш простим, не навантажувати його зайвими компонентами.
R2	3	Провести тестування на інших даних, провести детальну перевірку алгоритму.
R3	2	Знайти або зібрати новий датасет.

Продовження таблиці А.3

R4	2	1. Оптимізувати запити до даних 2. Використати хмарні рішення 3. Підібрати ефективніший сервер
R5	4	Створити детальніший план роботи над проектом
R6	2	Вирішити наявні недоліки вже під час роботи
R7	6	Найти альтернативне джерело електроенергії
R8	3	Мати додаткові сервери; підібрати надійні сервісі, які надають сервери.
R9	3	Виправляти баги в нових версіях додатку.
R10	1	Попросити допомоги більш кваліфікованого спеціаліста; детальніше прочитати документацію та уважніше повторити процес.

Продовження таблиці А.3

R11	6	Чітко визначити вимоги на початку проєкта; створити гнучкий план на такий випадок; проводити регулярний моніторинг та контроль.
R12	3	Використовувати методи оцінки для прогнозування часу на виконання робіт; мати буфери часу і ресурсів.
R13	9	Проводити регулярне тестування під час розробки; провести бета-тестування для користувачів для отримання відгуків; своєчасне оновлення та покращення додатку.

ДОДАТОК Б

Лістинг основних модулів

main.dart

```
import 'package:camera/camera.dart';
import 'package:flutter/material.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';
import 'package:recipe_app/screens/splash_screen.dart';
import 'package:recipe_app/themes.dart';

List<CameraDescription> cameras = [];

Future<void> main() async {
  try {
    WidgetsFlutterBinding.ensureInitialized();
    cameras = await availableCameras();
  } on CameraException catch (e) {
    print('Error in fetching the cameras: $e');
  }
  runApp(const MainApp());
}

class MainApp extends StatelessWidget {
  const MainApp({super.key});

  @override
  Widget build(BuildContext context) {
    return ScreenUtilInit(
      designSize: const Size(393, 808),
      builder: (_, child) {
        return MaterialApp(
          debugShowCheckedModeBanner: false,
          theme: lightTheme,
          darkTheme: darkTheme,
          home: const SplashScreen(),
        );
      },
    );
  }
}
```

home_screen.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';
import 'package:flutter_svg/flutter_svg.dart';
import 'package:recipe_app/screens/favorite_screen.dart';
import 'package:recipe_app/screens/scaner_screen.dart';
import 'package:recipe_app/widgets/app_icon.dart';
import 'package:recipe_app/widgets/large_text.dart';

import 'package:recipe_app/widgets/search_panel.dart';

class HomeScreen extends StatefulWidget {
  const HomeScreen({super.key});

  @override
  State<HomeScreen> createState() => _HomeScreenState();
}

class _HomeScreenState extends State<HomeScreen> {
  bool isFinished = false;
  @override
  Widget build(BuildContext context) {
    return Scaffold(
        backgroundColor: Theme.of(context).colorScheme.background,
        body: SafeArea(
            child: Column(
          children: [
            Padding(
              padding: EdgeInsets.only(top: 20.h, left: 20.w, right:
20.w),
              child: Row(
                mainAxisAlignment: MainAxisAlignment.spaceBetween,
                children: [
                  Row(
                    children: [
                      SvgPicture.asset(
                        'assets/images/cook-face-svgrepo-com.svg',
                        height: 40.0,
                        width: 40.0,
                      ),
                      const LargeText(
                        text: "Scan-n-Cook",
                        color: Colors.black87,
                      ),
                    ],
                  ),
                  Row(
                    children: [
                      AppIcon(
                          icon: Icons.favorite,
                          onPressed: () {
```

```
                            Navigator.push(
                              context,
                              MaterialPageRoute(
                                  builder:    (context)    =>    const
FavoriteScreen()),
                              );
                            }),
                        // SizedBox(
                        //   width: 10.w,
                        // ),
                        //  AppIcon(icon:  Icons.person,  onPressed:  ()
{}),
                      ],
                    )
                  ],
                ),
              ),
              Padding(
                padding: const EdgeInsets.all(20).r,
                child: Container(
                    height: 180.h,
                    width: 1.sw,
                    decoration: const BoxDecoration(
                      borderRadius:
BorderRadius.all(Radius.circular(40)),
                      image: DecorationImage(
                          image: NetworkImage(

"https://i.pinimg.com/564x/53/67/fc/5367fc8d1aa4c53bc412c397c501d73b
.jpg"),
                          fit: BoxFit.cover),
                    ),
                    child: Container(
                      decoration: BoxDecoration(
                          borderRadius:
                              const
BorderRadius.all(Radius.circular(40)),
                          gradient: LinearGradient(
                            begin: Alignment.bottomLeft,
                            end: Alignment.topRight,
                            colors: [
                              const    Color.fromARGB(255,    0,    0,
0).withOpacity(0.8),
                              const    Color.fromARGB(255,    0,    0,
0).withOpacity(0.1),
                            ],
                          )),
                      child: Padding(
                        padding: const EdgeInsets.all(20).r,
                        child: Row(
                          children: [
                            Expanded(
```

```
                              flex: 1,
                              child: Text(
                                "Get your Recipes Easier with AR Food
Scaner",
                                style:
Theme.of(context).textTheme.displaySmall,
                                maxLines: 3,
                              ),
                            ),
                            Expanded(
                                flex: 1,
                                child: ElevatedButton(
                                    onPressed: () {
                                      Navigator.push(
                                        context,
                                        MaterialPageRoute(
                                            builder: (context) =>
                                                const ScanerScreen()),
                                      );
                                    },
                                    style: ButtonStyle(
                                        shape:
MaterialStateProperty.all<

RoundedRectangleBorder>(
                                            RoundedRectangleBorder(
                                          borderRadius:

BorderRadius.circular(25.0.r),
                                        )),
                                        padding:

MaterialStatePropertyAll<EdgeInsets>(
                                                const
EdgeInsets.all(13).r),
                                        backgroundColor:
                                            const
MaterialStatePropertyAll<Color>(
                                                Colors.yellow)),
                                    child: Row(
                                      mainAxisAlignment:
MainAxisAlignment.center,
                                      children: [
                                        const Icon(
                                          Icons.qr_code_scanner_sharp,
                                          color: Colors.black,
                                        ),
                                        SizedBox(
                                          width: 10.r,
                                        ),
                                        const Text("Scan now",
                                            style: TextStyle(
```

```
                                        color: (Colors.black),
                                      )),
                                  ],
                                ))),
                          ],
                        ),
                      ),
                    ) // Foreground widget here
                    ),
              ),
              const Expanded(child: SizedBox()),
              const SearchPanel()
            ],
          )));
  }
}
```

scaner_screen.dart

```
import 'dart:io';

import 'package:camera/camera.dart';
import 'package:flutter/foundation.dart';
import 'package:flutter/material.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';
import 'package:image_picker/image_picker.dart';
import 'package:recipe_app/screens/picture_screen.dart';
import 'package:recipe_app/widgets/top_bar.dart';

import '../main.dart';
import '../providers/vision_provider.dart';

class ScanerScreen extends StatefulWidget {
  const ScanerScreen({super.key});

  @override
  State<ScanerScreen> createState() => _ScanerScreenState();
}

class _ScanerScreenState extends State<ScanerScreen> {
  CameraController? controller;
  late XFile? _image = null;
  bool _isCameraInitialized = false;
  bool scanning = false;

  void onNewCameraSelected(CameraDescription cameraDescription) async
{
    final previousCameraController = controller;
    // Instantiating the camera controller
    final CameraController cameraController = CameraController(
      cameraDescription,
      ResolutionPreset.high,
```

```
      imageFormatGroup: ImageFormatGroup.jpeg,
      enableAudio: false,
    );

    // Dispose the previous controller
    await previousCameraController?.dispose();

    // Replace with the new controller
    if (mounted) {
      setState(() {
        controller = cameraController;
      });
    }

    // Update UI if controller updated
    cameraController.addListener(() {
      if (mounted) setState(() {});
    });

    // Initialize controller
    try {
      await cameraController.initialize();
    } on CameraException catch (e) {
      print('Error initializing camera: $e');
    }

    // Update the Boolean
    if (mounted) {
      setState(() {
        _isCameraInitialized = controller!.value.isInitialized;
      });
    }
  }

  @override
  void initState() {
    //       SystemChrome.setEnabledSystemUIMode(SystemUiMode.manual,
overlays: []);
    onNewCameraSelected(cameras[0]);
    super.initState();
  }

  @override
  void dispose() {
    controller?.dispose();
    super.dispose();
  }

  void didChangeAppLifecycleState(AppLifecycleState state) {
    final CameraController? cameraController = controller;
```

```
    if          (cameraController          ==          null          ||
!cameraController.value.isInitialized) {
      return;
    }

    if (state == AppLifecycleState.inactive) {
      // Free up memory when camera not active
      cameraController.dispose();
    } else if (state == AppLifecycleState.resumed) {
      // Reinitialize the camera with same properties
      onNewCameraSelected(cameraController.description);
    }
  }

  Future<void> _getImage() async {
    final imagePicker = ImagePicker();
    final pickedImage =
        await imagePicker.pickImage(source: ImageSource.gallery);

    if (pickedImage != null) {
      setState(() {
        _image = pickedImage;
      });
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      floatingActionButtonLocation:
FloatingActionButtonLocation.centerFloat,
      body: _isCameraInitialized
          ? Stack(children: [
              _image != null
                  ? Container(
                      height: 1.sh,
                      child: Image.file(
                        File(_image!.path),
                        fit: BoxFit.cover,
                      ),
                    )
                  : Container(
                      child: controller!.buildPreview(),
                    ),
              Positioned(
                  top: 30.h,
                  left: 10.w,
                  right: 10.w,
                  child: TopBar(
                    controller: controller,
                    title: "Scanner",
                    rightIcon: true,
```

```
                )),
            Padding(
              padding: EdgeInsets.all(20.r),
              child: Align(
                alignment: Alignment.bottomCenter,
                child: FloatingActionButton(
                  heroTag: "btn1",
                  backgroundColor:
Theme.of(context).colorScheme.background,
                  shape: RoundedRectangleBorder(
                      side: const BorderSide(width: 2, color:
Colors.black26),
                      borderRadius: BorderRadius.circular(100)),
                  onPressed: () async {
                    try {
                      _isCameraInitialized;

                      setState(() {
                        scanning = true;
                      });

                      final image = await
controller?.takePicture();

                      if (!mounted) return;

                      final scanResult = await visionImage(
                          file: _image != null
                              ? File(_image!.path)
                              : File(image!.path));

                      setState(() {
                        scanning = false;
                      });

                      await Navigator.of(context).push(
                        MaterialPageRoute(
                          builder: (context) => PictureScreen(
                            imagePath:
                                _image != null ? _image!.path :
image!.path,
                            scanResult: scanResult,
                          ),
                        ),
                      );
                    } catch (e) {
                      if (kDebugMode) {
                        print(e);
                      }
                    }
                  },
                  child: const Icon(Icons.camera_alt),
```

```
                ),
              ),
            ),
            Padding(
              padding: EdgeInsets.all(20.r),
              child: Align(
                alignment: Alignment.bottomLeft,
                child: FloatingActionButton(
                  heroTag: "btn2",
                  backgroundColor:
Theme.of(context).colorScheme.background,
                  shape: RoundedRectangleBorder(
                      side:  const  BorderSide(width:  2,  color:
Colors.black26),
                      borderRadius: BorderRadius.circular(100)),
                  onPressed: _getImage,
                  tooltip: 'Pick Image',
                  child: Icon(Icons.add_a_photo),
                ),
              ),
            ),
            scanning
                ? Container(
                    color: Colors.black.withOpacity(0.5),
                    child: const Center(
                      child: CircularProgressIndicator(),
                    ),
                  )
                : Container()
          ])
        : Container(),
    );
  }
}
```

picture_screen.dart

```
import 'dart:io';

import 'package:flutter/foundation.dart';
import 'package:flutter/material.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';
import 'package:recipe_app/models/vision_model.dart';
import 'package:recipe_app/screens/recipes_screen.dart';
import 'package:recipe_app/widgets/labels_list.dart';
import 'package:recipe_app/widgets/text_on_image.dart';
import 'package:recipe_app/widgets/top_bar.dart';

class PictureScreen extends StatefulWidget {
  final String imagePath;
  final Vision scanResult;
```

```
  const PictureScreen(
      {super.key,       required       this.imagePath,       required
this.scanResult});

  @override
  State<PictureScreen> createState() => _PictureScreenState();
}

class _PictureScreenState extends State<PictureScreen> {
  late List<String> selectedProducts;

  @override
  void initState() {
    selectedProducts = widget.scanResult.uniqueNames;
    super.initState();
  }

  void handleProductTap(String product) {
    if (selectedProducts.contains(product)) {
      setState(() {
        selectedProducts.remove(product);
      });
    } else {
      setState(() {
        selectedProducts.add(product);
      });
    }
  }

  @override
  Widget build(BuildContext context) {
    String namesString = selectedProducts.join(', ');
    List<String> labels = widget.scanResult.labelsObjects;

    return Scaffold(
      body: Stack(children: [
        SizedBox(
            height: 1.sh,
            width: 1.sw,
            child: kIsWeb
                ? Image.network(widget.imagePath)
                : Image.file(
                    File(widget.imagePath),
                    fit: BoxFit.cover,
                  )),
        for (var obj in widget.scanResult.objectData)
          TextOnImage(
              x:   obj.coordinates.x,   y:   obj.coordinates.y,   name:
obj.name),
        Positioned(
            top: 30.h,
            left: 10.w,
```

```
            right: 10.w,
            child: const TopBar(
              rightIcon: false,
              title: "Recipe Scanner",
            )),
        Align(
          alignment: Alignment.centerLeft,
          child: Container(
              margin: EdgeInsets.only(left: 5.w),
              width: 0.35.sw,
              child: LabelsList(
                labels: labels,
                handleProductTap: handleProductTap,
              )),
        ),
        Padding(
          padding: const EdgeInsets.all(20).r,
          child: Align(
            alignment: Alignment.bottomCenter,
            child: Column(
              mainAxisAlignment: MainAxisAlignment.end,
              children: [
                Container(
                  padding: EdgeInsets.all(15.r),
                  decoration: BoxDecoration(
                      borderRadius:
BorderRadius.all(Radius.circular(20.r)),
                      border: Border.all(color: Colors.white, width:
1.r),
                      color: Theme.of(context)
                          .colorScheme
                          .secondary
                          .withOpacity(0.6)),
                  child: Text(
                    namesString,
                    style:   const   TextStyle(color:   Colors.white,
fontSize: 14),
                  ),
                ),
                SizedBox(
                  height: 10.h,
                ),
                FloatingActionButton(
                  backgroundColor:
Theme.of(context).colorScheme.background,
                  shape: RoundedRectangleBorder(
                      side:   const   BorderSide(width:   2,   color:
Colors.black26),
                      borderRadius: BorderRadius.circular(100)),
                  onPressed: () async {
                    Navigator.of(context).push(
                      MaterialPageRoute(
```

```
                            builder: (context) => RecipesScreen(
                                imagePath: widget.imagePath,
                                ingredients: selectedProducts)),
                      );
                    },
                    //    onPressed:    ()    =>    visionImage(file:
File(imagePath)),
                    child: const Icon(Icons.all_inclusive_outlined),
                  ),
                ],
              ),
            ),
          )
        ]),
    );
  }
}
```

recipes_screen.dart

```
import 'dart:io';

import 'package:flutter/foundation.dart';
import 'package:flutter/material.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';
import 'package:recipe_app/widgets/sliding_panel.dart';
import 'package:sliding_up_panel/sliding_up_panel.dart';

import '../models/recipe_model.dart';
import '../providers/recipes_provider.dart';
import '../widgets/app_icon.dart';

class RecipesScreen extends StatefulWidget {
  final String imagePath;
  final List<String> ingredients;

  const RecipesScreen(
      {super.key,       required       this.imagePath,       required
this.ingredients});

  @override
  State<RecipesScreen> createState() => _RecipesScreenState();
}

class _RecipesScreenState extends State<RecipesScreen> {
  late Future<List<Recipe>> _recipes;
  late int _offset = 0;

  @override
  void initState() {
    super.initState();
    _recipes = fetchRecipes(widget.ingredients, 5, _offset);
```

```
  }

  void refreshRecipes() {
    int offset = _offset + 5;
    Future<List<Recipe>> newRecipes =
        fetchRecipes(widget.ingredients, 5, offset);

    setState(() {
      _offset = offset;
      _recipes = newRecipes;
    });
  }

  @override
  Widget build(BuildContext context) {
    final PanelController panelController = PanelController();

    return Scaffold(
      body: Stack(children: [
        SlidingUpPanel(
          backdropEnabled: true,
          color: Theme.of(context).colorScheme.background,
          minHeight: 250.h,
          controller: panelController,
          body: kIsWeb
              ? Image.network(widget.imagePath)
              : Image.file(
                  File(widget.imagePath),
                  fit: BoxFit.cover,
                ),
          panelBuilder: (controller) => Padding(
            padding: const EdgeInsets.only(left: 10, right: 10, top:
10).r,
            child: SlidingPanel(
              panelController: panelController,
              scrollController: controller,
              recipes: _recipes,
            ),
          ),
          borderRadius:       const       BorderRadius.vertical(top:
Radius.circular(30)),
        ),
        Positioned(
            top: 30.h,
            left: 10.w,
            right: 10.w,
            child: Container(
              padding:     EdgeInsets.symmetric(horizontal:     10.r,
vertical: 10.r),
              width: 1.sw,
              child: Row(
                mainAxisAlignment: MainAxisAlignment.spaceBetween,
```

```
                children: [
                  AppIcon(
                    icon: Icons.arrow_back_ios_sharp,
                    opacity: 0.4,
                    onPressed: () {
                      Navigator.of(context).pop();
                    },
                  ),
                ],
              ),
            )),
        Padding(
          padding: const EdgeInsets.all(20).r,
          child: Align(
            alignment: Alignment.bottomCenter,
            child: FloatingActionButton(
              backgroundColor: Colors.yellow,
              shape: RoundedRectangleBorder(
                  side:    const    BorderSide(width:    4,    color:
Colors.white),
                  borderRadius: BorderRadius.circular(100)),
              onPressed: () async {
                refreshRecipes();
              },
              // onPressed: () => visionImage(file: File(imagePath)),
              child: const Icon(Icons.repeat_sharp),
            ),
          ),
        ),
      ]),
    );
  }
}
```

recipe_screen.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_screenutil/flutter_screenutil.dart';
import 'package:recipe_app/utils/favorite_recipes_manager.dart';
import 'package:recipe_app/widgets/medium_text.dart';

import '../constants.dart';
import '../models/recipe_model.dart';

class RecipeScreen extends StatelessWidget {
  final Recipe recipe;
  const RecipeScreen({super.key, required this.recipe});

  @override
  Widget build(BuildContext context) {
    String ingredientsString =
        recipe.ingredients.replaceAll(RegExp(r"^\['|'\]$"), '');
```

```
    List<String> ingredientsList = ingredientsString.split('\', \'');

    return Scaffold(
      body: CustomScrollView(
        slivers: [
          SliverAppBar(
            bottom: PreferredSize(
              preferredSize: Size.fromHeight(70.h),
              child: Container(
                  padding: EdgeInsets.all(15.r),
                  width: double.maxFinite,
                  decoration: BoxDecoration(
                      color: Colors.white,
                      borderRadius: BorderRadius.only(
                          topLeft: Radius.circular(30.r),
                          topRight: Radius.circular(30.r))),
                  child: Center(
                    child: MediumText(
                      text: recipe.title,
                      color: Colors.black,
                    ),
                  )),
            ),
            pinned: true,
            actions: [
              IconButton(
                icon: const Icon(Icons.favorite_border),
                onPressed: () async {
                  final favoritesManager = FavoriteRecipesManager();
                  await favoritesManager.addFavoriteRecipe(recipe);
                },
              ),
            ],
            expandedHeight: 300.h,
            flexibleSpace: FlexibleSpaceBar(
              // title: MediumText(text: recipe.title),
              background: Container(
                decoration: BoxDecoration(
                    image: DecorationImage(
                        fit: BoxFit.cover,
                        image: NetworkImage(

"${ApiConstants.baseUrl}/${recipe.imageName}.jpg"))),
              ),
            ),
          ),
          SliverToBoxAdapter(
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                Container(
                  padding: EdgeInsets.all(15.r),
```

```
                  width: double.maxFinite,
                  color: Colors.black12,
                  child: const MediumText(
                    text: "Ingredients",
                    color: Colors.black,
                  ),
                ),
                Padding(
                  padding: EdgeInsets.all(15.r),
                  child: Column(
                    crossAxisAlignment: CrossAxisAlignment.start,
                    children: ingredientsList.map((ingredient) {
                      return Padding(
                        padding: EdgeInsets.symmetric(vertical: 1.h),
                        child: Text('● $ingredient'),
                      );
                    }).toList(),
                  ),
                ),
                Container(
                  padding: EdgeInsets.all(15.r),
                  width: double.maxFinite,
                  color: Colors.black12,
                  child: const MediumText(
                    text: "Instructions",
                    color: Colors.black,
                  ),
                ),
                Padding(
                  padding: EdgeInsets.all(15.r),
                  child: Text(
                    recipe.instructions,
                    textAlign: TextAlign.justify,
                    style:  const  TextStyle(fontSize:  18.0,  height:
1.4),
                  ),
                ),
              ],
            ),
          ),
        ],
      ),
    );
  }
}
```

vision_provider.dart

```
import 'dart:io';
import 'package:http_parser/http_parser.dart';
import 'package:recipe_app/constants.dart';
import 'package:recipe_app/models/vision_model.dart';
```

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

Dio dio = Dio();
Future<Vision> visionImage({required File file}) async {
  var formData = FormData.fromMap({
    'file': await MultipartFile.fromFile(file.path,
        contentType: MediaType('image', 'jpg')),
  });

  final response = await Dio()

.post('${ApiConstants.baseUrl}${ApiConstants.visionEndpoint}/image',
          data: formData,
          options: Options(headers: {
            "Content-type": "multipart/form-data",
          }));

  if (response.statusCode == 200) {
    // Перевірка на успішну відповідь з серверу (код 200)
    Vision result = Vision.fromJson(
        response.data);
    print(result.uniqueNames.toString());
    return result;
  } else {
    throw Exception('Помилка запиту на візуальне розпізнавання');
  }
}
```

recipes_provider.dart

```
import 'package:recipe_app/constants.dart';
import 'package:recipe_app/models/recipe_model.dart';
import 'package:dio/dio.dart';

Dio dio = Dio();
Future<List<Recipe>> fetchRecipes(
    List<String> ingredients, int limit, int offset) async {
  final response = await Dio().get(
      '${ApiConstants.baseUrl}${ApiConstants.recipeEndpoint}',
      queryParameters: {
        'ingredients': ingredients,
        'limit': limit,
        'offset': offset
      });

  if (response.statusCode == 200) {
    // Перевірка на успішну відповідь з серверу (код 200)
    final List<dynamic> data = response.data;

    return data.map((json) => Recipe.fromJson(json)).toList();
    ; // Парсинг відповіді у об'єкт Vision
  } else {
```

```
    throw Exception('Помилка отримання рецептів');
  }
}
```

favorite_recipes_manager.dart

```
import 'dart:convert';

import 'package:shared_preferences/shared_preferences.dart';

import '../models/recipe_model.dart';

class FavoriteRecipesManager {
  static const _favoritesKey = 'favorite_recipes';

  Future<List<Recipe>> getFavoriteRecipes() async {
    final prefs = await SharedPreferences.getInstance();
    final jsonString = prefs.getString(_favoritesKey);
    if (jsonString != null) {
      List<Recipe> favoriteRecipes = (json.decode(jsonString) as List)
          .map((i) => Recipe.fromJson(i))
          .toList();
      return favoriteRecipes;
    }
    return [];
  }

  Future<void> addFavoriteRecipe(Recipe recipe) async {
    final prefs = await SharedPreferences.getInstance();
    final favoriteRecipes = await getFavoriteRecipes();
    favoriteRecipes.add(recipe);
    final jsonString =
        json.encode(favoriteRecipes.map((recipe)                    =>
recipe.toJson()).toList());
    prefs.setString(_favoritesKey, jsonString);
  }

  Future<void> removeFromFavorites(Recipe recipe) async {
    final prefs = await SharedPreferences.getInstance();
    final favoriteRecipes = await getFavoriteRecipes();
    favoriteRecipes.removeWhere((jsonString) {
      return jsonString.title == recipe.title;
    });
    final jsonString =
        json.encode(favoriteRecipes.map((recipe)                    =>
recipe.toJson()).toList());
    prefs.setString(_favoritesKey, jsonString);
  }
}
```

main.py

```
import pandas as pd
from flask import Flask
from flask import request
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

app = Flask(__name__)

@app.route('/')
def home():
    dataset_path = "recipes.csv"
    df = pd.read_csv(dataset_path)

    user_input_ingredients = request.args.getlist('ingredients')
    number_recommendations = request.args.get('limit')
    offset = request.args.get('offset')

    recommended_recipes = recommend_recipes(df,
user_input_ingredients, offset,
num_recommendations=number_recommendations)

    return recommended_recipes

def recommend_recipes(data, user_input_ingredients, offset,
num_recommendations=5):

    tfidf_vectorizer = TfidfVectorizer()
    tfidf_matrix =
tfidf_vectorizer.fit_transform(data['Ingredients'])

    user_input_vector = tfidf_vectorizer.transform(['
'.join(user_input_ingredients)])
    similarities = cosine_similarity(user_input_vector, tfidf_matrix)


    # Визначення найбільш схожих рецептів
    top_recipe_indices = similarities.argsort()[0][::-
1][int(num_recommendations):int(offset)+int(num_recommendations)]
    top_recipes = data.loc[top_recipe_indices, ['Title',
'Ingredients', 'Instructions', 'Image_Name']]

    json_data = top_recipes.to_dict(orient="records")

    return json_data

# main driver function
if __name__ == '__main__':
     app.run(host='0.0.0.0', port=8080)
```