

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МОБІЛЬНОГО
ЗАСТОСУНКУ ДЛЯ ОБЛІКУ ХАРЧУВАННЯ
RESEARCH AND SOFTWARE IMPLEMENTATION OF A MOBILE
APPLICATION FOR NUTRITION TRACKING

Виконала: студентка 2 курсу, групи КНТ-114м

Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ПАНИЧУК К.С.

(ПРИЗВИЩЕ та ініціали)

Керівник ОЛІЙНИК А.О.

(ПРИЗВИЩЕ та ініціали)

Рецензент СКРУПСЬКИЙ С.Ю.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
« » 2025 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТКИ

ПАНИЧУК Катерини Сергіївни

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація мобільного застосунку для обліку харчування. Research and Software Implementation of a Mobile Application for Nutrition Tracking

керівник проєкту (роботи) д.т.н., професор, ОЛІЙНИК Андрій Олександрович,

(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від «30» вересня 2025 року №447

2. Строк подання студентом проєкту (роботи) 08 грудня 2025 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Вибір та обґрунтування структури системи, яка проєктується. 3. Основні рішення з реалізації компонентів системи. 4. Керівництво програміста. 5. Керівництво оператора. 6. Тестування і результати.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-6 Основна частина	ОЛІЙНИК А.О., професор		
Нормоконтролер	ДЕЙНЕГА Л.Ю., ст. викл.		

7. Дата видачі завдання «30» вересня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	2-3 тижні	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	4 тиждень	Розділ 2
4	Розробка архітектури програми.	5 тиждень	Розділ 3
5	Розробка програми.	6-7 тижні	Розділи 3-6
6	Створення керівництв програміста та оператора.	8 тиждень	Розділи 4, 5
7	Тестування та експериментальне дослідження програмного забезпечення.	9 тиждень	Розділ 6
8	Оформлення пояснювальної записки та документів до неї.	10-11 тижні	Додатки
9	Нормоконтроль та рецензування.	12 тиждень	
10	Захист роботи.	12тиждень	

Студентка

Підпис
(підпис)

Катерина ПАНИЧУК
(Ім'я, ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис)

Андрій ОЛІЙНИК
(Ім'я, ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
152 с., 22 табл., 73 рис., 4 дод., 40 джерел.

ОБЛІК, ЇЖА, ПРИЙМАННЯ ЇЖІ, КАЛОРІЙНІСТЬ,
РОЗПІЗНАВАННЯ, МОБІЛЬНИЙ ЗАСТОСУНОК, .NET MAUI, PYTHON.

Об'єкт дослідження – процес розробки програмного забезпечення для обліку харчування.

Предмет дослідження – програмне забезпечення для обліку харчування.

Мета роботи – створення програмного забезпечення, яке дозволяє користувачеві вести облік харчування, отримувати дані про спожиту їжу, здійснювати розпізнавання їжі, зчитувати фізіологічні дані з розумних ваг.

Результати. Створено мобільний застосунок, який дозволяє вести облік харчування, створювати фотографії страв для розпізнавання, отримувати дані про продукти з відкритої бази Open Food Facts за штрихкодом, зчитувати дані з розумних ваг Mi Body Composition Scale 2.

Наукова новизна полягає у комплексному поєднанні сучасних методів комп'ютерного зору, мобільної розробки та засобів криптографічного захисту даних для забезпечення процесу обліку харчування.

Висновки. Створено модифіковану модель згорткової нейронної мережі MealNet для розпізнавання їжі, швидкість навчання якої серед 6 моделей є найкращою (741,4 с. для 10 класів, 3120,6 с. – 30, 6127,5 с. – 60, 9602,2 с. – 101). Модель ResNet50V2 продемонструвала найкращу точність тестування (71% для 10 класів, 48% – 101). Впроваджено методи шифрування AES, RSA та комбінацію цих алгоритмів, яка виявилась найкращою.

Сфера застосування – будь-яка сфера, де користувач має потребу в контролі харчування та калорійності.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 152 pages, 22 tables, 73 figures, 4 appendixes, 40 sources.

ACCOUNTING, FOOD, MEAL INTAKE, CALORIC VALUE, RECOGNITION, MOBILE APPLICATION, .NET MAUI, PYTHON.

The object of study is the process developing software for dietary accounting.

The subject of study is software designed for dietary accounting.

The purpose of this work is to develop software that enables the user to track food intake, obtain information about consumed products, perform food recognition, and read physiological data from smart scales.

Results. A mobile application was created that allows users to track their nutrition, take photos of meals for recognition, retrieve product data from the Open Food Facts database via barcode scanning, and receive data from the Mi Body Composition Scale 2 smart scales.

The scientific novelty lies in the comprehensive combination of modern methods of computer vision, mobile development, and cryptographic data protection tools to ensure the process of nutrition tracking.

Conclusions. A modified convolutional neural network model, MealNet, was created for food recognition, demonstrating the best training speed among six evaluated models (741.4 s for 10 classes, 3120.6 s for 30, 6127.5 s for 60, 9602.2 s for 101). The ResNet50V2 model achieved the highest testing accuracy (71% for 10 food classes and 48% for 101). AES, RSA, and a hybrid combination of these encryption algorithms were implemented, with the hybrid approach proving to be the most effective.

The scope of application is any area where users need to monitor nutrition and caloric intake.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	9
Вступ.....	10
1 Аналіз предметної області.....	11
1.1 Огляд відомих методів обліку харчування.....	11
1.2 Огляд наявних програмних засобів.....	12
1.3 Огляд відомих методів машинного навчання	16
1.4 Постановка завдання роботи.....	20
1.5 Висновки за розділом 1	22
2 Вибір та обґрунтування структури системи, яка проєктується.....	23
2.1 Аналіз вимог	23
2.2 Розробка архітектури системи.....	24
2.3 Вибір мови програмування	25
2.4 Порівняння мов програмування	35
2.5 Вибір середовища розробки.....	36
2.6 Порівняння середовищ розробки	42
2.7 Вибір системи управління базами даних.....	43
2.8 Порівняння систем управління базами даних.....	48
2.9 Висновки за розділом 2	48
3 Основні рішення з реалізації компонентів системи	50
3.1 Вибір шаблону проєктування	50
3.2 Опис модулів розроблюваної системи.....	52
3.3 Створення модифікованого методу для розпізнавання їжі	55
3.4 Проєктування бази даних	69

3.5 Реалізація взаємодії з відкритою базою даних Open Food Facts	71
3.6 Реалізація взаємодії з розумними вагами Xiaomi Mi Body Composition Scale 2	72
3.7 Основні рішення щодо шифрування зображень	73
3.8 Основні рішення щодо розробки інтерфейсу	80
3.9 Обробка виняткових ситуацій	83
3.10 Висновки за розділом 3	84
4 Керівництво програміста	85
4.1 Призначення та умови застосування програми	85
4.2 Характеристика програми	86
4.3 Звертання до програми	87
4.4 Вхідні та вихідні дані	88
4.5 Повідомлення програмісту	88
4.6 Висновки за розділом 4	90
5 Керівництво оператора	91
5.1 Призначення програми	91
5.2 Умови виконання програми	91
5.3 Виконання програми	91
5.4 Повідомлення оператору	98
5.5 Висновки за розділом 5	100
6 Тестування і результати	101
6.1 Опис методики проведення тестування	101
6.2 Результати використання розробленого програмного забезпечення ..	102
6.3 Висновки за розділом 6	108
Висновки	109

Перелік джерел посилання	111
Додаток А Технічне завдання	116
Додаток Б Опис програми	128
Додаток В Текст програми	135
Додаток Г Слайди презентації	147

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

ACID	– Atomicity, Consistency, Isolation, Durability;
AES	– Advanced Encryption Standard;
API	– Application Programming Interface;
CFB	– Cipher Feedback;
EAN	– European Article Number;
GC	– Garbage Collection;
GPS	– Global Positioning System;
HTTP	– Hypertext Transfer Protocol;
IDE	– Integrated Development Environment;
JSON(B)	– JavaScript Object Notation (Binary);
JVM	– Java Virtual Machine;
LINQ	– Language Integrated Query;
MVVM	– Model-View-ViewModel;
RCP	– Rich Client Platform;
REST	– Representational State Transfer;
RSA	– Rivest, Shamir, Adleman;
SDK	– Software Development Kit;
SQL	– Structured Query Language;
SVM	– Support Vector Machines;
UI/UX	– User Interface/User Experience;
URL	– Uniform Resource Locator;
UUID	– Universally Unique Identifier;
.NET MAUI	– .NET Multi-Platform App UI;
БД	– база даних;
ЗНМ	– згортова нейронна мережа;
ОС	– операційна система;
СУБД	– система управління базами даних;
⊕	– операція виключне АБО.

ВСТУП

У сучасному світі, де технології стрімко інтегруються у повсякденне життя, питання контролю харчування та ведення здорового способу життя стає дедалі актуальнішим. Зростання темпів життя, нерегулярне харчування та поширення нездорових звичок створюють нові виклики для збереження фізичного та психологічного здоров'я людей. Важливим аспектом підтримки здорового способу життя є контроль за споживаною їжею та розуміння її харчової цінності [1].

Одним із можливих шляхів вирішення цих проблем є створення зручного та інтуїтивно зрозумілого мобільного застосунку, який дозволяє не лише вести облік приймань їжі, а й автоматизувати процеси збору та обробки даних за допомогою розпізнавання їжі за фотографією, отримання інформації з відкритої бази даних, а також синхронізації з розумними вагами.

Актуальність цієї теми обумовлена також зростанням кількості хронічних захворювань, пов'язаних із харчуванням, таких як ожиріння, діабет, серцево-судинні захворювання. Інструменти, які допомагають людям краще контролювати своє харчування, можуть відіграти важливу роль у зниженні цих ризиків та сприяти формуванню здорових звичок [1].

Мета цієї роботи – створення мобільного застосунку, що дозволяє користувачеві зручно вести облік харчування, отримувати інформацію про спожиту їжу та контролювати фізичний стан тіла. Така система може стати цінним інструментом як для звичайних користувачів, так і для закладів громадського харчування, медичних закладів, фахівців фітнес-індустрії та дієтологів. Розробка також сприяє інтеграції новітніх технологій у сферу охорони здоров'я та харчування, створюючи передумови для подальшого розвитку інноваційних рішень, що покращують якість життя.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Облік харчування — це процес фіксації кількості та якості спожитої їжі з метою контролю за здоров'ям, досягнення фітнес-цілей або медичного моніторингу [2]. Цей процес охоплює збір даних про тип продукту, його харчову цінність, час і обсяг приймання їжі. У поєднанні з даними про фізіологічні показники, такими як вага, індекс маси тіла або опір тіла, він дозволяє більш точно оцінити вплив харчування на загальний стан організму. З розвитком мобільних технологій і доступом до відкритих харчових баз, з'явилася можливість автоматизувати облік, зробивши його зручнішим, швидшим і персоналізованим [3].

1.1 Огляд відомих методів обліку харчування

На сьогодні існує кілька підходів до ведення обліку харчування, які відрізняються за точністю, зручністю та рівнем автоматизації. Основними серед них є ручний запис у щоденниках або таблицях, використання мобільних застосунків із ручним введенням та автоматизований облік із використанням штрихкодів, зображень чи інтеграції з пристроями [3]. Кожен із цих методів має свої переваги та обмеження, що визначають доцільність використання в залежності від цілей користувача і наявних технічних засобів.

Ручний облік харчування передбачає запис усієї інформації про їжу в блокнот, таблицю Excel або інші текстові документи [3]. Цей метод не потребує спеціальних програм, однак вимагає значних зусиль і часу, не гарантує точності розрахунків та не забезпечує перевірку даних.

Облік за допомогою мобільних застосунків із ручним введенням є поширеним підходом серед користувачів [3]. Людина самостійно шукає продукти у базі, вказує кількість та час приймання їжі. Такий метод є точнішим і швидшим за ручний запис, проте все ще потребує активної участі користувача, а точність результату залежить від повноти харчової бази.

Автоматизований облік базується на використанні додаткових технологій — застосування штрихкодів, розпізнавання зображень, інтеграції з фітнес-пристроями та вагами тощо [3]. Це дозволяє мінімізувати ручне введення, підвищити точність даних і забезпечити персоналізовану аналітику. Проте такі методи потребують складнішого програмного забезпечення та доступу до відповідного обладнання.

Вище було наведено відомі методи обліку харчування. Вони мають свої переваги і недоліки. В табл. 1.1 наведено порівняння цих методів.

Таблиця 1.1 — Порівняння методів обліку харчування [3]

Метод	Точність даних	Зручність використання	Швидкість введення	Технічні вимоги
Ручний облік	Низька	Низька	Низька	Відсутні
Мобільні застосунки (ручне введення)	Середня	Середня	Середня	Смартфон
Автоматизований облік	Висока	Висока	Висока	Смартфон, камера, ваги

У межах роботи буде реалізовано мобільний застосунок, що базується на автоматизованому методі обліку харчування з можливістю ручного введення даних. Такий підхід дозволяє поєднати переваги високої точності та швидкості автоматизованих засобів — зокрема, розпізнавання їжі за зображенням, інтеграцію з розумними вагами та відкритою базою даних — із гнучкістю ручного введення, що залишається доступним для ситуацій, коли автоматичне визначення неможливе або потребує уточнення. Це зробить застосунок зручним та універсальним для щоденного використання.

1.2 Огляд наявних програмних засобів

У процесі розробки мобільного застосунку для обліку харчування важливо проаналізувати наявні рішення, що вже зарекомендували себе на ринку. Такі застосунки зазвичай поєднують функції підрахунку калорій,

ведення харчового щоденника, моніторингу фізичної активності та формування персоналізованих планів харчування [4]-[6]. Деякі з них використовують алгоритми штучного інтелекту для автоматизації введення даних і надання індивідуальних рекомендацій. Нижче розглянуто найбільш популярні й функціонально насичені приклади – Lifesum, FatSecret та DietAI.

Lifesum — це багатофункціональний застосунок (рис. 1.1) для здорового способу життя, який поєднує облік харчування, гідратації та фізичної активності [4]. Він пропонує персоналізовані плани харчування, понад 1000 рецептів та гнучкі програми для досягнення різних цілей, таких як схуднення, набір м'язової маси тощо [4]. Користувачі можуть додавати приймання їжі за допомогою фотографій, голосового або текстового введення, сканування штрихкодів. Унікальна функція «Life Score» оцінює загальний стан здоров'я на основі харчових звичок, гідратації та рівня активності, надаючи рекомендації [4]. Інтеграція з Google Health та Wear OS дозволяє автоматизувати облік фізичної активності. Базова версія Lifesum доступна безкоштовно, повний функціонал відкривається за підпискою.

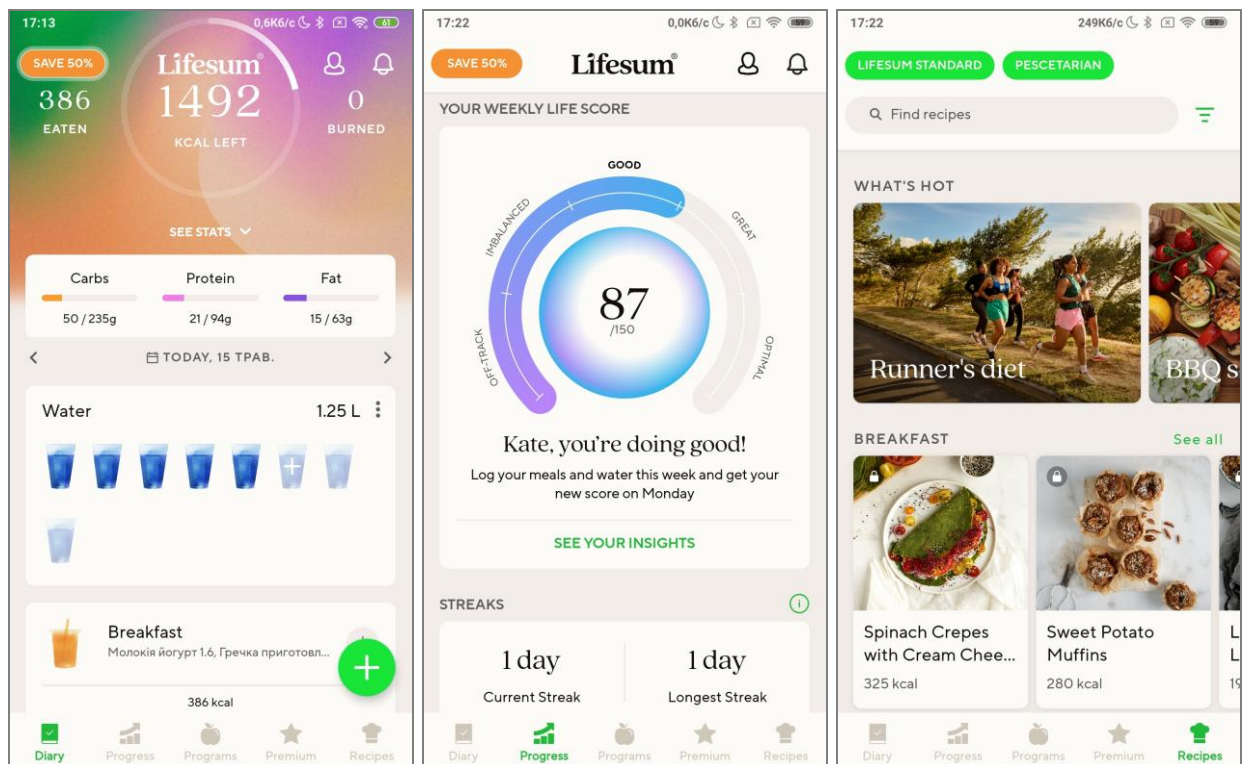


Рисунок 1.1 — Мобільний застосунок «Lifesum» [4]

FatSecret — це мобільний застосунок (рис. 1.2) для підрахунку калорій і макроелементів, який містить зручний щоденник харчування, календар дієти, трекер ваги та щоденник вправ [5]. Він підтримує сканування штрихкодів для швидкого додавання продуктів, а також інтегрується з Google Fit, Samsung Health і Fitbit [5]. Застосунок надає детальні звіти про споживання калорій і макроелементів, а також дозволяє користувачам приєднуватися до спільноти для отримання підтримки та мотивації [5]. У преміум-версії надаються персоналізовані плани харчування, розширене планування калорій та додаткові функції для контролю вживання води.

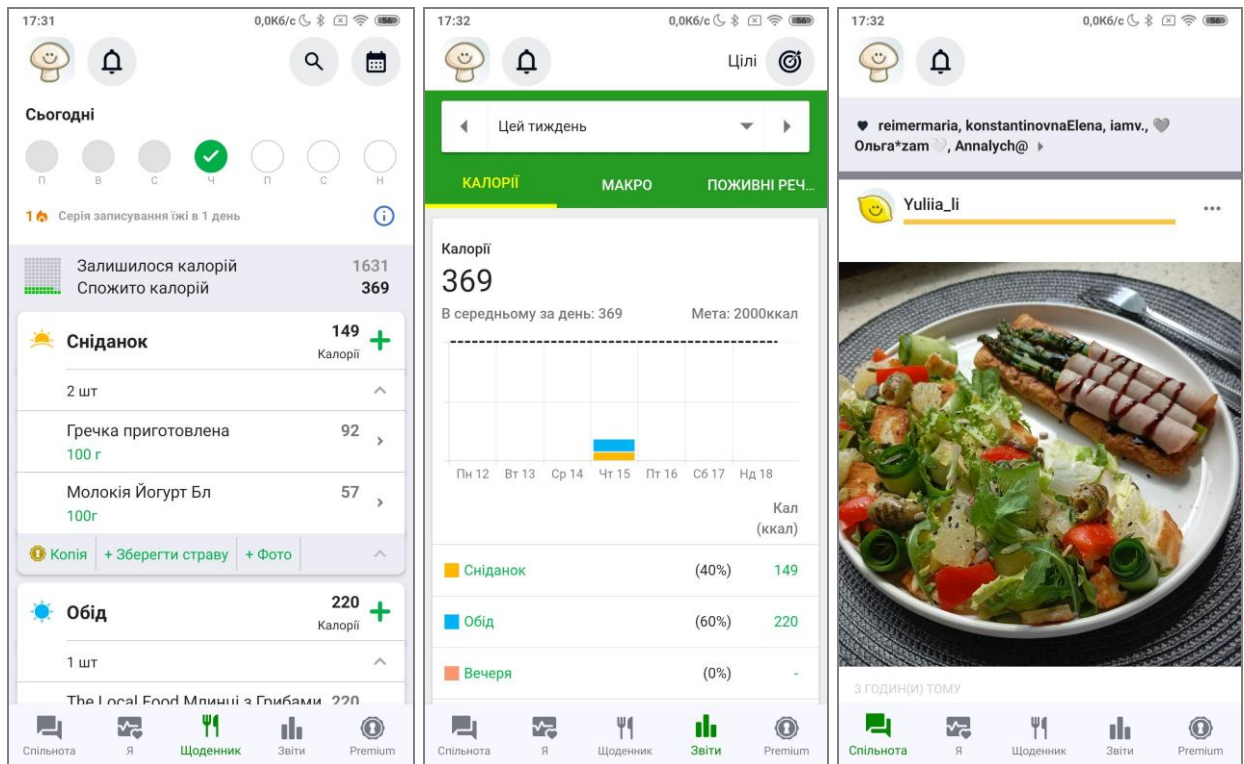


Рисунок 1.2 — Мобільний застосунок «FatSecret» [5]

DietAI — сучасний трекер харчування, який активно використовує штучний інтелект для покращення користувацького досвіду (рис. 1.3). Його ключова особливість — можливість додавання приймання їжі за допомогою фотографій або текстових описів, з автоматичним визначенням макро- та мікронутрієнтів [6]. Застосунок генерує персоналізовані плани дієт на основі індивідуальних параметрів користувача. Програма надає поглиблений аналіз

раціону, включаючи понад 25 поживних речовин, і підтримує рекомендації від віртуального дієтолога [6]. Також реалізовано інтеграцію з трекерами фізичної активності та відображення прогресу користувача через графіки та статистику. Більшість функцій доступні тільки за підпискою.

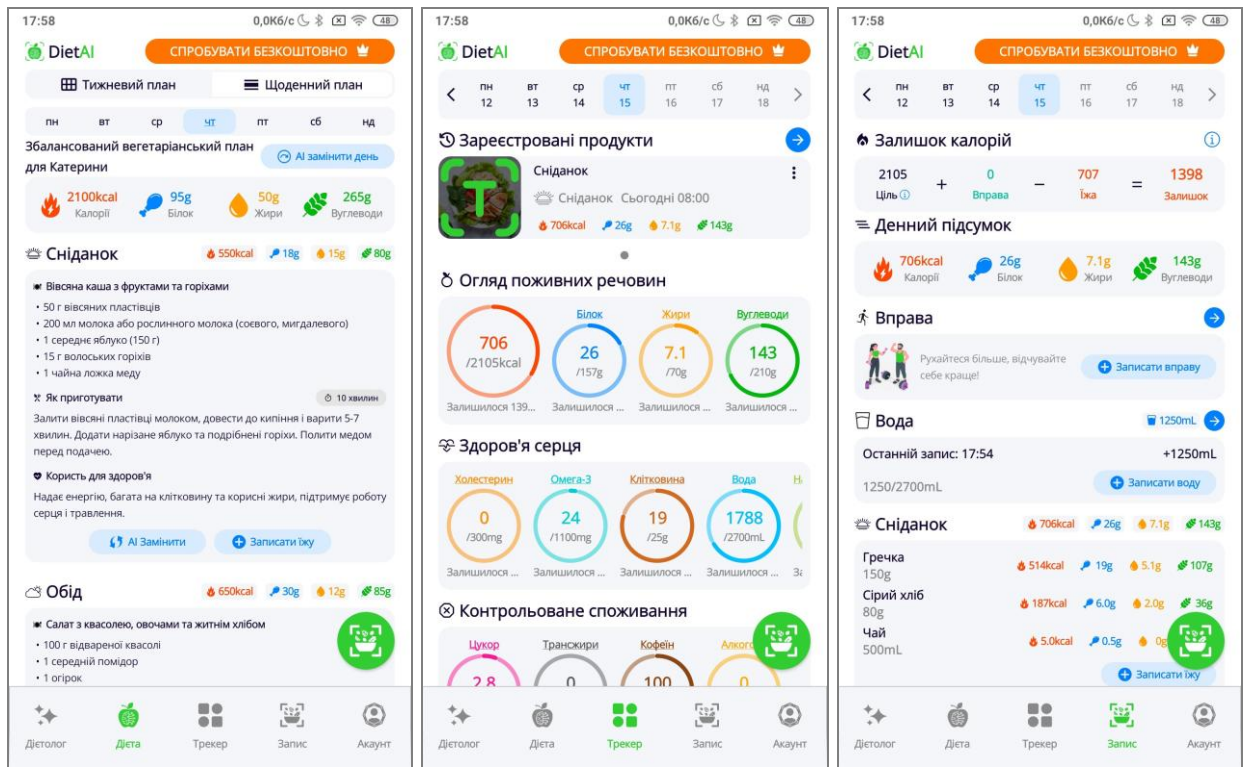


Рисунок 1.3 — Мобільний застосунок «DietAI» [6]

В таблиці 1.2 проведено порівняння наявних мобільних застосунків та програми, яка буде розроблена в ході виконання роботи.

Таблиця 1.2 — Порівняння мобільних застосунків [4]-[6]

Можливості	«Lifsum»	«FatSecret»	«DietAI»	Власний застосунок
1	2	3	4	5
Багатоплатформність	+	+	—	+
Українська локалізація	—	+	+	+
Робота з інформацією про приймання їжі	+	+	+	+
Робота з інформацією про користувача	+	+	+	+
Розпізнавання їжі за зображенням	—	—	+	+

Продовження таблиці 1.2

1	2	3	4	5
Використання баз даних	+	+	+	+
Відображення статистики	+	+	+	+
Використання штрихкодів продуктів	+	+	+	+
Взаємодія з розумними вагами	—	—	—	+
Підтримка дієт і планів харчування	+	+	+	—
Інтеграція з фітнес-трекерами	+	+	+	—
Трекер вживання води	+	+	+	—
Доступ до базової функціональності безкоштовно	+	+	—	+

На основі проведеного порівняння можна зробити висновок, що розроблюваний застосунок має низку унікальних функцій, які вирізняють його серед наявних аналогів. Зокрема, він єдиний з-поміж розглянутих буде підтримувати безпосередню взаємодію з розумними вагами Xiaomi Mi Body Composition Scale 2, що дозволяє отримувати розширені фізіологічні показники користувача. Також застосунок буде забезпечувати розпізнавання їжі за зображенням, що присутнє не у всіх аналогах, та використовувати відкриту базу даних для отримання інформації про продукти, що підвищує прозорість та гнучкість системи. Важливою перевагою також буде наявність української локалізації, чого бракує деяким наявним рішенням. Хоча у програмі не буде реалізовано підтримку фітнес-трекерів, трекер вживання води та планування дієт, її функціональні можливості добре узгоджуються із завданнями персоналізованого моніторингу харчування та фізіологічного стану користувача, що робить її конкурентоспроможною у цій сфері.

1.3 Огляд відомих методів машинного навчання

Традиційна класифікація – це ключовий підхід до аналізу даних, який фокусується на сортуванні точок даних за попередньо визначеними класами або категоріями з використанням певних правил і встановлених ознак [7]. До появи глибинного навчання для цього зазвичай застосовували різні

традиційні методи, такі як дерева рішень, метод опорних векторів (SVM), наївний Байєс та k-найближчих сусідів [7].

З розвитком глибинного навчання згорткові нейронні мережі стали потужною альтернативою, пропонуючи значний прогрес в обробці та аналізі даних зі складною структурою, наприклад, зображень і відеоматеріалів [7].

1.3.1 Дерево рішень

Дерево рішень – це ієрархічний, заснований на правилах, метод, який використовує непараметричний підхід [8]. Він визначає приналежність до класу шляхом рекурсивного поділу набору даних на однорідні підмножини. Як ієрархічний класифікатор, він дозволяє приймати або відхиляти класи на кожному проміжному кроці [8]. Процес складається з трьох основних етапів: розбиття вузлів, визначення кінцевих вузлів і присвоєння класів цим кінцевим вузлам [8].

1.3.2 Метод опорних векторів

Ядерні SVM неявно перетворюють вхідні вектори ознак у простір вищої розмірності за допомогою функції ядра, наприклад, ядра Гауса [9]. У цьому перетвореному просторі будується максимальна розділяюча гіперплощина, особливо для двокласової задачі [9]. Потім створюються дві паралельні гіперплощини симетрично по обидва боки від розділяючої гіперплощини [9]. Мета полягає в тому, щоб максимізувати відстань, відому як інтервал, між цими двома зовнішніми гіперплощинами [9]. Вважається, що чим більша відстань, тим менша помилка узагальнення класифікатора.

В основі SVM лежить принцип мінімізації структурного ризику, який спрямований на мінімізацію верхньої межі помилки узагальнення, на відміну від багатьох класифікаторів, які зосереджені на мінімізації емпіричного ризику, або помилки на навчальній вибірці [9].

Алгоритм SVM працює для пошуку цільової функції, яка мінімізує певний функціонал. Крім того, SVM здатні до навчання нелінійних класифікаторів у високорозмірних просторах навіть з невеликою навчальною вибіркою, завдяки вибору підмножини векторів, відомих як опорні вектори, які визначають оптимальні межі між класами [9].

1.3.3 Наївний байєсівський класифікатор

Наївний байєсівський класифікатор працює на ймовірнісній основі, присвоюючи клас з найвищою оціненою апостеріорною ймовірністю вектору ознак, отриманому з області інтересу [10]. Цей метод є оптимальним, коли атрибути є незалежними (ортогональними), але він добре працює навіть без цього припущення [10]. Його простота забезпечує високу продуктивність на невеликих навчальних вибірках, а завдяки побудові ймовірнісних моделей він залишається стійким до відхилень [10].

Крім того, наївний Байєс створює м'які межі рішень, допомагаючи уникнути перенавчання. Однак довільний вибір моделі розподілу для оцінки ймовірностей $P(x)$ та обмежена гнучкість меж прийняття рішень можуть знизити його ефективність у більш складних багатокласових задачах [10].

1.3.4 Класифікатор k-найближчих сусідів

Класифікатор k-найближчих сусідів визначає гіперсфери в просторі екземплярів, призначаючи клас більшості k-найближчих екземплярів на основі певної метрики [11]. Він є асимптотично оптимальним і дозволяє проводити швидке тестування. Однак, цей метод має декілька обмежень. Він дуже чутливий до фактору розмірності, оскільки збільшення розмірності має тенденцію до розсіювання простору ознак, що призводить до розосередження локальних однорідних областей, які представляють прототипи класів [12]. Ефективність класифікації сильно залежить від обраної метрики. Крім того,

вибір малого значення для k може призвести до нестабільних меж прийняття рішень, що робить класифікатор більш чутливим до відхилень [12].

1.3.5 Згорткова нейронна мережа

Згорткові нейронні мережі (ЗНМ) – це регуляризована форма багат шарових персептронів, які, як правило, є повнозв'язними мережами, де кожен нейрон в одному шарі пов'язаний з кожним нейроном у наступному [13]. ЗНМ відрізняються тим, що принаймні в одному з їхніх шарів замість стандартного матричного множення використовується математична операція, яка називається згортка [13]. Як тип нейронної мережі прямого поширення, ЗНМ добре підходять для обробки даних з сітчастою структурою. Вони вивчають особливості та закономірності даних за допомогою згорткових шарів. Нейрони в ЗНМ мають ваги та зсуви, що навчаються, де кожен нейрон обробляє вхідні дані, виконує скалярний добуток і може застосовувати нелінійність [14]. ЗНМ натхненні біологічними процесами в зоровій корі головного мозку і стали ключовим рішенням для завдань комп'ютерного зору в штучному інтелекті, таких як аналіз зображень і відео [14].

На відміну від традиційних методів, ЗНМ використовують багаторівневі архітектури для автоматичного навчання та вилучення ознак з необроблених вхідних даних, що значно покращує продуктивність у задачах візуального розпізнавання та виявлення шаблонів [7].

1.3.6 Порівняльна таблиця відомих методів машинного навчання

Для розпізнавання образів у проєкті було розглянуто кілька методів машинного навчання, які широко використовуються в задачах класифікації. У таблиці 1.3 наведено порівняння методів за основними характеристиками: точність, вимоги до обчислювальних ресурсів, здатність працювати з великою кількістю ознак і складність налаштування.

Таблиця 1.3 — Порівняння методів машинного навчання [7]-[14]

Метод	Точність	Обчислювальні ресурси	Робота з великою кількістю ознак	Складність налаштування
Дерево рішень	Середня	Низькі	Середня	Низька
Метод опорних векторів	Висока	Високі	Добра	Висока
Наївний Байєс	Середня	Низькі	Середня	Низька
k-найближчих сусідів	Середня	Середні	Погана	Низька
Згорткова нейронна мережа	Дуже висока	Високі	Відмінна	Висока

Згорткові нейронні мережі було обрано для реалізації розпізнавання їжі завдяки їхній здатності досягати дуже високої точності в задачах класифікації зображень. Вони добре працюють з великою кількістю ознак, автоматично виділяючи ключові особливості зображення, що є критичним для обробки складних даних, як-от фотографії їжі. Хоча ЗНМ вимагають високих обчислювальних ресурсів і складного налаштування, їх переваги в точності та універсальності значно переважають ці недоліки. Це робить їх найбільш підходящим інструментом для розробки інтелектуальної системи у сфері розпізнавання їжі.

1.4 Постановка завдання роботи

Проблема контролю харчування є надзвичайно актуальною в умовах сучасного стилю життя, коли більшість людей не мають змоги дотримуватись збалансованого раціону через брак часу або знань. Зростання кількості хронічних захворювань, пов'язаних з неправильним харчуванням, зумовлює потребу в інструментах, які б допомагали користувачам усвідомлено ставитися до свого раціону [1]. Наявні програмні рішення частково вирішують це завдання, однак не завжди враховують усі потреби користувачів. У цьому контексті розробка мобільного застосунку для персонального обліку харчування з підтримкою розпізнавання їжі, пошуку

інформації про продукти за штрихкодом та синхронізацією з розумними вагами є актуальною.

Мета роботи — створення програмного забезпечення, яке дозволяє користувачеві зручно вести облік харчування, отримувати інформацію про спожиту їжу, здійснювати розпізнавання їжі за зображеннями, а також отримувати фізіологічні дані з розумних ваг.

Для досягнення поставленої мети необхідно виконати такі завдання:

- вивчити наявний стек технологій;
- виконати проєктування програмного забезпечення (аналіз вимог, моделювання, розробку архітектури, вибір інструментарію розробки);
- виконати конструювання програмного забезпечення (алгоритм функціонування, структура програми, опис особливостей програмної реалізації, опис інтерфейсу);
- розробити сценарії тестування, провести досліdну експлуатацію програмного забезпечення.

Реалізація зазначених завдань дозволить створити сучасний, зручний та безпечний інструмент для контролю харчування та фізіологічного стану користувача з можливістю адаптації під індивідуальні потреби.

Для опису процесу роботи застосунку можна подати узагальнену модель системи у вигляді формули (1.1):

$$F = \{I, R, D, B, W, U, C\}, \quad (1.1)$$

де I – вхідні дані (дані про їжу та користувача, зображення їжі, штрихкоди, дані з ваг);

R – результати розпізнавання їжі нейронною мережею;

D – база даних продуктів харчування та їх харчової цінності;

W – оброблені дані з розумних ваг (ідеальна вага, м'язова маса тощо);

U – профіль користувача з переліком споживання;

C – компоненти користувацького інтерфейсу, що забезпечують взаємодію із системою.

Таким чином, програмний застосунок можна розглядати як функціональну систему, що перетворює множину вхідних даних I на вихідну інформацію про користувача та його харчування через взаємодію компонентів R, D, W, U та інтерфейсу C .

1.5 Висновки за розділом 1

В першому розділі було здійснено аналіз предметної області, обґрунтовано актуальність і доцільність створення програмного забезпечення. Розглянуто відомі методи виконання завдання обліку харчування, методи машинного навчання та мобільні застосунки, що забезпечують цей облік. У ході аналізу було виявлено їхні переваги й недоліки та визначено методи та функціонал, які буде реалізовано у розроблюваному застосунку.

2 ВИБІР ТА ОБҐРУНТУВАННЯ СТРУКТУРИ СИСТЕМИ, ЯКА ПРОЄКТУЄТЬСЯ

2.1 Аналіз вимог

На цьому етапі здійснюється аналіз вимог до програмного забезпечення. Формалізація функціональних і нефункціональних вимог дає змогу більш чітко визначити очікувану поведінку системи та сформувану основу для подальшого проєктування, реалізації та тестування застосунку.

Мобільний застосунок повинен забезпечувати виконання нижче перерахованих функцій:

- додавання, редагування, видалення інформації про приймання їжі;
- сортування переліку приймання їжі;
- виведення статистики калорійності приймань їжі;
- надання можливості зробити фотографію їжі, її шифрування, розшифрування та подальше розпізнавання;
- отримання даних про продукт з відкритої бази даних Open Food Facts за допомогою штрихкоду;
- редагування інформації про користувача;
- взаємодія з розумними вагами Xiaomi Mi Body Composition Scale 2 для отримання значень ваги та опору тіла, розрахунку показників тіла за даними про користувача.

До системи також висуваються такі нефункціональні вимоги:

- багатоплатформність – застосунок має коректно функціонувати на операційних системах Android та iOS;
- безпека даних користувача;
- зручність використання – інтерфейс має бути інтуїтивно зрозумілим;
- продуктивність – обробка фотографій та запити до бази даних і серверів мають виконуватись з мінімально можливою затримкою;
- підтримка української мови як основної мови інтерфейсу;

– масштабованість – структура застосунку має дозволяти подальше розширення функціональності без значних змін архітектури.

На рисунку 2.1 представлено діаграму прецедентів, яка ілюструє основні сценарії взаємодії користувача з системою. Кожен прецедент відповідає одній або декільком функціональним вимогам.

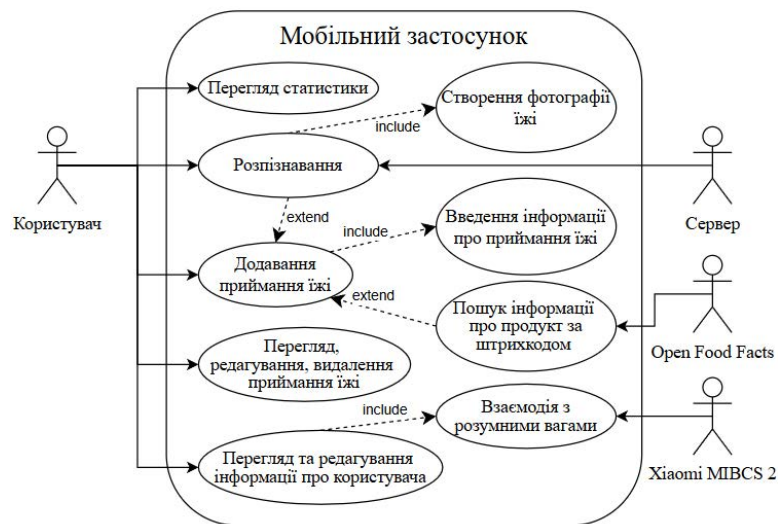


Рисунок 2.1 — Діаграма прецедентів

2.2 Розробка архітектури системи

Програмний комплекс буде складатися з двох основних частин: мобільного застосунку та серверної складової. На рис. 2.2 подано структурну схему, що демонструє їхню організацію та взаємодію між компонентами.



Рисунок 2.2 — Структурна схема програмного комплексу

Опис модулів програмного забезпечення та визначення їхніх функцій подано в таблицях 2.1-2.2.

Таблиця 2.1 — Характеристика модулів мобільного застосунку

Модуль	Опис
Модуль роботи з переліком приймання їжі	Модуль відповідає за відображення та сортування списку приймання їжі, видалення приймання їжі, відображення статистики калорійності приймань їжі.
Модуль додавання приймання їжі	Модуль забезпечує можливість додавання приймання їжі.
Модуль редагування приймання їжі	Модуль реалізує редагування інформації про приймання їжі.
Модуль розпізнавання	Модуль відповідає за створення зображення їжі за допомогою камери пристрою, його шифрування, надсилання на сервер, отримання результату розпізнавання, додавання приймання їжі на основі цього результату.
Модуль отримання даних про продукт	Модуль призначений для отримання даних про продукт з відкритої бази даних Open Food Facts за допомогою штрихкоду.
Модуль роботи з розумними вагами	Модуль призначений для взаємодії з розумними вагами Xiaomi Mi Body Composition Scale 2 для отримання значень ваги та опору тіла, розрахунку показників тіла за даними про користувача.
Модуль роботи з БД	Модуль забезпечує створення та відкриття БД, отримання даних з неї, додавання, оновлення та видалення записів про приймання їжі та користувача в БД.

Таблиця 2.2 — Характеристика модулів серверної частини

Модуль	Опис
Модуль розпізнавання	Модуль обробляє HTTP-запити від мобільного застосунку, що містять зображення їжі, виконує їх розшифрування, розпізнавання та надсилає результати назад у застосунок.
Модуль створення моделі для розпізнавання	Модуль призначений для створення та навчання моделей розпізнавання на навчальній вибірці, проведення тестування, виведення статистичних показників та збереження моделей.

2.3 Вибір мови програмування

Вибір мови програмування є одним із ключових етапів у процесі створення програмного забезпечення, оскільки він безпосередньо впливає на продуктивність, масштабованість та зручність подальшої підтримки системи.

Для розроблення мобільних застосунків найчастіше використовуються мови Java, Swift, Kotlin і C# [15], кожна з яких має власні особливості.

Під час розробки серверної частини також постає питання вибору оптимальної мови програмування. Серед найпоширеніших варіантів — Java, C++, C# та Python [15]. Залежно від вимог до продуктивності, швидкості обробки запитів та зручності інтеграції з клієнтською частиною, кожна з цих мов може бути ефективним інструментом для створення надійних і високопродуктивних серверних рішень.

2.3.1 Java

Мова програмування Java — це високорівнева, об'єктно-орієнтована мова, розроблена компанією Sun Microsystems (зараз — Oracle Corporation) у середині 1990-х років [16]. Вона спроектована таким чином, щоб бути незалежною від платформи: код компілюється у байт-код, який виконується в середовищі Java Virtual Machine (JVM), що дозволяє запускати одну і ту ж програму на різних операційних системах [16].

Серед парадигм Java підтримує класичне об'єктно-орієнтоване програмування із класами, об'єктами, наслідуванням, інтерфейсами [16]. У ній передбачена велика стандартна бібліотека, автоматичне збирання сміття (GC — garbage collection), строгий статичний контроль типів, широка екосистема інструментів (IDE, фреймворки) та корпоративна підтримка [16].

Переваги [16]:

- портативність: завдяки JVM програми на Java можуть запускатись на різних платформах без змін — це одна з ключових переваг Java;
- об'єктно-орієнтований підхід та багата стандартна бібліотека: це полегшує створення масштабованих систем із чіткою архітектурою;
- велика екосистема та добрий рівень підтримки: багато фреймворків, бібліотек, багатий набір інструментів (підтримка від Oracle, активна спільнота);

- статичне типізування і компіляція дають переваги в продуктивності та надійності (перевірка типів на етапі компіляції);

- безпека та управління пам'яттю: JVM із GC знижує ризики витоків пам'яті, контролює виконання середовища.

Недоліки [16]:

- продуктивність/час запуску: хоча Java досить ефективна, вона все ж може поступатись (у певних сценаріях) нативним мовам через накладні витрати JVM та байт-кодування;

- повільний старт (startup time): JVM-завантаження, компіляція та ініціалізація можуть створювати затримки на запуску програм;

- помірна складність: об'єктно-орієнтований підхід, строгі правила типізації та велика кількість бібліотек можуть бути бар'єром для швидкого старту початківця;

- відносно велика витрата пам'яті: JVM і бібліотеки Java часто споживають більше пам'яті, ніж легковагові мови чи нативні рішення;

- накладні витрати абстракцій: широкий функціонал, багато шарів абстракцій (наприклад, фреймворки) можуть ускладнювати оптимізацію, а також уповільнювати розробку у випадках простих задач.

2.3.2 Swift

Мова програмування Swift — це сучасна, компільована та багатопарадигмова мова, розроблена компанією Apple Inc. і представлена у 2014 році [17]. Спочатку Swift створювалася як заміна мові Objective-C для розробки під платформи iOS, macOS, tvOS та watchOS, але з часом набрала силу також у серверній та мультиплатформній розробці [17].

Swift підтримує об'єктно-орієнтований, функціональний та процедурний стилі програмування, має сучасний статичний типізатор, високу увагу до безпеки пам'яті та синтаксис, орієнтований на читабельність [17].

Переваги [17]:

- лаконічність синтаксису: Swift створена з метою бути більш простою і читабельною, ніж Objective-C, що дозволяє писати код ефективніше;

- безпека пам'яті та типів: мова має механізми, які знижують ризик помилок і підвищують надійність програм;

- висока продуктивність: Swift компілюється у нативний код і може забезпечувати хорошу швидкість виконання;

- мультипарадигмальність і сучасні функції;

- інтероперабельність з C/C++/Objective-C: можливість підключення бібліотек і кодової бази на C/C++;

- підходить і для навчання, і для комерційної розробки.

Недоліки [17]:

- обмежена екосистема поза Apple-платформами: хоча Swift розвивається як мультиплатформна мова, її первинна орієнтація на екосистему Apple може створювати бар'єри для використання у повністю кросплатформних проєктах;

- проблеми сумісності та нестабільність у ранніх версіях: швидкий розвиток іноді призводив до проблем із зворотною сумісністю;

- порівняно невелика кількість бібліотек/фреймворків (особливо поза iOS/macOS): оскільки екосистема виникла пізніше багатьох мов, доступність деяких фреймворків може бути меншою;

- не така потужна підтримка у корпоративному середовищі (для мультиплатформних систем) як у мов з довшою історією;

- наявність «вузької» спеціалізації: через історичний фокус на мобільні/Apple-платформи, Swift може не бути першим вибором для систем із великою базою вже наявного коду на інших мовах.

2.3.3 Kotlin

Мова програмування Kotlin — це сучасна, статично типізована мова програмування, що розроблена компанією JetBrains і вперше представлена в 2011 році, а з 2017 року отримала офіційний статус однієї з мов розробки для платформи Android Studio [18]. Kotlin працює на JVM, а також підтримує транс-компіляцію в JavaScript та на нативні платформи (Kotlin/Native), що робить її придатною для розробки не тільки мобільних Android-застосунків, але й мультиплатформних рішень [18].

Ключовими парадигмами мови є об'єктно-орієнтований підхід та функціональні можливості (наприклад, замикання, функції вищого порядку, типи даних із зразками), що дозволяє писати код більш виразно та з меншою кількістю шаблонів, ніж у більш традиційних JVM-мовах [18]. Серед іншого, Kotlin має повну сумісність з кодом на Java, що дозволяє легко інтегрувати її в існуючі проекти на Java або поступово переходити на Kotlin без повного переписування [18].

Переваги [18]:

- сумісність з Java: Kotlin може використовуватись у проєктах, які вже написані на Java, без необхідності повної міграції;
- більш лаконічний та виразний синтаксис;
- сучасні функції безпеки;
- зростаюче прийняття в Android-спільноті: багато Android-застосунків переходять на Kotlin;
- мультиплатформність: Kotlin/Native та Kotlin/JS дозволяють використовувати мову за межами JVM-середовища — наприклад, для розробки серверної логіки, фронтенду чи нативних застосунків.

Недоліки [18]:

- менша кількість корпоративно-зрілого досвіду ніж у Java;

- навчальна крива для функціональних конструкцій: використання функцій вищого порядку, розширених функціональних можливостей вимагає від розробника нових навичок;
- проекти із сильною залежністю від Java-інфраструктури можуть стикатись із міграційними труднощами;
- потенційна невизначеність щодо довгострокової підтримки деяких бібліотек: через те, що Kotlin відносно «молодша» за Java, певні бібліотеки чи фреймворки можуть бути менш розвинені або мати меншу спільноту;
- продуктивність JVM-середовища: хоч Kotlin компілюється у байт-код JVM і зазвичай має ті ж переваги, що й Java — накладні витрати JVM ті ж, що й у Java.

2.3.4 C++

Мова програмування C++ — це потужна і універсальна мова загального призначення, яка поєднує процедурні, об'єктно-орієнтовані та узагальнені (generic) парадигми програмування [19]. Вона була створена як надбудова над мовою C, з метою надати можливості управління пам'яттю, низькорівневого доступу до ресурсів, а також підтримки абстракцій високого рівня [19].

C++ дозволяє програмістові працювати дуже близько до «заліза» (асемблерного чи машинного рівня), але водночас дає інструменти для побудови складних структурованих систем [19].

Переваги [19]:

- висока продуктивність і ефективність: C++ компілюється у машинний код і дозволяє оптимізувати виконання програм, що особливо важливо в системах із вимогами до ресурсів;
- контроль над ресурсами та пам'яттю: через прямий доступ до пам'яті, маніпуляції покажчиками та можливість ручного управління

об'єктами, C++ дає програмісту великий контроль над тим, як використовуються ресурси;

- багатопарадигмовість: C++ підтримує процедурний, об'єктно-орієнтований і узагальнений підходи, що робить мову гнучкою для багато типів застосувань;

- широке застосування в системному/вбудованому/ігровому програмуванні: завдяки поєднанню продуктивності, контролю та можливостей, C++ часто використовується там, де важлива швидкість, ресурсна ефективність та безпосередній доступ до апаратури;

- масштабованість та портативність: C++ дозволяє створювати великі проєкти, які можуть працювати на різних апаратних платформах, за умови належної адаптації.

Недоліки [19]:

- складність мови та крута крива навчання: багато концептів (наприклад, шаблони, ручне управління пам'яттю, множинне спадкування) можуть бути складними для розуміння, особливо початківцям;

- вища ймовірність помилок (безпеки, пам'яті): пряма робота з пам'яттю і покажчиками підвищує ризик помилок: витоки пам'яті, порушення безпеки;

- великі витрати на підтримку/супровід: складний код, використання старих конструкцій чи шаблонів може ускладнити читабельність, розуміння та супровід програмної бази;

- неавтоматичне управління пам'яттю: на відміну від мов із збирачем сміття, у C++ програміст несе більшу відповідальність за життєвий цикл об'єктів, що ускладнює розробку в командах з менш досвідченими програмістами;

- менша продуктивність розробки: через складну синтаксисичну базу та потребу ручної роботи з низькорівневими аспектами, C++-розробка може бути повільнішою у порівнянні з мовами вищого рівня, коли швидкість розробки важливіша за максимальну продуктивність.

2.3.5 C#

Мова програмування C# — це сучасна, багатопарадигмова мова, розроблена компанією Microsoft та представлена у 2000 році як частина платформи .NET Framework [20]. Вона підтримує об'єктно-орієнтований, компонентно-орієнтований, функціональний та імперативний стилі програмування [20]. Мова проектувалася з метою об'єднати продуктивність, зручність розробки та безпеку типів, і призначена для створення корпоративних, десктопних, веб- і мобільних застосунків [20].

Серед основних особливостей C# — строгий статичний контроль типів, автоматичне управління пам'яттю через збирач сміття (garbage collection), розвинена стандартна бібліотека (бібліотеки .NET), підтримка подій, делегатів, LINQ (Language-Integrated Query) та асинхронного програмування через `async/await` [20].

Переваги [20]:

- продуктивність розробки: завдяки розвиненій екосистемі .NET і багатій стандартній бібліотеці, C# дозволяє швидше створювати застосунки з порівняно невеликими зусиллями;
- типова безпека та сучасний контроль: C# реалізує перевірку типів під час компіляції, підтримує автоматичний контроль пам'яті, що зменшує ризики помилок низького рівня;
- мультипарадигмальність і гнучкість: мова підтримує об'єктно-орієнтований підхід, узагальнення (generics), делегати, події, асинхронність, що дозволяє вирішувати широкий спектр задач;
- сильна корпоративна підтримка і екосистема: платформа .NET, середовище розробки (наприклад, Visual Studio), велика спільнота — усе це забезпечує добрий рівень підтримки для проєктів на C#;
- кросплатформність (у сучасних версіях): хоч C# історично була тісно пов'язана з Windows, з появою .NET Core/.NET 5+ з'явилась підтримка Linux, macOS та контейнерів — це розширює можливості застосування мови.

Недоліки [20]:

- продуктивність/накладні витрати: хоч С# має хорошу продуктивність для типових задач, вона може відставати від нативних мов (наприклад С++) у ситуаціях, коли критичні час виконання або оптимізація ресурсів;
- залежність від екосистеми .NET: використання С# часто передбачає .NET-середовище, що може бути обмеженням для деяких проєктів або середовищ з особливими вимогами;
- вища крива входу для певних парадигм: незважаючи на популярність, застосування таких конструкцій як асинхронність (async/await), делегати, події, LINQ може потребувати від розробника значно більше знань, ніж в більш простих мовах;
- менша універсальність у деяких доменах: для дуже низькорівневого або вбудованого програмування С# може бути менш підходящою ніж, скажімо, С++ або мови спеціалізованого призначення;
- розмір і складність проєктів: у великих проєктах, особливо з довгою історією або специфічними вимогами до архітектури, використання С# із масивною бібліотекою .NET може призводити до надмірності або ускладнень підтримки.

2.3.6 Python

Мова програмування Python — це високорівнева, інтерпретована, багатопарадигмова мова програмування загального призначення, створена Гвідо ван Россумом і вперше представлена в 1991 році [21]. Вона підтримує об'єктно-орієнтоване, процедурне та функціональне програмування, має динамічну типізацію, автоматичне управління пам'яттю і акцент на читабельність коду та швидкість розробки [21]. За останні роки Python здобув значну популярність у таких галузях, як наука про дані, машинне навчання, веброботка та автоматизація [21].

Переваги [21]:

- простота і читабельність: синтаксис Python спроектований так, щоб бути інтуїтивно зрозумілим; він дозволяє розробникам швидко писати та підтримувати код, що особливо корисно як для новачків;
- широка екосистема бібліотек і фреймворків: Python має розгалужену спільноту та безліч модулів для наукових обчислень (NumPy, Pandas), машинного навчання (TensorFlow, PyTorch), веброзробки (Django, Flask) та автоматизації;
- гнучкість і мультипарадигмовість: мова підтримує кілька підходів — об'єктно-орієнтований, процедурний, функціональний, що робить її придатною для різноманітних задач;
- швидкість розробки та прототипування: завдяки високому рівню абстракції та відсутності необхідності в ручному управлінні пам'яттю, розробка на Python часто відбувається швидше, ніж на мовах нижчого рівня;
- популярність і підтримка спільноти.

Недоліки [21]:

- продуктивність: через інтерпретований характер, динамічну типізацію та автоматичне управління пам'яттю, виконання програм на Python може бути значно повільніше, ніж на компільованих нативних мовах;
- витрати пам'яті і ресурсів: динамічні типи та високий рівень абстракції можуть призводити до більшого споживання пам'яті і менш оптимального використання ресурсів;
- обмеження багатопоточності: в стандартній реалізації Python існує глобальне блокування інтерпретатора, що обмежує ефективність багатопоточних обчислень;
- менша придатність для низькорівневого або вбудованого програмування;
- менше оптимізацій для певних видів задач.

2.4 Порівняння мов програмування

На основі розглянутих характеристик було виконано порівняльну оцінку мов програмування, що можуть бути використані для створення мобільних застосунків. Узагальнені результати аналізу подано в таблиці 2.3.

Таблиця 2.3 — Порівняння мов програмування [16]-[21]

Критерії порівняння	Java	Swift	Kotlin	C#
Зрозумілість синтаксису	—	+	+	+
Підвищена продуктивність	+	+	+	+
Підтримка багатопоточності	+	+	+	+
Підтримка IDE	+	+	+	+
Простота вивчення	—	+	+	+
Підтримка спільноти	+	+	+	+
Бібліотеки сторонніх розробників	+	+	+	+
Інструментарій та екосистема	+	+	+	+
Розробка для Android	+	—	+	+
Розробка для iOS	—	+	—	+
Кросплатформна розробка	—	—	+	+

За підсумками наведеного порівняння можна відзначити, що найбільш придатним рішенням для кросплатформної розробки є використання мови C# у поєднанні з фреймворком .NET MAUI. C# вирізняється широкими можливостями, розвиненою екосистемою та наявністю великої кількості бібліотек, що забезпечують підтримку розробки для різних операційних систем, включно з Android та iOS. Фреймворк .NET MAUI надає інструментарій для створення інтерфейсів користувача, які можуть працювати на кількох платформах, що дозволяє застосовувати спільну кодову базу та зменшити витрати часу на розробку. Такий підхід сприяє скороченню трудомісткості проєкту та забезпечує стабільну роботу застосунку на різних пристроях. Таким чином, для реалізації мобільного застосунку було обрано C# та технологію .NET MAUI.

Під час аналізу можливих мов програмування для реалізації серверної частини було сформовано порівняльну характеристику, яка подана в таблиці 2.4.

Таблиця 2.4 — Порівняння мов програмування [16]-[21]

Критерії порівняння	Java	C++	C#	Python
Зрозумілість синтаксису	—	—	+	+
Підвищена продуктивність	+	+	+	±
Підтримка багатопоточності	+	+	+	+
Підтримка IDE	+	+	+	+
Простота вивчення	—	—	+	+
Підтримка спільноти	+	+	+	+
Бібліотеки сторонніх розробників	+	+	+	+
Інструментарій та екосистема	+	+	+	+
Бібліотеки машинного навчання	—	+	+	+
Вебфреймворки	+	—	+	+
Кросплатформна розробка	+	+	+	+

Отримані результати свідчать, що найбільш функціональними у контексті серверної розробки є C# та Python. Водночас Python демонструє переваги саме у сфері веброботи та задач, пов'язаних із обробкою даних і розпізнаванням. Завдяки використанню Flask забезпечується зручний і гнучкий механізм побудови серверної логіки, а бібліотеки TensorFlow, Keras і PyCryptodome надають можливість реалізувати алгоритми машинного навчання та криптографічні механізми. Таким чином, для створення серверної частини програми було обрано мову Python у поєднанні з Flask, Keras, TensorFlow та PyCryptodome, що дозволяє ефективно реалізувати необхідний функціонал і забезпечити надійну роботу системи.

2.5 Вибір середовища розробки

Під час створення мобільного застосунку та серверної частини важливим етапом є вибір відповідного середовища розробки. Середовище визначає зручність роботи розробника, доступні інструменти та підтримувані

технології. Нині існує низка поширених рішень, що забезпечують широкий спектр можливостей для проєктування та побудови програм. До таких середовищ належать Xcode, Android Studio, Eclipse та Visual Studio [15].

2.5.1 Xcode

Інтегроване середовище розробки (IDE) Xcode, розроблене компанією Apple Inc., призначене для створення нативних застосунків під платформи iOS, iPadOS, macOS, watchOS та tvOS [22]. Xcode має редактор коду, інтегрований Interface Builder для створення UI на iOS/macOS, засоби для налагодження, профілювання, виконання, симулятори пристроїв, систему керування версіями і компілятори для Swift, Objective-C, C, C++ тощо [22].

Оскільки Xcode призначений саме для екосистеми Apple, він забезпечує глибоку інтеграцію з системними SDK, UI-фреймворками (UIKit/SwiftUI), App Store та іншими службами Apple [22].

Переваги [22]:

- максимальна інтеграція з платформою Apple: завдяки тому, що Xcode є офіційним середовищем Apple, він забезпечує повний доступ до всіх SDK, API і можливостей платформи;
- оптимізована продуктивність нативних застосунків;
- потужні засоби UI-розробки й налагодження: Interface Builder, симулятори різних пристроїв, профайлери дають можливість легкого створення і тестування інтерфейсів і поведінки застосунків;
- гарна підтримка сучасних мов Apple (Swift, Objective-C);
- керування життєвим циклом застосунку: інтеграція з App Store, підписування застосунків, тестування на реальних пристроях.

Недоліки [22]:

- обмежена кросплатформність: оскільки Xcode орієнтований на екосистему Apple, створення застосунків під Android або інші ОС потребує використання інших інструментів;

- залежність від апаратної платформи (macOS): для використання Xcode потрібна система macOS, що створює додатковий ресурсний бар'єр;
- швидкість змін і оновлень: часті оновлення платформ Apple можуть вимагати швидкої адаптації середовища, оновлення бібліотек, іноді виникають помилки сумісності з попередніми версіями — що створює додаткові витрати на супровід;
- велика абстракція нативного коду: для розробників, які не знайомі з мовами Swift/Objective-C або специфікою Apple платформи, поріг входу може бути вище через необхідність освоєння екосистеми і специфічних SDK.

2.5.2 Android Studio

Android Studio — це офіційне IDE від Google, призначене для створення Android-застосунків [23]. Воно побудоване на базі IntelliJ IDEA і включає інструменти для написання коду (підтримує Java, Kotlin, C++), інтеграцію з Android SDK, емулятори різних пристроїв, редактор макетів (UI), систему побудови Gradle, відладчики та профайлери [23]. Android Studio створено з метою оптимізувати весь життєвий цикл розробки Android-застосунків — від дизайну інтерфейсу до тестування і розгортання [23].

Переваги [23]:

- глибока інтеграція з платформою Android: Android Studio включає Android SDK, емулятори, інструменти профілювання й оптимізації, що дозволяє розробникам працювати з усіма аспектами Android-застосунку;
- підтримка сучасних мов програмування: офіційна підтримка Kotlin і Java, а також можливість використовувати C++ дають гнучкість у виборі технологій;
- потужна система складання і автоматизації: Gradle-базована система збірки дозволяє конфігурувати різні варіанти збірок, залежності, ресурси;

- редактор UI з візуальним попереднім переглядом: Layout Editor підтримує drag-and-drop, перегляд на різних розмірах екранів і емуляторах, що спрощує створення адаптивних інтерфейсів;

- безкоштовність і широке поширення: Android Studio розповсюджується під вільною ліцензією, має велику спільноту й численні плагіни/доповнення.

Недоліки [23]:

- високі системні вимоги: Android Studio відома тим, що споживає значну кількість ресурсів — пам'яті, процесора, час на запуск емулятора;

- повільна збірка/довгі цикли: через використання Gradle, великої кількості ресурсів та емулятора, час побудови проєкту і запуску може бути тривалим, особливо в великих проєктах;

- крива навчання та складність налаштування: для початківців середовище може бути надто складним через велику кількість опцій, складну архітектуру проєктів і необхідність розуміти Android-специфіку;

- повільний старт середовища/емюлятора.

2.5.3 Eclipse

Eclipse IDE — це безкоштовне інтегроване середовище розробки з відкритим кодом, яке спочатку було орієнтовано на розробку на мові Java, але з часом розширило свою функціональність і тепер підтримує багато мов програмування (C, C++, C#, JavaScript, Python тощо) [24]. Його архітектура базується на платформі Rich Client Platform (RCP), що дозволяє створювати плагіни й розширення — завдяки цьому Eclipse стає платформою для побудови спеціалізованих середовищ розробки [24].

Переваги [24]:

- відкрите джерело і безкоштовність;

- модульність і розширюваність: через плагіни та архітектуру RCP Eclipse легко адаптувати під різні мови і задачі, створювати спеціалізовані плагіни;
- широка підтримка мов та платформ: Eclipse підтримує багато мов програмування і працює на різних ОС;
- добре усталений вибір в корпоративному середовищі: завдяки стабільності проєкту та великій спільноті Eclipse часто використовується в організаціях як стандартне середовище.

Недоліки [24]:

- складність для новачків: велика кількість плагінів, параметрів конфігурації та розширень може створювати значний поріг входу для початківців;
- нижча продуктивність на великих проєктах;
- менша інтегрована підтримка у порівнянні з комерційними IDE: деякі функції налагодження, автозаповнення або UX-інструменти можуть бути менш потужними чи зручними, ніж у комерційних середовищах;
- залежність від плагінів: хоч модульність є перевагою, вона також означає, що багато функцій базового середовища залежать від плагінів — що може призводити до несумісностей або додаткового часу на налаштування;
- для мобільної розробки чи сучасних UI-фреймворків середовище Eclipse може бути менш зручним, ніж спеціалізовані IDE.

2.5.4 Visual Studio

Microsoft Visual Studio — це потужне IDE, створене компанією Microsoft, яке підтримує широкий спектр мов програмування (C#, C++, VB.NET, F#, JavaScript та інші) і дозволяє створювати застосунки для різних платформ: настільні (Windows, macOS), веб-, мобільні застосунки (через Xamarin/.NET MAUI), ігри і сервіси [24].

Visual Studio має редактор коду з підтримкою автодоповнення, системою налагодження, засобами тестування, профілювання, інтеграцію з системами контролю версій (Git, Azure DevOps), велику екосистему розширень та компонентів [24]. Завдяки цьому IDE добре підходить для корпоративної розробки та великих командних проєктів.

Переваги [24]:

- висока продуктивність розробки: Visual Studio надає багатий набір інструментів (редактор з автодоповненням, налагодження, тестування), що дозволяє програмістам працювати швидше;
- широка підтримка мов і платформ;
- добре розвинена екосистема і розширення: завдяки великій спільноті та маркету розширень, можна легко додавати плагіни і інструменти, що підходять під конкретні задачі;
- інтеграція з корпоративними сервісами: Visual Studio має глибоку інтеграцію з Azure DevOps, системами управління версіями, що робить його привабливим для великих підприємств;
- стабільність і надійність: Visual Studio часто оновлюється та має підтримку, що підвищує довіру до нього в комерційних проєктах.

Недоліки [24]:

- високі системні вимоги і «важкість» середовища;
- ліцензійні/комерційні витрати: хоч існує безкоштовна версія (Community Edition), повна версія зі всіма корпоративними можливостями може бути дорогою;
- складність налаштування і велика кількість функцій: для новачка середовище може бути інтенсивним — велика кількість опцій, налаштувань, плагінів;
- фокус на екосистему Microsoft: хоч Visual Studio підтримує кросплатформну розробку, найповніша інтеграція часто досягається у рішеннях, пов'язаних з Microsoft-технологіями.

2.6 Порівняння середовищ розробки

На основі зібраних даних було здійснено порівняльний огляд доступних середовищ розробки. Узагальнені результати оцінювання подано в таблиці 2.5.

Таблиця 2.5 — Порівняння середовищ розробки [22]-[24]

Критерії порівняння	Xcode	Android Studio	Eclipse	Visual Studio
Кросплатформність	—	—	+	+
Підсвічування синтаксису	+	+	+	+
Автозавершення коду	+	+	+	+
Швидкий та зручний пошук	+	+	+	+
Перевірка та аналіз коду	+	+	+	+
Швидкий та безпечний рефакторинг коду	+	+	+	+
Простота встановлення	+	+	—	+
Інтуїтивний інтерфейс	+	+	—	+
Підтримка пакетів	+	+	+	+
Підтримка репозиторіїв	+	+	+	+
Підтримка C#	—	—	+	+
Підтримка Python	—	—	+	+

Проведений аналіз показав, що Visual Studio найбільш повно відповідає вимогам, що висуваються до середовища розробки в межах даного проєкту. Воно забезпечує комфортну роботу як із C#, так і з Python, що стає можливим завдяки наявності розширень на кшталт Python Tools for Visual Studio. Це дозволяє виконувати написання, налагодження та тестування коду на обох мовах в єдиному середовищі. Visual Studio також вирізняється потужними засобами для рефакторингу, аналізу та перевірки якості коду, що позитивно впливає на підтримуваність та надійність створюваного програмного забезпечення. Додатковою перевагою є вбудований Android Emulator, який дає змогу проводити тестування мобільних застосунків під Android безпосередньо із середовища розробки. Таким чином, для реалізації програмного забезпечення було обрано Visual Studio.

2.7 Вибір системи управління базами даних

Під час створення мобільних застосунків важливо обрати відповідну систему керування базами даних, яка гарантуватиме стабільне, продуктивне та безпечне зберігання інформації. У сфері мобільної розробки найчастіше застосовуються такі СУБД, як PostgreSQL, MySQL, MongoDB та SQLite [25]-[27].

2.7.1 PostgreSQL

PostgreSQL — це потужна об'єктно-реляційна СУБД з відкритим вихідним кодом, відома своєю надійністю, розширюваністю й підтримкою складних типів даних [25]. Вона підтримує ACID-транзакції, механізм керування паралельним доступом, а також широкий набір індексів, що дозволяє ефективно працювати зі складними запитами й великими обсягами даних [25]. Крім того, PostgreSQL має підтримку JSON/JSONB, геопросторових даних, і може бути розгорнута в хмарних середовищах або використовуватися для аналітики [25].

Переваги [25]:

- висока продуктивність SELECT-запитів;
- стабільність під навантаженням: PostgreSQL демонструє стабільність при складних сценаріях вставки та читання даних одночасно;
- гнучкість моделювання даних: PostgreSQL добре адаптується до різних моделей даних — підтримує як реляційні структури, так і JSON-формати, забезпечуючи баланс між структурованістю й гнучкістю;
- надійність і консистентність: завдяки підтримці ACID і прозорому контролю транзакцій PostgreSQL підходить для систем, де важлива цілісність даних;

- розвинуті індекси та оптимізація: PostgreSQL надає багато варіантів індексування та оптимізації запитів, що покращує продуктивність навіть для великих або складних баз даних.

Недоліки [25]:

- складність конфігурації та адміністрування: через різноманітний функціонал PostgreSQL може вимагати значних зусиль для тонкого налаштування, особливо в великих або нестандартних сценаріях;

- вища затримка при вставці в деяких сценаріях: конкурентні операції можуть створювати накладні;

- модель не підходить для всіх типів даних: у випадках, коли дані дуже нерегулярні або гнучкі, NoSQL-бази можуть забезпечувати більш природну модель зберігання;

- ресурсоемність: при інтенсивному використанні транзакцій та паралельної обробки даних PostgreSQL може використовувати значні ресурси, особливо якщо не оптимізовано конфігурацію.

2.7.2 MySQL

MySQL — це широко розповсюджена реляційна система управління базами даних з відкритим вихідним кодом, яка підтримує стандарт SQL [25]. Вона часто використовується в вебзастосунках, системах управління контентом, а також в інших системах, де важлива швидка обробка запитів при значному обсязі транзакцій [25]. MySQL підтримує реплікацію, індекси, кешування запитів і має добре налагоджену екосистему, що робить її популярною для легких та середніх навантажень [25].

Переваги [25]:

- швидкість операцій вставки;

- добре оптимізована для простих запитів: MySQL часто добре працює з простими SELECT-запитами;

- широка підтримка та екосистема: MySQL — дуже поширена СУБД, має багатьох користувачів, документацію, інструменти адміністрування та бібліотеки для різних мов програмування;

- легкість налаштування та використання: для базових сценаріїв MySQL досить просто встановити й налаштувати, що робить її популярною для стартапів і вебпроектів;

- відкрита версія та комерційні варіанти: MySQL доступна як безкоштовна версія, а також існують комерційні редакції з додатковими можливостями та підтримкою.

Недоліки [25]:

- низька продуктивність читання під навантаженням із умовами;

- падіння продуктивності під одночасними запитами: у складних сценаріях, де одночасно виконуються операції вставки та читання, продуктивність MySQL може сильно погіршуватись;

- менша стабільність при дуже високій конкуренції транзакцій: через механіки блокувань або інших обмежень MySQL може стати менш ефективною для сильно навантажених транзакційних систем;

- обмежені можливості складного моделювання даних: MySQL може бути менш зручною для роботи зі складними схемами або нетрадиційно структурованими даними;

- потреба в тонкому налаштуванні: щоб досягти високої продуктивності на серйозних проєктах, часто треба налаштовувати параметри індексації, кешу, конфігурації запитів — це вимагає досвіду.

2.7.3 MongoDB

MongoDB — це документоорієнтована (NoSQL) система управління базами даних, яка зберігає дані у форматі JSON-подібних документів [26]. Завдяки гнучкій схемі даних, вона дозволяє легко зберігати структуровану, напівструктуровану або нерегулярну інформацію без необхідності жорсткої

схеми таблиць [26]. Також MongoDB підтримує горизонтальне масштабування, реплікацію, транзакції (в мультидокументному режимі), а її модель добре підходить для систем, де дані змінюються часто або мають складну, глибоко вкладену структуру [26].

Переваги [26]:

- гнучка модель даних: завдяки документоорієнтованій структурі MongoDB легко працювати з динамічною схемою;
- горизонтальне масштабування;
- гнучкість запитів: MongoDB підтримує ad-hoc-запити, індекси на різні поля документів, а також агрегування, що дає можливість створювати потужні запити без попередньої жорсткої схеми.

Недоліки [26]:

- обмежена транзакційна консистентність: хоча MongoDB підтримує транзакції, у розподілених кластерах може зростати складність підтримки повної ACID-консистентності;
- менш ефективна для складних реляційних запитів: MongoDB може поступатися реляційним СУБД у запитах з багатьма зв'язками між даними;
- навантаження на диск: документоорієнтована модель іноді призводить до дублювання даних (наприклад, якщо великі піддокументи повторюються), що може збільшити використання дискового простору;
- схема неявна — ризики неконсистентності: гнучка схема — це перевага, але вона також може призвести до проблем зі структурою даних, якщо не ввести валідацію або строгі правила схеми, особливо в командних проєктах;
- складність при великій конкуренції або складних транзакціях: у сценаріях з дуже високим навантаженням або дуже великими транзакціями MongoDB може вимагати додаткових налаштувань, щоб забезпечити продуктивність і консистентність.

2.7.4 SQLite

SQLite — це вбудована реляційна система управління базами даних з відкритим кодом, яка працює без окремого серверного процесу: база даних зберігається як один файл, а виклики до неї здійснюються через бібліотеку, що додається в застосунок [27]. Вона використовується на мобільних пристроях, в браузерях, телевізорах, автомобілях тощо [27]. SQLite спроектована з акцентом на мінімальні залежності, портативність, надійність і легкість вбудовування у програми [27].

Переваги [27]:

- вбудованість і портативність: SQLite легко інтегрується в різні платформи (мобільні, десктопні, вбудовані);
- висока надійність: SQLite має велику тестову базу, високу якість коду і перевірену роботу в реальних пристроях і критичних системах;
- мінімальні витрати на обслуговування: не потрібно налаштовувати сервер, менеджер пам'яті, адміністрування;
- формат даних кросплатформний.

Недоліки [27]:

- обмежена паралельність і масштабування;
- підходить для легких і середніх навантажень, але може бути недоцільною для розподілених або великовагових серверних систем з високим навантаженням;
- функціональні обмеження: як вбудована СУБД, SQLite має менше можливостей для масштабування, великої кластерності чи серверної взаємодії, ніж повноцінні серверні СУБД;
- файлова залежність: оскільки база даних — файл, робота на файлових системах із слабкою підтримкою блокувань може створювати накладні затрати чи ризики.

2.8 Порівняння систем управління базами даних

На основі поданих вище відомостей було здійснено порівняльний аналіз систем управління базами даних, результати якого узагальнено в таблиці 2.6.

Таблиця 2.6 — Порівняння СУБД [25]-[27]

Критерії порівняння	PostgreSQL	MySQL	MongoDB	SQLite
Швидкодія	+	+	+	+
Функціональність	+	+	+	+
Надійність	+	+	—	+
Масштабованість	+	+	+	—
Підтримка SQL	+	+	—	+
Зручність використання	+	+	—	+
Відкритість та спільнота	+	+	+	+
Сумісність з мобільними платформами	+	+	+	+
Вбудованість	—	—	—	+

Аналіз даних таблиці 2.6 показує, що SQLite є найкращим вибором для локального зберігання даних у мобільному застосунку. Основна перевага цієї СУБД полягає у її вбудованості: вона працює без серверної інфраструктури, є простою в інтеграції та не потребує додаткових налаштувань. Завдяки мінімальним вимогам до апаратних ресурсів SQLite оптимально підходить для мобільних пристроїв. Крім того, підтримка транзакцій, індексів та механізмів кешування забезпечує високу продуктивність і надійну роботу з даними. Таким чином, для реалізації мобільного застосунку було обрано SQLite.

2.9 Висновки за розділом 2

У межах цього розділу здійснено аналіз вимог до програмного забезпечення, обґрунтовано вибір засобів розробки та побудовано діаграму прецедентів. Також визначено загальну структуру системи та

охарактеризовано основні її компоненти. На підставі порівняння мов програмування встановлено, що для створення мобільного застосунку доцільно використовувати C# у поєднанні з фреймворком .NET MAUI, тоді як серверна частина реалізовуватиметься мовою Python із застосуванням Flask, Keras, TensorFlow та PyCryptodome.

Під час аналізу середовищ розробки з'ясовано, що найбільш придатним є Visual Studio, оскільки воно забезпечує зручні інструменти для роботи як з C#, так і з Python, а також має розширений функціонал для налагодження та тестування.

Крім того, проведене порівняння СУБД показало, що оптимальним рішенням для локального зберігання даних у мобільному застосунку є SQLite, оскільки ця система є вбудованою, характеризується мінімальними ресурсними потребами та добре пристосована для роботи на мобільних пристроях.

3 ОСНОВНІ РІШЕННЯ З РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Вибір шаблону проєктування

Перш ніж перейти до безпосередньої розробки програмного забезпечення, було визначено, який шаблон проєктування буде застосовано.

Шаблон проєктування — це перевірений підхід або узагальнений спосіб вирішення типових інженерних задач проєктування програмних систем [28]. Він описує набір ролей, обов'язків і взаємодій між компонентами, що дозволяє повторно використовувати архітектурні рішення, підвищувати зрозумілість коду та полегшувати його супровід. Шаблони спрямовані на те, щоб формалізувати досвід розробників у вигляді інструментів проєктування, які можна адаптувати під конкретні задачі.

MVVM (Model-View-ViewModel) — це архітектурний шаблон для побудови інтерфейсів користувача, який розділяє застосунок на три основні складові: модель (Model), представлення (View) та модель представлення (ViewModel) [28]. Модель відповідає за представлення даних і бізнес-логіку, представлення — за відображення інтерфейсу й обробку взаємодії користувача, а модель представлення є посередником, який інкапсулює стан і поведінку представлення, надаючи прив'язувані властивості і команди [28]. Ключовою особливістю MVVM є дво- або односпрямована прив'язка даних між View і ViewModel, що дозволяє автоматично відображати зміни стану без прямого маніпулювання елементами інтерфейсу [28].

Переваги застосування MVVM: чітка роздільність обов'язків, можливість повторного використання логіки представлення, полегшена автоматизація unit-тестів для ViewModel, а також сприяння паралельній роботі розробників і дизайнерів інтерфейсу. Водночас до недоліків відносять підвищену складність для простих інтерфейсів, потенційну потребу в додатковому шаблонному коді та необхідність правильної організації прив'язок. MVVM широко використовується в застосунках зі складним UI,

де переваги розділення логіки і відображення перекривають початкові витрати на архітектуру.

Для побудови архітектури мобільного застосунку було обрано шаблон проєктування Model-View-ViewModel. Загальну структуру архітектури подано на рисунку 3.1.

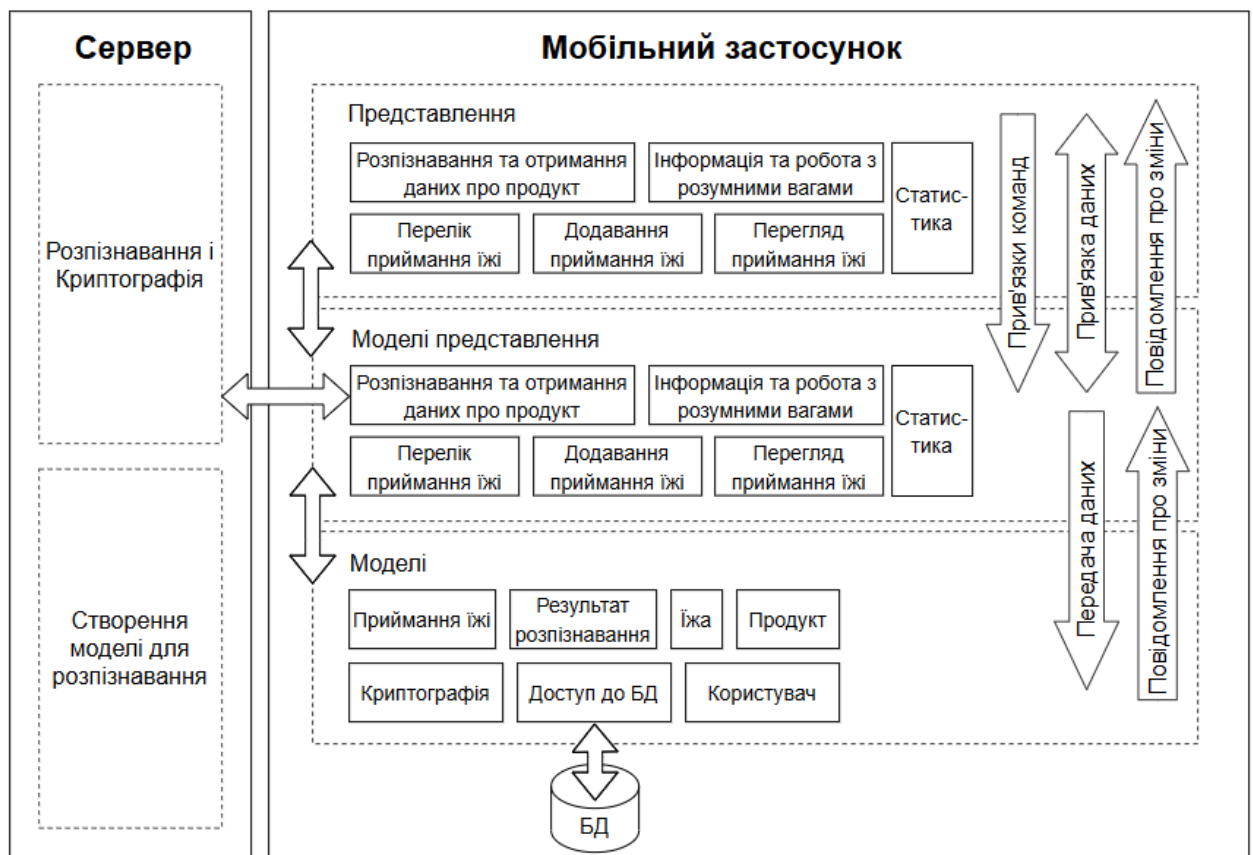


Рисунок 3.1 — Архітектура системи

Під час формування архітектурного рішення враховано, що програмна система буде розвиватися в процесі свого життєвого циклу, а повний обсяг вимог неможливо визначити на початковому етапі. У процесі реалізації та тестування можуть з'являтися нові деталі, які потребуватимуть коригування структури. Тому архітектура повинна бути гнучкою, здатною до адаптації та подальшого розширення, щоб забезпечити відповідність вимогам, що уточнюються в ході розробки.

3.2 Опис модулів розроблюваної системи

Архітектура мобільного застосунку складається з трьох основних компонентів: моделі, представлення та моделі представлення. Опис модулів, що входять до складу моделей, наведено в таблиці 3.1.

Таблиця 3.1 — Опис модулів частини з моделями

Модуль	Структура	Функції
Доступ до БД	class DBConstants, class CommonData, class MealsDatabase	Зчитування інформації з БД та файлу JSON; Запис та оновлення даних у БД; Видалення даних з БД.
Приймання їжі	class Meal	Зберігання даних про приймання їжі; Редагування даних.
Їжа	class MealFood	Зберігання даних про їжу; Редагування даних.
Продукт	class ProductRoot, class Product, class Nutriments	Зберігання даних про продукт.
Користувач	class UserInfo	Зберігання даних про користувача; Редагування даних.
Результат розпізнавання	class Prediction	Зберігання даних про результат розпізнавання.
Криптографія	class Cryptography	Шифрування зображень.

Відповідна діаграма класів подана на рисунку 3.2.

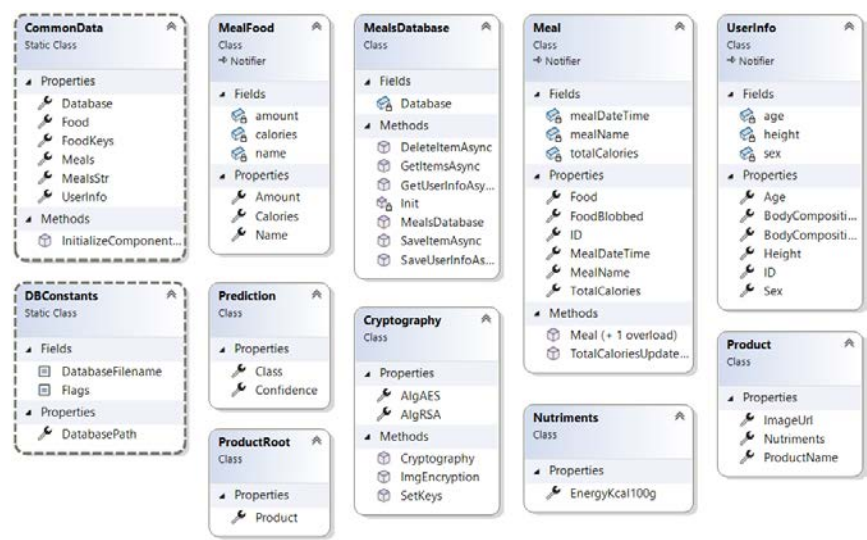


Рисунок 3.2 — Діаграма класів моделей

Модулі, що належать до моделей представлення, описані в таблиці 3.2.

Таблиця 3.2 — Опис модулів частини з моделями представлення

Модуль	Структура	Функції
Перелік приймання їжі	class AllMeals-VM	Отримання переліку приймання їжі з БД; Застосування сортування за обраними параметрами; Видалення обраного запису про приймання їжі з БД.
Статистика	class ChartsVM	Отримання дати для відображення даних; Визначення даних калорійності приймань їжі за обраний день та сумарної калорійності за днями відповідного тижня.
Додавання приймання їжі	class MealVM	Приймання даних від користувача; Створення нового запису в БД.
Перегляд приймання їжі		Отримання обраного приймання їжі; Оновлення інформації в БД після редагування.
Розпізнавання та отримання даних про продукт	class Scanner-VM	Отримання ключів шифрування від сервера; Отримання зображення їжі за допомогою камери, його шифрування та відправка на сервер; Отримання результату розпізнавання з сервера; Отримання даних про продукт з відкритої бази даних Open Food Facts за допомогою штрихкоду.
Інформація та робота з розумними вагами	class InfoVM	Отримання інформації про користувача; Взаємодія з розумними вагами Xiaomi Mi Body Composition Scale 2 для отримання значень ваги та опору тіла, розрахунку показників тіла за даними про користувача; Збереження інформації в БД.

Структура класів показана на рисунку 3.3.

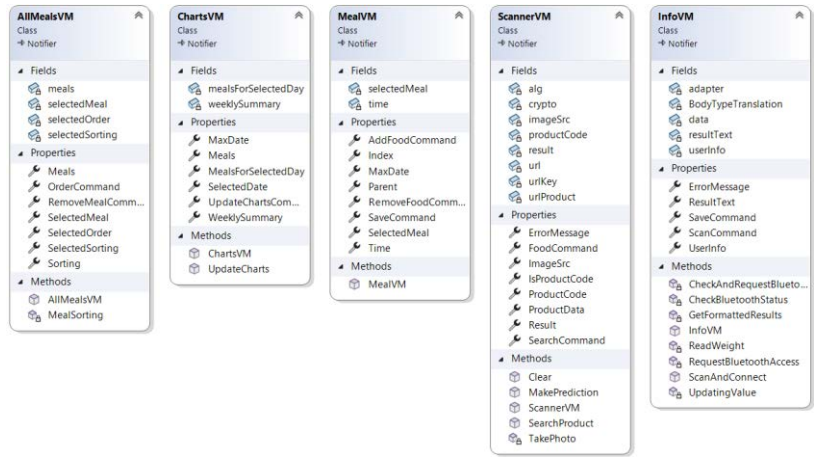


Рисунок 3.3 — Діаграма класів моделей представлень

Модулі, що належать до представлень, описані в таблиці 3.3.

Таблиця 3.3 — Опис модулів частини з представленнями

Модуль	Структура	Функції
Перелік приймання їжі	class AllMealsV	Відображення списку приймання їжі з урахуванням сортування; Встановлення параметрів сортування; Виділення конкретного приймання їжі та можливість його видалення або редагування; Перехід до додавання нового приймання їжі.
Статистика	class ChartsV	Введення дати для відображення (оновлення) даних; Відображення статистики калорійності приймань їжі за обраний день та сумарної калорійності за днями відповідного тижня.
Додавання приймання їжі	class MealV	Введення даних про приймання їжі; Збереження нової інформації (додавання до списку приймання їжі).
Перегляд приймання їжі		Відображення деталей обраного приймання їжі; Редагування даних про приймання їжі; Збереження змінених даних у базі даних.
Розпізнавання та отримання даних про продукт	class ScannerV	Використання камери для створення зображення їжі; Введення штрихкоду продукту для пошуку; Відображення зображення та результату розпізнавання або пошуку; Перехід до додавання приймання їжі з використанням результатів розпізнавання або пошуку.
Інформація та робота з розумними вагами	class InfoV	Відображення інформації про користувача, застосунок та розробника; Підключення до розумних ваг через Bluetooth; Збереження інформації про користувача.

Структура класів показана на рисунку 3.4.

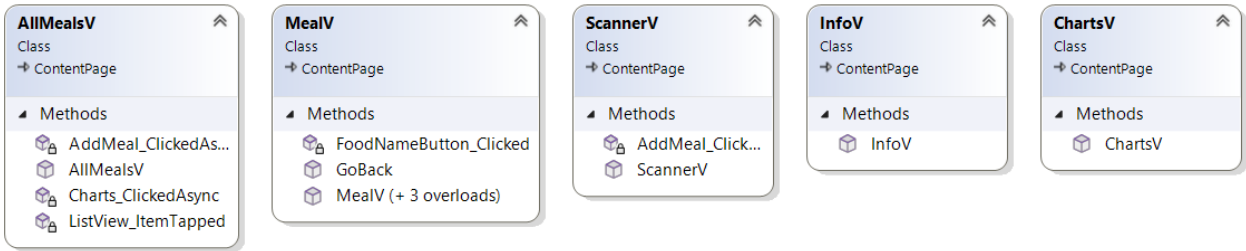


Рисунок 3.4 — Діаграма класів представлень

Серверна частина системи містить три ключові модулі: криптографічний, модуль розпізнавання та модуль створення моделі для розпізнавання. Опис зазначених компонентів наведено в таблиці 3.4.

Таблиця 3.4 — Опис модулів серверної частини

Модуль	Структура	Функції
Створення моделі для розпізнавання	MLModel.py pretrained.py evaluation.py visualization.py	Попередня обробка зображень їжі; Формування структури моделей для розпізнавання; Навчання моделей на навчальній вибірці; Тестування моделей на тестовій вибірці; Формування та виведення статистичних результатів; Збереження навченої моделі для використання.
Розпізнавання	MLWebServer.py	Надсилання ключів шифрування на застосунок; Отримання HTTP-запиту від застосунку, що містить зображення їжі; Розпізнавання їжі створеною моделлю; Передача результатів розпізнавання у застосунок.
Криптографія	Cryptography.py	Розшифрування отриманих зображень для обробки.

3.3 Створення модифікованого методу для розпізнавання їжі

Для впровадження функції розпізнавання зображень їжі було розроблено модифіковану згорткову нейронну мережу MealNet, а також створено 5 додаткових моделей на основі відомих архітектур: InceptionV3, Xception, ResNet50V2, MobileNetV2 і DenseNet169. Результати їхнього навчання використано для порівняльного аналізу, оцінювання точності та подальшого вдосконалення моделей.

3.3.1 Модифікована модель згорткової нейронної мережі

Згорткова нейронна мережа є різновидом глибинних нейронних архітектур для автоматичного виділення характерних ознак із вхідних даних [29]. Основною перевагою таких мереж є здатність самотійно навчатися структурам та визначати ключові особливості, необхідні для класифікації.

Операція згортки дає змогу поступово виокремлювати найважливіші шаблони: від простих елементів, таких як краї чи текстурні фрагменти, до складних об'єктів на пізніших етапах навчання [29]. Багатошаровість архітектури (рис. 3.5) забезпечує формування узагальнених, високорівневих представлень даних, завдяки чому мережа здатна інтерпретувати зображення комплексно. Такий підхід дозволяє моделі розпізнавати об'єкти з високою точністю, подібно до того, як це робить людина [29].

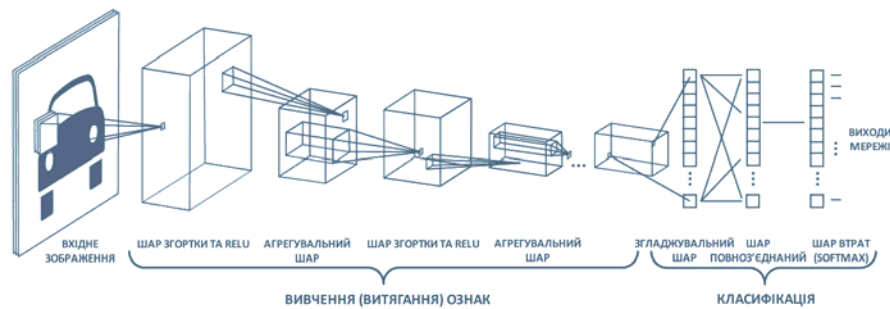


Рисунок 3.5 — Згорткова нейронна мережа [29]

Процес роботи нейрона можна формально описати за допомогою математичних виразів (3.1-3.2) [30]:

$$y = f\left(w_0 + \sum_{i=1}^k x_i w_i\right), \quad (3.1)$$

$$w_i \sim U\left[-\frac{\sqrt{6}}{\sqrt{n_{in} + n_{out}}}, \frac{\sqrt{6}}{\sqrt{n_{in} + n_{out}}}\right], \quad (3.2)$$

де y – значення вихідного сигналу;

f – функція активації;

w_0 – параметр зсуву;

x_i – i -тий елемент вектора входів;

w_i – i -те значення ваги входу (ініціалізатор Xavier/Glorot);

U – рівномірний розподіл;

n_{in} та n_{out} – кількість вхідних та вихідних одиниць у ваговому тензорі.

Ініціалізатори ваг — це процедура визначення початкових значень елементів вагової матриці для шару нейронної мережі [30]. У 2010 році Ксав'є Глорот та Йошуа Бенджіо запропонували метод, який враховує кількість нейронів на вході та виході шару для рівномірного розподілу початкових значень [30]. Такий підхід, відомий як ініціалізація Глорота, продемонстрував ефективність у більшості архітектур глибокого навчання й залишається одним із найпоширеніших, тому був застосований у моделі.

Оптимізатор Adam (Adaptive Moment Estimation) — це адаптивний стохастичний алгоритм оптимізації, який поєднує в собі переваги методу імпульсу (momentum) та RMSProp [31]. Алгоритм зберігає експоненціально згладжені оцінки першого моменту (m_t) та другого моменту (v_t) градієнтів у кожному кроці, а потім коригує їх для усунення зміщення (bias correction), щоб стабілізувати оновлення ваг [31].

На кроці t обчислюються градієнт та моменти (3.3-3.5) [31]:

$$g_t = \nabla_{\theta} L(\theta_{t-1}), \quad (3.3)$$

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t, \quad (3.4)$$

$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2, \quad (3.5)$$

де θ — параметри моделі;

L — функція втрат;

β_1 і β_2 — гіперпараметри згасання, які близькі до 1 (наприклад, 0,9 та 0,999).

Потім ці значення коригуються з урахуванням того, що на початкових ітераціях m_t і v_t можуть бути зміщеними до нуля (3.6) [31]:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}. \quad (3.6)$$

Останній крок — оновлення параметрів θ (3.7) [31]:

$$\theta_t = \theta_{t-1} - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t + \varepsilon}}, \quad (3.7)$$

де α – крок навчання (learning rate);

ε – невелика константа для чисельної стабільності.

Завдяки такій схемі Adam підлаштовує швидкість навчання окремо для кожного параметра, базуючись на історії градієнтів, що робить його дуже ефективним для тренування мереж, де градієнти можуть змінюватися в широкому діапазоні [31]. Тому цей оптимізатор разом з функцією втрат Sparse Categorical Crossentropy було використано для моделей.

В модифікованій моделі згорткової нейронної мережі MealNet вперше запропоновано структуру, яку представлено на рис. 3.6. Вона містить згорткові шари, шари максимізаційного агрегування, пакетної нормалізації, згладжування, виключення та повноз'єднані шари. Для підвищення варіативності даних на зображення додатково застосовуються перетворення: нормалізація, випадкове обертання, масштабування та горизонтальне віддзеркалення (це стосується всіх моделей). Як основну функцію активації використано ReLU (Rectified Linear Unit), тоді як у вихідному шарі — Softmax. Кількість класів (n) дорівнює 10, 30, 60 та 101, тобто для всіх моделей існує 4 варіанти моделі для різної кількості класів. В повноз'єднаних шарах додатково застосовується L2-регуляризація, яка сприяє зменшенню та більш рівномірному розподілу ваг за допомогою штрафного коефіцієнта.

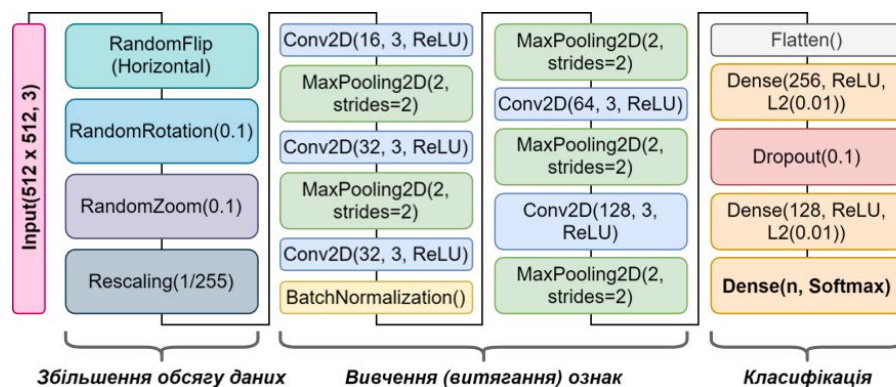


Рисунок 3.6 — Структура модифікованої моделі

3.3.2 Модель на основі InceptionV3

Inception V3 — це архітектура глибокого навчання на основі згорткових нейронних мереж, яка застосовується для класифікації та виділення ознак на зображеннях [32]. Модель є вдосконаленою версією початкової архітектури Inception, вперше представлена як GoogLeNet [32]. Розробка Inception V3 проводилася командою Google і спрямована на підвищення ефективності навчання та точності класифікації [32].

Архітектура Inception V3 (рис. 3.7) налічує 42 шари, що містять різні типи згорткових блоків, агрегування та нормалізацію [32]. У створеній моделі Inception V3 інтегровано разом із шаром глобального усереднення та повноз'єднаними шарами для формування остаточного виходу. Структурна схема цієї моделі наведена на рисунку 3.8.

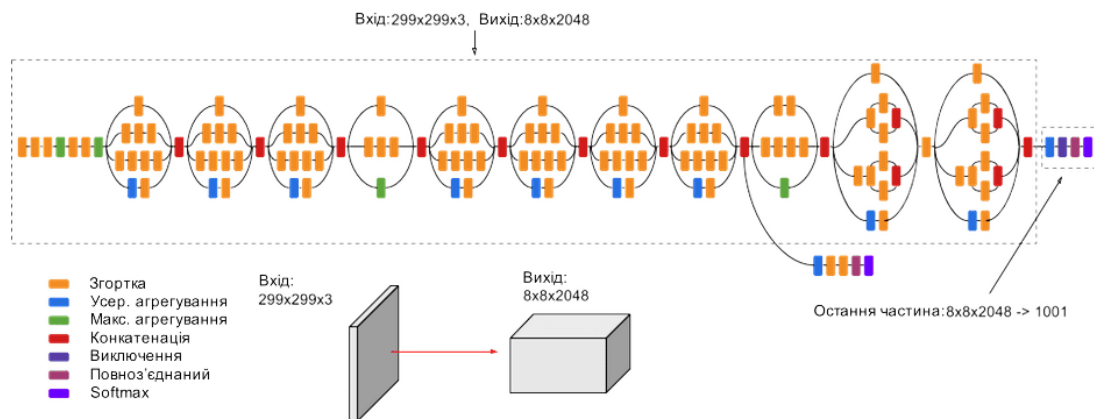


Рисунок 3.7 — Модель Inception V3 [32]

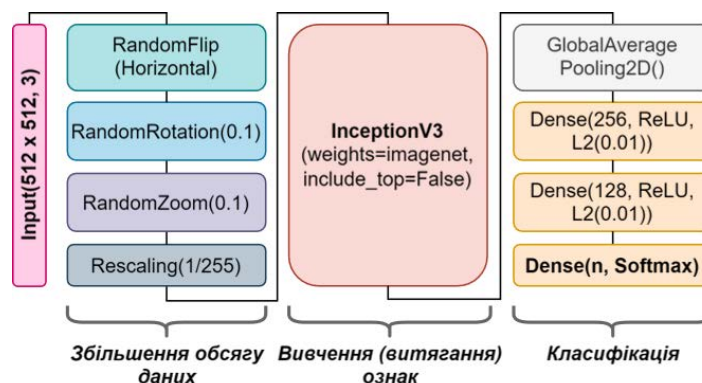


Рисунок 3.8 — Структура моделі з Inception V3

3.3.3 Модель на основі Xception

Xception — це архітектура глибокої ЗНМ, що використовує глибоко-розділені згортки (Depthwise Separable Convolutions) для підвищення ефективності навчання та зменшення кількості параметрів [33]. Модель була розроблена дослідником Франсуа Шоле з команди Google [33]. Основна ідея Xception полягає в тому, що вона інтерпретує модулі Inception як проміжний етап між традиційною згорткою та операцією глибокої розділеної згортки, де кожна глибока згортка супроводжується точковою (1×1) згорткою для поєднання каналів [33].

Архітектура Xception представлена на рисунку 3.9. Створена модель інтегрує Xception разом із шаром глобального усереднення та повноз'єднаними шарами, що забезпечує формування остаточного виходу. Структурна схема цієї моделі наведена на рисунку 3.10.

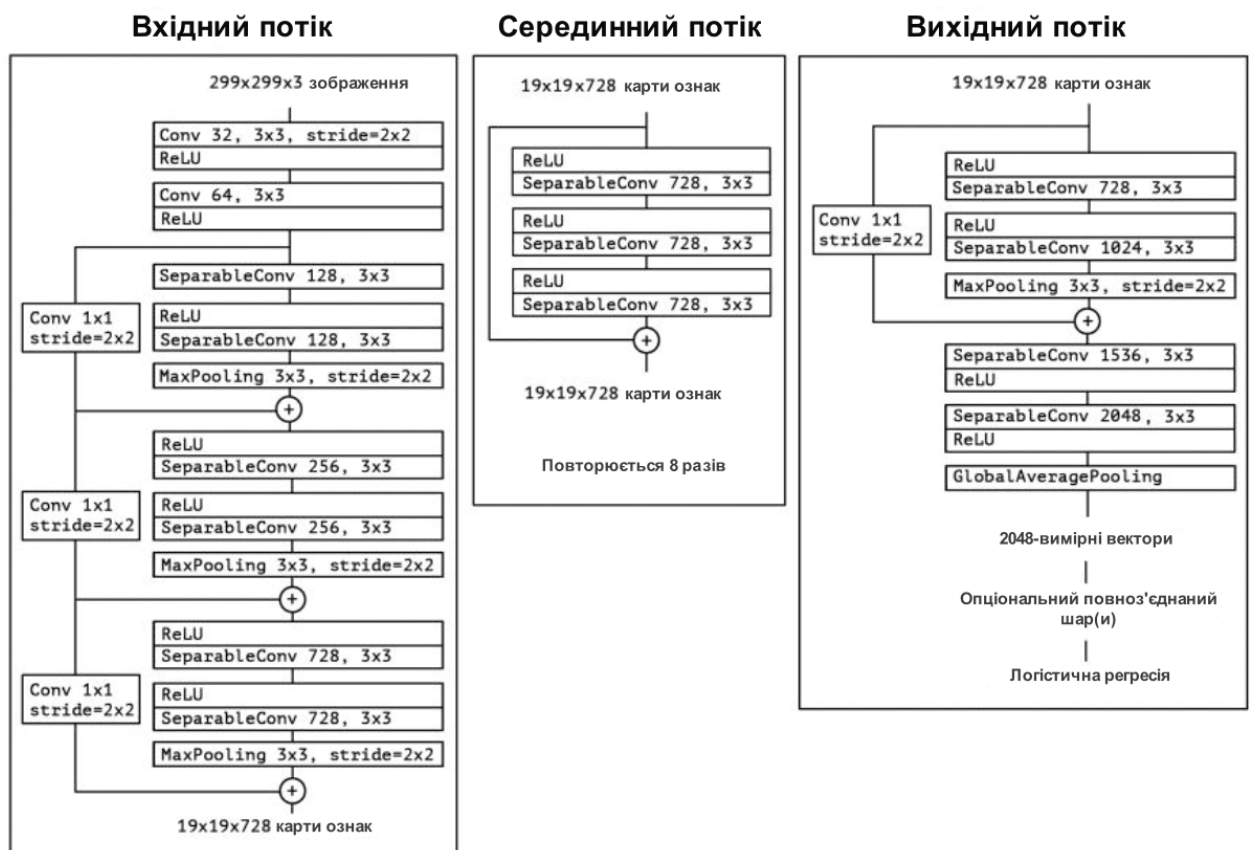


Рисунок 3.9 — Модель Xception [33]

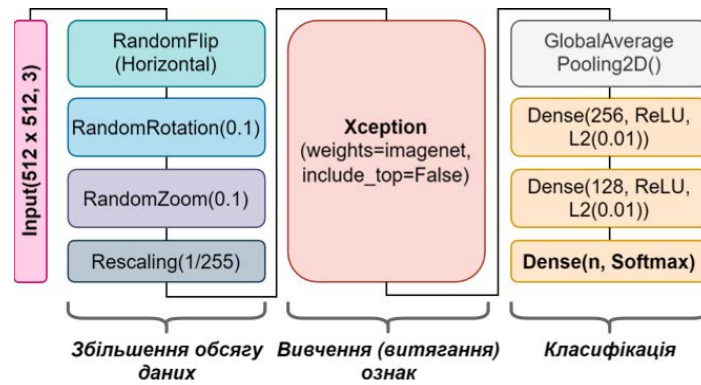


Рисунок 3.10 — Структура моделі з Xception

3.3.4 Модель на основі ResNet50V2

ResNet50 — це 50-шарова ЗНМ, яка містить 48 згорткових шарів, один шар максимізаційного агрегування і один шар усереднювального агрегування [32]. Залишкові нейронні мережі (Residual Networks) будуються шляхом складання залишкових блоків, що дозволяє ефективно передавати градієнти через глибокі шари [32].

Першою архітектурою цієї серії була ResNet34, яка складалася з 34 шарів і вперше продемонструвала можливість збільшення глибини мережі без проблеми зникання градієнта завдяки використанню пропускових з'єднань [32]. Такі з'єднання дозволяють «перескакувати» через деякі шари, перетворюючи звичайну мережу на залишкову [32].

Архітектура ResNet (рис. 3.11) дотримується двох основних принципів проєктування: по-перше, кількість фільтрів у кожному шарі визначається розміром вихідної карти ознак; по-друге, коли розмір карти ознак зменшується вдвічі, кількість фільтрів подвоюється, що дозволяє зберігати обчислювальну складність на рівні шару [32].

У 50-шаровій ResNet використовується концепція «вузьких місць» для залишкових блоків: застосовуються 1×1 згортки для зменшення кількості параметрів та обчислень, що прискорює навчання мережі [32]. Кожен залишковий блок складається зі стеку з трьох шарів замість двох [32].

Назва шару	Розмір	18 шарів	34 шарів	50 шарів	101 шарів	152 шарів
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Рисунок 3.11 — Моделі ResNet [32]

Створена модель інтегрує ResNet50V2 разом із шаром глобального усереднення та повноз'єднаними шарами, що забезпечує формування остаточного виходу. Структурна схема цієї моделі наведена на рисунку 3.12.

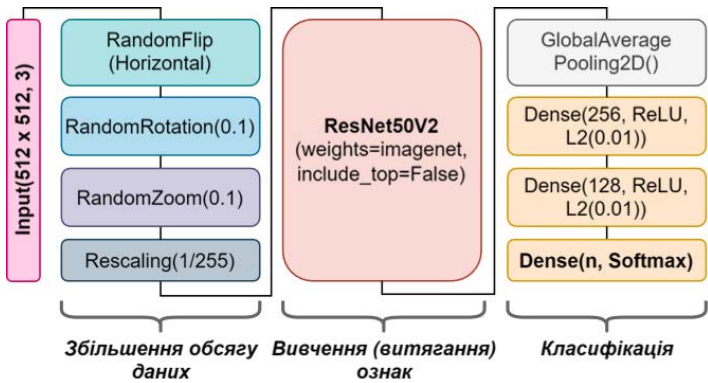


Рисунок 3.12 — Структура моделі з ResNet50V2

3.3.5 Модель на основі MobileNetV2

Моделі MobileNet відомі своєю компактністю та низькою затримкою, що робить їх зручними для розгортання на мобільних пристроях та вбудованих системах [34].

У першій версії архітектури застосовували глибинно-розділені згортки для скорочення обчислювальних витрат [34]. У другій версії

MobileNetV2 вдалося ще більше зменшити кількість параметрів та обчислень без істотної втрати точності [34]. Це досягнуто завдяки блоку «інвертовані залишки та лінійні вузькі місця» (inverted residuals and linear bottlenecks), у якому ознаки спочатку розширюються у більший простір, потім проходять через глибинно-розділену згортку, а на виході стискаються назад до меншого розмірного представлення [34].

Архітектура MobileNetV2 починається з базового двовимірного згорткового шару (рис. 3.13). Далі слідує послідовність блоків «вузьких місць», які відрізняються коефіцієнтом розширення (t), кроком (s), кількістю вихідних каналів (c) та числом повторів (n) [34].

Вхід	Оператор	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	-

Рисунок 3.13 — Модель MobileNetV2 [34]

Створена модель інтегрує MobileNetV2 разом із шаром глобального усереднення та повноз'єднаними шарами, що забезпечує формування остаточного виходу. Структурна схема цієї моделі наведена на рисунку 3.14.

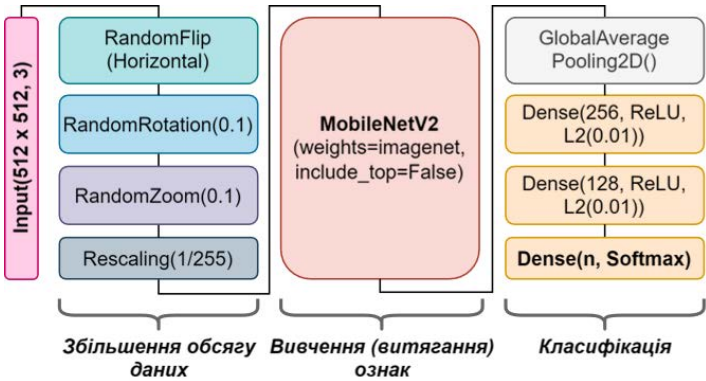


Рисунок 3.14 — Структура моделі з MobileNetV2

3.3.6 Модель на основі DenseNet169

DenseNet169 — це одна з архітектур сімейства DenseNet, що налічує 169 шарів і широко застосовується для задач класифікації зображень [35]. Ця модель має меншу кількість параметрів, що навчаються, порівняно з іншими архітектурами DenseNet з меншою глибиною [35]. DenseNet169 та інші варіанти сімейства DenseNet ефективно долають проблему зникання градієнтів, забезпечують інтенсивне поширення ознак, оптимізують кількість параметрів та сприяють повторному використанню характеристик, що робить ці архітектури надійними та продуктивними для глибинного навчання [35].

Архітектура DenseNet169 (рис. 3.15) містить згорткові шари, шари максимізаційного агрегування, проміжні та повноз'єднані шари. Для всіх шарів використовується функція активації ReLU, а для вихідного шару застосовано Softmax [35].

Шар	Розмір ядра	Параметри	Розмір тензора
Згортка	7×7 (Conv)	Stride = 2, ReLu	112×112
Агрегування	3×3 (MaxPool)	Stride = 2	56×56
Повноз'єднаний шар 1	$1 \times 1 \times 6$ (Conv)	Dropout = 0.2	56×56
	$3 \times 3 \times 6$ (Conv)		
Проміжний шар 1	1×1 (Conv)	Stride = 2	56×56
	2×2 (AvgPool)		28×28
Повноз'єднаний шар 2	$1 \times 1 \times 12$ (Conv)	Dropout = 0.2	28×28
	$3 \times 3 \times 12$ (Conv)		
Проміжний шар 2	1×1 (Conv)	Stride = 2	28×28
	2×2 (AvgPool)		14×14
Повноз'єднаний шар 3	$1 \times 1 \times 32$ (Conv)	Dropout = 0.2	14×14
	$3 \times 3 \times 32$ (Conv)		
Проміжний шар 3	1×1 (Conv)	Stride = 2	14×14
	2×2 (AvgPool)		7×7
Повноз'єднаний шар 4	$1 \times 1 \times 32$ (Conv)	Dropout = 0.2	7×7
	$3 \times 3 \times 32$ (Conv)		
Класифікація	1×1 (Global AvgPool)		1×1
	1000D (fully-connected softmax)		

Рисунок 3.15 — Модель DenseNet169 [35]

Створена модель інтегрує DenseNet169 разом із шаром глобального усереднення та повноз'єднаними шарами, що забезпечує формування остаточного виходу. Структурна схема цієї моделі наведена на рисунку 3.16.

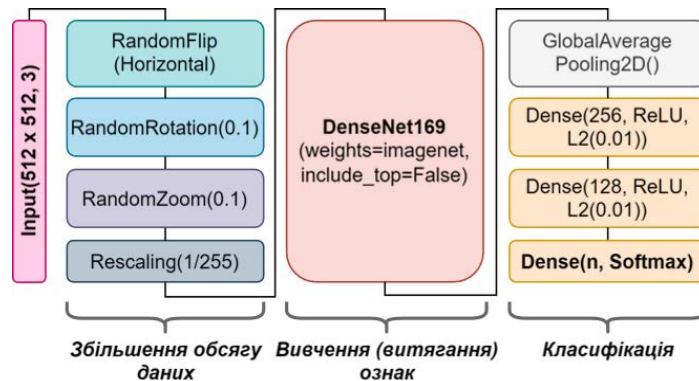


Рисунок 3.16 — Структура моделі з DenseNet169

3.3.7 Порівняння моделей

Для навчальної вибірки було використано 2525 зображень 101 класу їжі з набору даних Food101 (25 зображень на 1 клас їжі), а для тестової вибірки — 1010 зображень 101 класу їжі (10 зображень на 1 клас їжі). В таблиці 3.5 наведено результати навчання моделей за 60 епох. Кількість класів дорівнює 10, 30, 60 та 101, тобто для всіх створених моделей існує 4 варіанти моделі для різної кількості класів. Всього було отримано 24 моделі.

Таблиця 3.5 — Порівняння результатів навчання моделей

Модель	Час навч. (с)	Втрата навч.	Точність навч. (%)	Втрата тест.	Точність тест. (%)	Розмір (Мб)
1	2	3	4	5	6	7
10 класів						
MealNet	741,4	1,86	96	4,71	35	97,7
InceptionV3	3725,9	3,19	95	3,53	72	90,1
Xception	4618,0	2,45	96	2,67	81	86,2
ResNet50V2	4191,0	2,61	100	3,16	71	96,6
MobileNetV2	1536,7	3,22	98	3,86	62	13,1
DenseNet169	5407,3	2,50	97	2,94	76	54,5

Продовження таблиці 3.5

1	2	3	4	5	6	7
30 класів						
MealNet	3120,6	1,28	92	6,02	20	97,7
InceptionV3	7853,2	2,59	83	2,96	63	90,1
Xception	13628,2	2,40	78	2,70	63	86,3
ResNet50V2	10915,3	2,00	98	2,85	64	96,6
MobileNetV2	3415,2	2,67	87	3,42	57	13,1
DenseNet169	13947,9	2,37	80	2,87	58	54,6
60 класів						
MealNet	6127,5	1,44	88	7,73	9	97,7
InceptionV3	17200,9	2,78	61	3,04	52	90,2
Xception	25708,8	2,88	57	3,03	51	86,3
ResNet50V2	22869,0	2,17	87	3,08	54	96,7
MobileNetV2	7738,0	2,85	68	3,46	45	13,1
DenseNet169	26831,0	2,75	65	3,23	47	54,6
101 клас						
MealNet	9602,2	1,70	82	9,41	7	97,8
InceptionV3	28426,0	3,23	46	3,38	39	90,2
Xception	44037,7	3,34	41	3,44	37	86,4
ResNet50V2	35668,5	2,56	72	3,21	48	96,7
MobileNetV2	13798,4	3,27	50	3,70	37	13,2
DenseNet169	46003,3	3,15	49	3,45	38	54,7

Графіки порівнянь точності на навчальних та тестових даних наведено на рисунках 3.17-3.18.

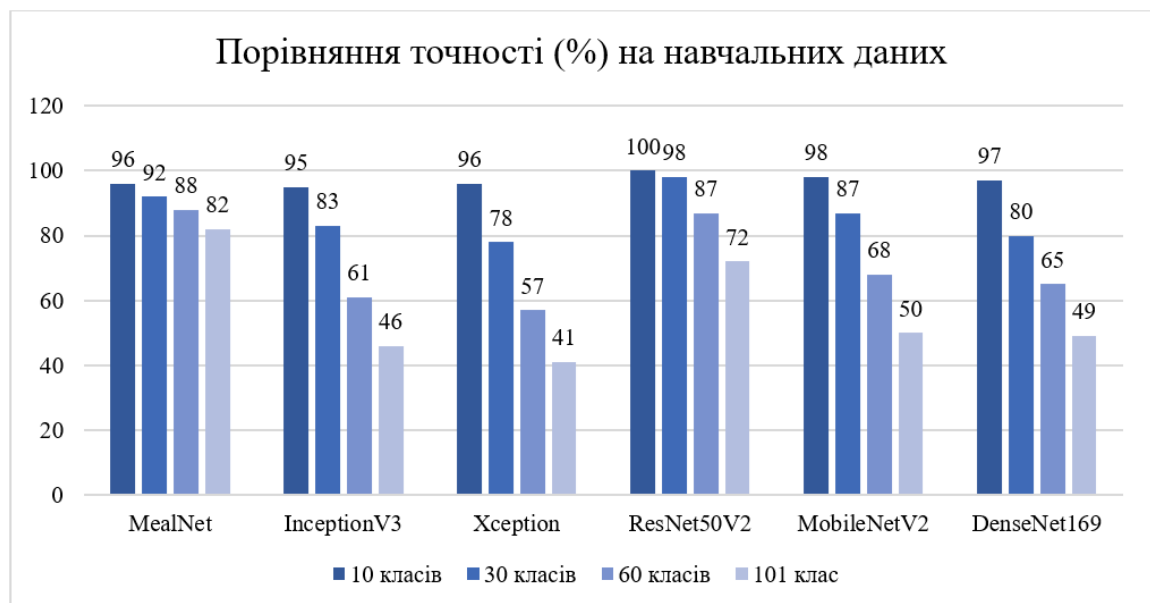


Рисунок 3.17 — Порівняння точності на навчальних даних

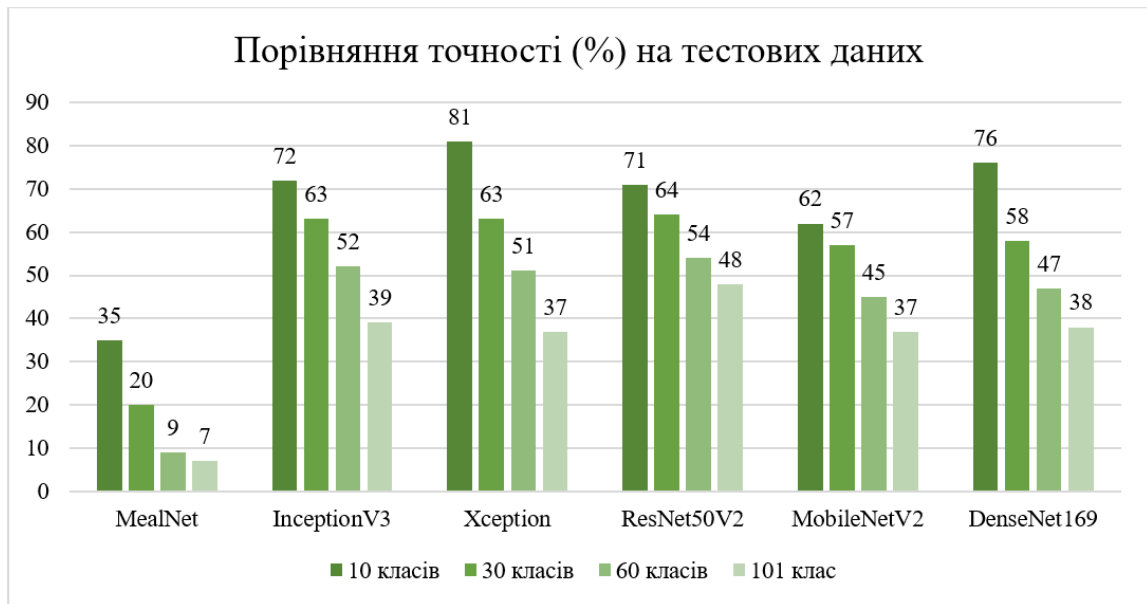


Рисунок 3.18 — Порівняння точності на тестових даних

MealNet (власна модель) показує високі результати точності на навчальній вибірці (96% для 10 класів і 82% для 101 класу). Проте точність тестування є значно нижчою і зменшується зі збільшенням кількості класів: від 35% для 10 до 7% для 101. Це свідчить про те, що модель MealNet має недостатню здатність до узагальнення, ймовірно, через невелику складність архітектури. Але серед усіх моделей швидкість її навчання є найкращою (741,4 с. для 10 класів, 3120,6 с. – 30, 6127,5 с. – 60, 9602,2 с. – 101).

ResNet50V2 демонструє кращу збалансованість між точністю для навчальної (100% для 10 класів і 72% для 101 класу) та тестової вибірок (71% і 48% відповідно). Ця модель показує стабільну здатність до узагальнення навіть при великій кількості класів.

InceptionV3 та Xception також мають високі показники точності навчання (до 95%), однак їх точність тестування залишається нижчою, ніж у ResNet50V2. Крім того, ці моделі потребують значного часу на навчання.

MobileNetV2 має найменший розмір серед моделей, що є значною перевагою для застосунків з обмеженими ресурсами. Проте її точність тестування не перевищує 62% навіть для невеликої кількості класів.

DenseNet169 демонструє середні показники точності тестування (38-76%), але має найбільші вимоги до обчислювальних ресурсів і час навчання.

Попри високі точність та швидкість навчання, MealNet було відхилено як основну модель для розпізнавання через низьку точність тестування. Для задачі з розпізнавання їжі найбільш ефективною виявилася ResNet50V2, яка забезпечує хорошу продуктивність на тестових даних та демонструє стійкість до збільшення кількості класів.

Для детальнішого порівняння ефективності цих двох моделей було розраховано додаткові метрики класифікації — влучність (Precision), повнота (Recall) та F1-міру, які дають змогу оцінити баланс між правильними позитивними та хибними класифікаціями. Результати наведено в таблиці 3.6.

Таблиця 3.6 — Порівняльна оцінка моделей за основними метриками класифікації (%)

Модель	Точність (Accuracy)	Влучність (Precision)	Повнота (Recall)	F1-міра (F1-score)
10 класів				
MealNet	35	33,9	35	31,7
ResNet50V2	71	71,7	71	70,9
30 класів				
MealNet	19,7	19,1	19,7	18,6
ResNet50V2	63,7	66	63,7	63
60 класів				
MealNet	9,2	9,9	9,2	8,9
ResNet50V2	54,3	58	54,3	53,8
101 клас				
MealNet	6,6	8,5	6,6	6,4
ResNet50V2	47,6	50,4	47,6	46,5

За всіма метриками спостерігається подібна тенденція до зниження показників зі збільшенням кількості класів, що є очікуваним результатом через ускладнення задачі класифікації. Проте ResNet50V2 зберігає високий рівень збалансованості між Precision, Recall і F1-score. Навіть при 101 класі всі метрики залишаються близькими ($\approx 47\text{--}50\%$), що свідчить про стабільність узагальнення та стійкість до перенавчання. MealNet, навпаки,

демонструє різке падіння усіх метрик при збільшенні кількості класів — F1-міра знижується з 31,7% (10 класів) до 6,4% (101 клас). Це підтверджує, що власна модель недостатньо ефективно відтворює складні простори ознак і потребує вдосконалення.

Додатковий аналіз за допомогою Precision, Recall і F1-міри підтвердив, що серед розглянутих моделей ResNet50V2 є найбільш збалансованою та надійною. Вона забезпечує оптимальне співвідношення точності, стабільності й обчислювальної ефективності при збільшенні кількості класів. MealNet, попри швидке навчання, виявилась недостатньо гнучкою для багатокласової класифікації, що визначило подальший вибір ResNet50V2 як базової моделі для системи розпізнавання їжі.

3.4 Проєктування бази даних

Інформація програмного забезпечення зберігається у власноруч створеній базі даних, доступ до якої здійснюється за допомогою SQLite. Концептуальна схема цієї бази даних представлена на рисунку 3.19.

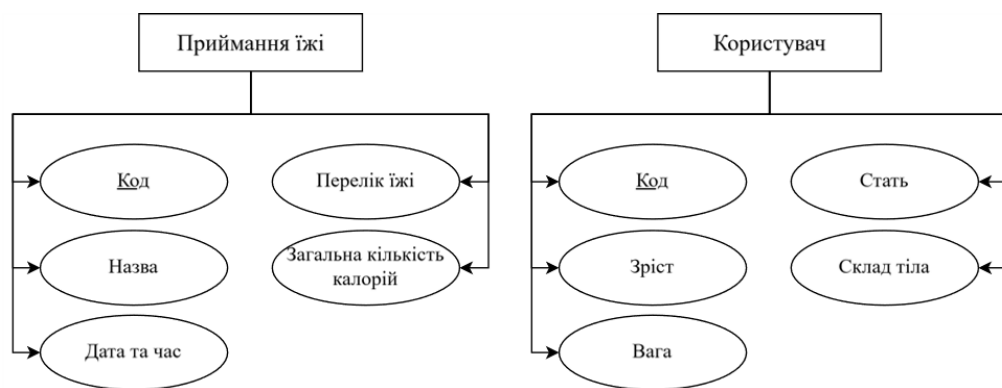


Рисунок 3.19 — Концептуальна схема бази даних

Сутність «Приймання їжі» в базі даних застосунку використовується для зберігання інформації про споживання їжі користувачем. Вона містить кілька важливих полів, що дають змогу відстежувати не лише типи приймання їжі, але й детальну інформацію про харчову цінність спожитої

їжі. Одним з основних полів є «Назва», яке визначає тип приймання їжі: сніданок, обід, вечеря або перекус.

Поле «Дата та час» дозволяє фіксувати точний момент споживання їжі, що важливо для створення хронологічного запису раціону. Це дозволяє користувачам відслідковувати свої звички та зберігати історію, що може бути корисним для контролю над дієтою або медичних рекомендацій.

Поле «Перелік їжі» містить об'єкти, які мають назву, кількість і калорійність. Назви та калорійність їжі можуть бути отримані з каталогу (файлу Food.json), що містить інформацію про їжу, результатів розпізнавання або пошуку за штрихкодом, що дозволяє автоматично поповнювати базу даних актуальними даними без необхідності вручну вводити харчові характеристики їжі. Це забезпечує точність і зручність обробки даних.

Поле «Загальна кількість калорій» обчислюється на основі кількості та калорійності їжі, що входить до переліку. Це поле автоматично підсумовує калорії всієї їжі, що дозволяє користувачам легко контролювати споживання енергії та порівнювати його з рекомендованими нормами.

Сутність «Користувач» у базі даних представляє окрему особу, яка взаємодіє з мобільним застосунком для обліку харчування та відстеження фізіологічних показників. Ця сутність зберігає основну інформацію про користувача, необхідну для персоналізації функціоналу застосунку, розрахунку індивідуальних показників тіла.

Поле «Зріст» зберігає значення зросту користувача в сантиметрах. Цей параметр є необхідним для точного розрахунку індексу маси тіла, добової потреби в калоріях та інших фізіологічних характеристик.

Поле «Вік» фіксує вік користувача у роках. Воно враховується під час обчислення базового метаболізму, визначення метаболічного віку та інших індивідуальних фізіологічних характеристик.

Поле «Стать» відображає біологічну стать користувача. Цей параметр впливає на формули розрахунку жирової маси, м'язової маси, базового метаболізму тощо, оскільки чоловіки і жінки мають різні фізіологічні норми.

Поле «Склад тіла» є складною структурою, що містить набір детальних показників тіла, розрахованих з отриманих із розумних ваг даних. У цій структурі зберігається вага користувача в кілограмах, що є основою для подальших обчислень. Розрахованими показниками є індекс маси тіла, ідеальна вага, жир, вісцеральний жир, м'язова маса, білок, вода, кісткова маса, базовий метаболізм, метаболічний вік, тип тіла.

Опис полів сутностей та їхніх типів наведено у таблиці 3.7.

Таблиця 3.7 — Опис полів сутностей

№	Назва поля	Тип даних	Розмір
Сутність «Приймання їжі»			
1	ID	INTEGER	1-8 байтів
2	MealName	TEXT	Залежить від довжини рядка та обмежень
3	MealDateTime	TEXT	Залежить від довжини рядка та обмежень
4	Food	TEXT	Залежить від довжини рядка та обмежень
5	TotalCalories	REAL	8 байтів
Сутність «Користувач»			
1	ID	INTEGER	1-8 байтів
2	Height	INTEGER	1-8 байтів
3	Age	INTEGER	1-8 байтів
4	Sex	TEXT	Залежить від довжини рядка та обмежень
5	BodyComposition	TEXT	Залежить від довжини рядка та обмежень

3.5 Реалізація взаємодії з відкритою базою даних Open Food Facts

Open Food Facts — це відкрита база даних харчових продуктів з усього світу, яка містить детальну інформацію про склад, харчову цінність, алергени, країну виробництва, штрихкод (EAN) тощо [36]. У межах проєкту ця база використовується для автоматизованого отримання даних (зображення, назва, калорійність) про продукти харчування за їхнім штрихкодом, що значно підвищує зручність та точність обліку.

Взаємодія з базою реалізована за допомогою REST API (інтерфейс, що дозволяє програмі обмінюватися даними з сервером через стандартні HTTP-запити), що надається Open Food Facts [36]. Для запиту до сервера

використовується формат URL, що включає штрихкод продукту. Наприклад, <https://world.openfoodfacts.net/api/v2/product/{barcode}>, де {barcode} — це EAN-код, вручну введений користувачем.

Запит здійснюється асинхронно, що запобігає блокуванню інтерфейсу користувача під час обробки відповіді. Отриманий JSON-об'єкт містить набір ключових полів, таких як `product_name`, `nutriments`, `image_url` тощо. Для подальшої обробки ці дані зберігаються у внутрішню структуру застосунку, після чого користувач може додати продукт до списку приймань їжі.

У випадках, коли продукт не знайдено в базі (наприклад, якщо товар не зареєстрований або введено неправильний штрихкод), користувач може ввести інформацію вручну. Це забезпечує гнучкість системи та підтримку нестандартних або локальних продуктів.

Таким чином, взаємодія з Open Food Facts дозволяє автоматизувати процес введення інформації про харчові продукти, зменшити навантаження на користувача та підвищити точність інформації про харчову цінність продуктів.

3.6 Реалізація взаємодії з розумними вагами Xiaomi Mi Body Composition Scale 2

У межах проєкту реалізовано підтримку взаємодії з розумними вагами Xiaomi Mi Body Composition Scale 2 для автоматичного отримання фізіологічних даних користувача. Комунікація з пристроєм відбувається через протокол Bluetooth Low Energy. Попри наявність певної структури характеристик, основні дані (вага та опір тіла) передаються не через прямі запити, а у вигляді широкомовних повідомлень, які пристрій періодично надсилає без встановлення постійного з'єднання [37].

Ваги передають інформацію через характеристику Body Composition Measurement (UUID 00002a9c-0000-1000-8000-00805f9b34fb), яка надсилає повідомлення довжиною 17 байтів [37]. Перші 4 байти є службовими, а

решта 13 байтів містять корисні дані. Формат повідомлення передбачає закодовану дату та час вимірювання, значення електричного опору тіла, а також вагу. Усі числові значення передаються в порядку little-endian, тобто від молодшого байта до старшого.

Після отримання даних вони обробляються в мобільному застосунку за допомогою математичних моделей, що реалізовані в спеціальній бібліотеці `MiScaleBodyComposition`. Ця бібліотека дозволяє розрахувати додаткові показники складу тіла, зокрема: індекс маси тіла, відсоток жиру, вісцеральний жир, м'язову масу, рівень білка та води, кісткову масу, базовий метаболізм, метаболічний вік та тип тіла. Для точності розрахунків використовуються персональні дані користувача, зокрема стать, вік та зріст, що вже збережені в профілі.

Реалізація взаємодії з вагами дозволяє автоматизувати процес збору фізіологічних показників та забезпечити точнішу оцінку стану тіла. Це значно підвищує інформативність та корисність мобільного застосунку.

3.7 Основні рішення щодо шифрування зображень

Для реалізації функції шифрування зображень їжі у програмі були створені окремі модулі, що виконують операції шифрування та розшифрування даних за допомогою симетричного алгоритму AES у режимі CFB, асиметричного алгоритму RSA, а також їхньої комбінації. Отримані результати застосовувалися для порівняння ефективності методів, аналізу та вибору оптимального способу захисту візуальної інформації.

3.7.1 AES

AES (Advanced Encryption Standard) — це симетричний блочний шифр, який оперує на фіксованих блоках даних (наприклад, 128 біт) і використовує ключі різної довжини [38]. Алгоритм AES містить низку

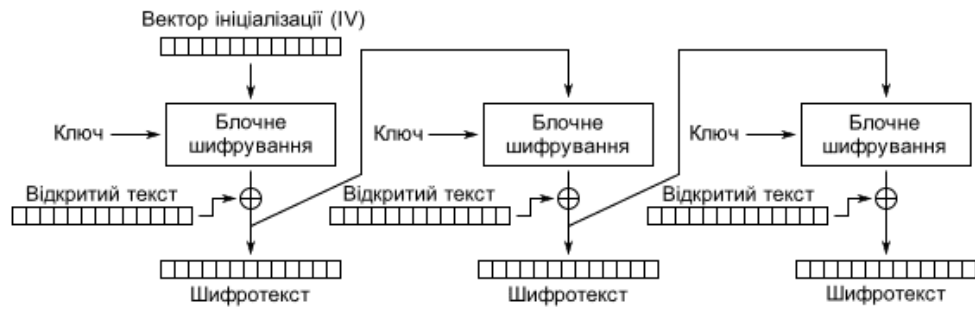
раундів трансформацій, серед яких — байтові підстановки (SubBytes), циклічні зсуви (ShiftRows), лінійні перетворення стовпців (MixColumns) та додавання раундового ключа (AddRoundKey) [38]. Ці операції повторюються, формуючи складний нелінійний механізм шифрування, здатний забезпечити високий рівень безпеки при правильній генерації та управлінні ключем [38].

У контексті AES ключовий простір розширюється через процедуру «key schedule», за якою початковий ключ (майстер-ключ) трансформується в набір раундових ключів, що використовуються на кожному раунді шифрування [38]. Ця схема ключів є критичною для безпеки й ефективності AES, адже від неї залежить рандомізація та зміна ключів між раундами.

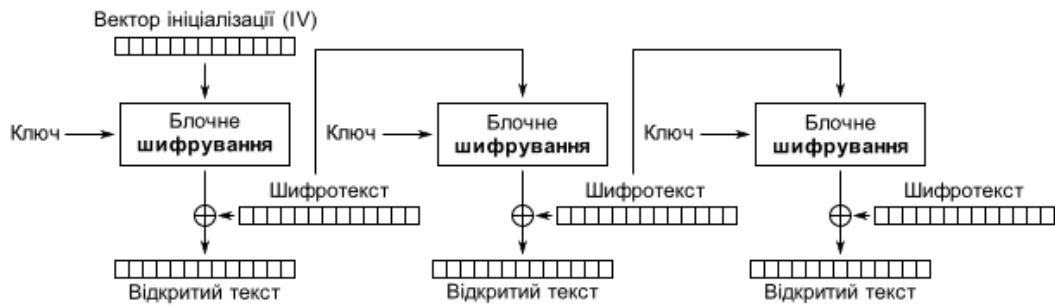
Cipher Feedback (CFB) — це режим роботи блочного шифру, який перетворює AES на самосинхронізований потоковий шифр [38]. У цьому режимі ініціалізаційний вектор (IV) шифрується стандартним блоковим шифром, а результат подається на виключне АБО із відкритим повідомленням, забезпечуючи шифротекст першого сегмента [38]. Потім цей шифротекст «повертається назад» (feedback) у шифруючий блок, утворюючи новий вхід для шифрування наступного сегменту, і так далі [38]. Такий підхід поєднує властивості блочного шифру із гнучкістю потокового шифру, дозволяючи працювати з фрагментами даних змінної довжини.

CFB має властивість самосинхронізації: якщо частина шифротексту втрачена або пошкоджена, шифрування/дешифрування може повернутися до синхронного стану після декількох блоків [38]. Проте цей режим також вносить затримку, оскільки кожен новий блок залежить від попереднього шифротексту, що накладає послідовну природу роботи та обмежує можливість паралелізації [38].

Процес шифрування та розшифрування даних у режимі CFB ілюструється на рисунку 3.20.



Шифрування в режимі зворотного зв'язку за шифротекстом (CFB)



Розшифрування в режимі зворотного зв'язку за шифротекстом (CFB)

Рисунок 3.20 — Схема шифрування та розшифрування в режимі CFB [38]

Операції шифрування й розшифрування в цьому режимі можна подати математично за допомогою формул (3.8-3.10) [38]:

$$C_0 = IV, \quad (3.8)$$

$$C_i = E_k(C_{i-1}) \oplus P_i, \quad (3.9)$$

$$P_i = E_k(C_{i-1}) \oplus C_i, \quad (3.10)$$

де IV – вектор ініціалізації;

i – номер блоку;

k – ключ;

C_i – зашифрований блок (шифротекст);

P_i – блок відкритого тексту;

E_k – функція блокового шифрування.

3.7.2 RSA

RSA — це класичний асиметричний криптографічний алгоритм, безпека якого ґрунтується на складності факторизації великих простих чисел [39]. У своїй основі процес RSA містить генерацію пари ключів: приватного й публічного [39]. Спочатку вибираються два великі прості числа, після чого обчислюється їхній добуток, який слугує модулем шифрування, а також функція Ейлера (ϕ) за цими числами [39]. Потім підбирають публічну експоненту так, щоб вона була взаємно простою з ϕ , і на її основі обчислюють приватну експоненту, обернену за модулем ϕ [39]. Така конструкція гарантує, що публічний ключ (експонента та модуль) може використовуватись для шифрування повідомлень, а приватний — для їх дешифрування, причому операції шифрування й дешифрування здійснюються за допомогою модульного піднесення до степеня [39].

У процесі шифрування повідомлення перетворюється на ціле число, до якого застосовується публічна експонента в степені модулю, утворюючи шифротекст [39]. Для відновлення оригінального повідомлення шифротекст підноситься до степеня приватної експоненти знову під модулем, що повертає вихідні дані [39]. Такий механізм не лише гарантує конфіденційність даних, але й дозволяє підтримувати цілісність та автентичність через цифрові підписи: хеш повідомлення спочатку обробляється, а потім підписується за допомогою приватного ключа, — при верифікації підпису застосовується публічний ключ, і результат порівнюється з хешем отриманого повідомлення [39]. У цілому, RSA залишається надійним і широко розповсюдженим рішенням для захисту чутливої інформації, особливо у сценаріях, де важлива безпечна передача, цифровий підпис і конфіденційний обмін ключами.

Процеси шифрування та розшифрування за допомогою асиметричного методу можуть бути представлені формулами (3.11-3.15) [39]:

$$n = p \cdot q, \quad (3.11)$$

$$\varphi(n) = (p-1)(q-1), \quad (3.12)$$

$$1 < e \leq \varphi(n), \text{ НСД}(e, \varphi(n)) = 1, \quad (3.13)$$

$$d = e^{-1} \pmod{\varphi(n)}, \quad (3.14)$$

$$C = M^e \pmod{n}, \quad M = C^d \pmod{n}, \quad (3.15)$$

де n – модуль;

p та q – великі прості числа;

$\varphi(n)$ – функція Ейлера;

e та d – відкрита й секретна експоненти;

C – зашифроване повідомлення (шифротекст);

M – повідомлення (відкритий текст).

3.7.3 Гібридне шифрування

Процес гібридного шифрування зазвичай містить чотири основні стадії: створення симетричного ключа, шифрування даних цим ключем, подальше шифрування самого симетричного ключа відкритим ключем та передавання одержувачу пари «зашифровані дані + зашифрований ключ» [40]. Симетричний ключ може бути сформований генератором випадкових чисел або отриманий на основі паролльної фрази чи попередньо узгодженого секрету [40]. Дані шифруються за допомогою будь-якого симетричного алгоритму, наприклад AES [40]. Далі симетричний ключ шифрується асиметричним методом, таким як RSA, використовуючи відкритий ключ одержувача [40]. Після цього зашифровані дані та зашифрований симетричний ключ передаються адресату, який за допомогою приватного ключа розшифровує симетричний ключ і надалі відновлює оригінальні дані, застосовуючи відповідний симетричний алгоритм [40].

3.7.4 Отримані результати

Для захисту даних зображень їжі, що передаються з мобільного клієнта на сервер, у програму інтегровано криптографічний модуль на основі гібридного підходу до шифрування (рис. 3.21). Цей модуль поєднує AES для шифрування вмісту зображення та RSA для безпечної передачі симетричного ключа. Гібридна модель забезпечує баланс між обчислювальною ефективністю та високим рівнем безпеки.

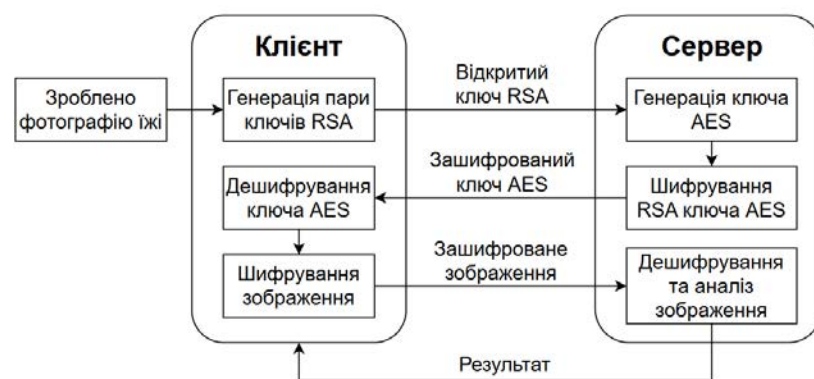


Рисунок 3.21 — Шифрування та передача зображень їжі

У мобільному клієнті, реалізованому за допомогою .NET MAUI, було розроблено спеціальний клас під назвою `Cryptography` з використанням простору імен `System.Security.Cryptography`. Цей клас відповідає за виконання таких завдань:

- генерація пари ключів RSA: після фотографування їжі на клієнтському пристрої генерується пара ключів RSA довжиною 2048 біт. Закритий ключ зберігається в таємниці, а відкритий ключ надсилається на сервер;

- розшифрування ключа AES: симетричний ключ AES, згенерований і зашифрований за допомогою відкритого ключа RSA клієнта на сервері, отримується і розшифровується за допомогою закритого ключа RSA клієнта;

- шифрування зображення: розшифрований ключ AES потім використовується для шифрування зробленої фотографії їжі в режимі AES-

256 Cipher Feedback (CFB). Це зашифроване зображення далі безпечно передається на сервер для аналізу.

Ця реалізація гарантує, що через мережу передаються лише зашифровані дані. Зображення у вигляді відкритого тексту ніколи не залишає пристрій у незашифрованому вигляді.

На стороні сервера було реалізовано криптографічний модуль на мові Python під назвою Cryptography з використанням бібліотеки PyCryptodome. Цей модуль відповідає за наступне:

- генерація ключа AES: випадковий 256-бітний ключ AES генерується на сервері за допомогою генератора випадкових чисел щоразу, коли починається нова сесія клієнта;

- шифрування ключа AES за допомогою RSA: ключ AES шифрується за допомогою відкритого ключа RSA, отриманого від клієнта, і повертається в зашифрованому вигляді;

- розшифрування та обробка зображення: після отримання зашифрованого зображення їжі воно розшифровується за допомогою ключа AES. Потім зображення у вигляді відкритого тексту передається до системи класифікації їжі, яка використовує модель глибокого навчання (ResNet50V2) для ідентифікації категорії їжі.

Ця інтеграція забезпечує чітке розділення криптографічних функцій і логіки аналізу зображень, підвищуючи модульність і зручність обслуговування.

Завдяки поєднанню стандартних криптографічних бібліотек (System.Security.Cryptography і PyCryptodome) програма використовує перевірені інструменти для безпечного та ефективного гібридного шифрування на різних платформах.

Для оцінки продуктивності шифрування було виміряно час виконання шифрування та дешифрування даних зображення за трьома криптографічними підходами: AES (симетричний), RSA (асиметричний) та гібридна комбінація AES і RSA. Кожен тест було повторено 3 рази та

усереднено для забезпечення узгодженості. Основним показником був середній час виконання (у секундах), необхідний для безпечної підготовки зображення до передачі та класифікації. У таблиці 3.8 наведено порівняльний аналіз трьох підходів до шифрування на основі декількох критеріїв.

Таблиця 3.8 — Порівняння методів шифрування [38]-[40]

Критерії порівняння	AES	RSA	AES та RSA
Ключі	Один спільний ключ	Приватний і публічний ключі	Комбінація ключів
Розмір ключа	256 біт	2048 біт	256 та 2048 біт
Швидкодія	2,06 с.	34,51 с.	1,01 с.
Обчислювальні витрати	Менші	Більші	Середні
Передача ключів	Складна	Легка	Легка
Складність управління ключами	Висока	Низька	Низька

Оцінка продуктивності показує, що гібридний підхід перевершує обидва окремі методи за часом виконання. Хоча AES і так значно швидший за RSA завдяки своїм легким операціям, гібридний метод ще більше оптимізує продуктивність, використовуючи RSA виключно для безпечного обміну ключем AES, мінімізуючи витрати RSA. Крім того, гібридна модель спрощує розподіл і управління ключами без шкоди для безпеки, що робить її оптимальним вибором для клієнт-серверних застосунків у режимі реального часу, що передбачають роботу з чутливими медіа-даними, такими як зображення їжі.

3.8 Основні рішення щодо розробки інтерфейсу

Під час розробки мобільного застосунку були створені основні елементи графічного інтерфейсу користувача:

- вкладка «Моє харчування» (рис. 3.22), яка містить відсортований перелік приймання їжі, параметри сортування та кнопку «Додати приймання їжі» для переходу до додавання приймання їжі;

- сторінка «Статистика» (рис. 3.23), яка відображає статистику калорійності приймань їжі за обраний день та сумарної калорійності за днями відповідного тижня;
- сторінка «Додати приймання їжі» (рис. 3.23), яка містить поля для введення інформації про приймання їжі та кнопку «Зберегти»;
- сторінка «Переглянути приймання їжі» (рис. 3.23), яка містить поля з інформацією про приймання їжі та кнопку «Зберегти»;
- вкладка «Розпізнавання» (рис. 3.22), яка надає можливість зробити фотографію їжі для розпізнавання за допомогою кнопки «Зображення їжі», ввести штрихкод продукту, відображає фотографію їжі або продукту, результат розпізнавання або пошуку та містить кнопку «Додати» для переходу до додавання приймання їжі з використанням результату;
- вкладка «Інфо» (рис. 3.22), яка містить інформацію про користувача, застосунок, розробника та кнопки «Під'єднатися до ваг» і «Зберегти».

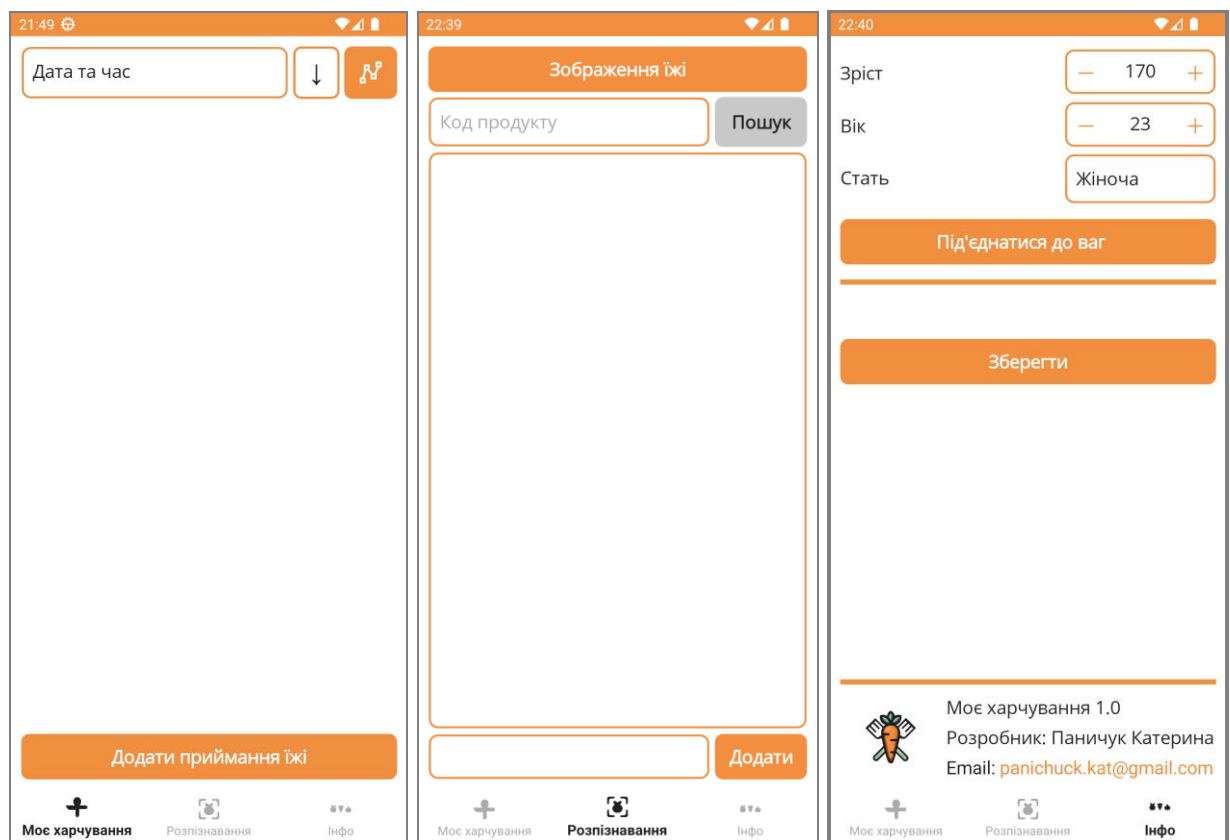


Рисунок 3.22 — Вкладки «Моє харчування», «Розпізнавання» та «Інфо»

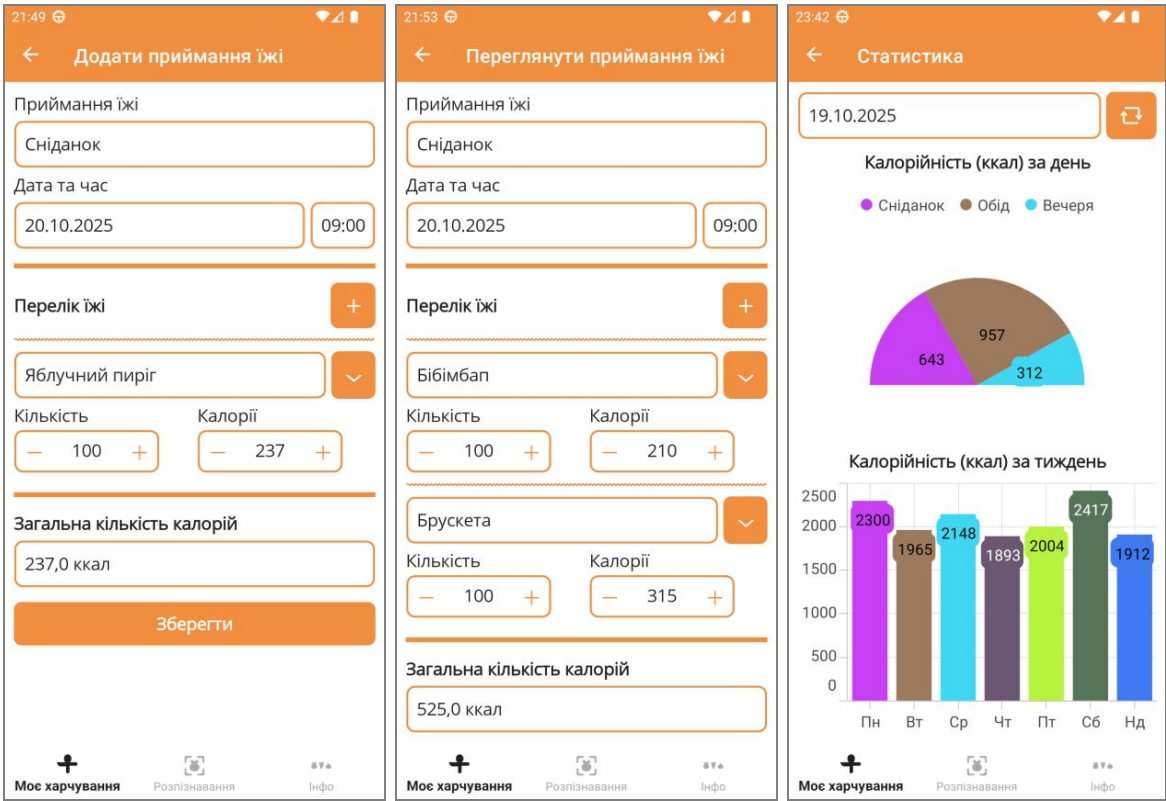


Рисунок 3.23 — Сторінки «Додати приймання їжі», «Переглянути приймання їжі» та «Статистика»

Для оформлення застосунку обрано три основні кольори, представлені на рисунку 3.24, а також розроблено логотип та зображення вкладок, які наведено на рисунку 3.25.



Рисунок 3.24 — Кольори для застосунку

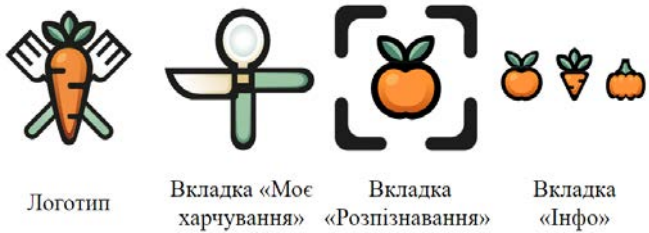


Рисунок 3.25 — Зображення для застосунку

Інтерфейс користувача містить такі компоненти:

- мітки для відображення опису елементів та важливої інформації для користувача;
- кнопки для взаємодії з програмою;
- графіки для відображення статистики;
- поля для введення тексту, вибору дати та часу;
- лічильники для введення числових значень;
- зображення для демонстрації фотографій продуктів або їжі;
- список, що розкривається, для вибору назви приймання їжі, назви їжі, поля для сортування переліку приймання їжі, статі користувача.

При зміні орієнтації пристрою, що визначається за допомогою гіроскопа, інтерфейс застосунку автоматично адаптується під новий режим відображення (рис. 3.26).

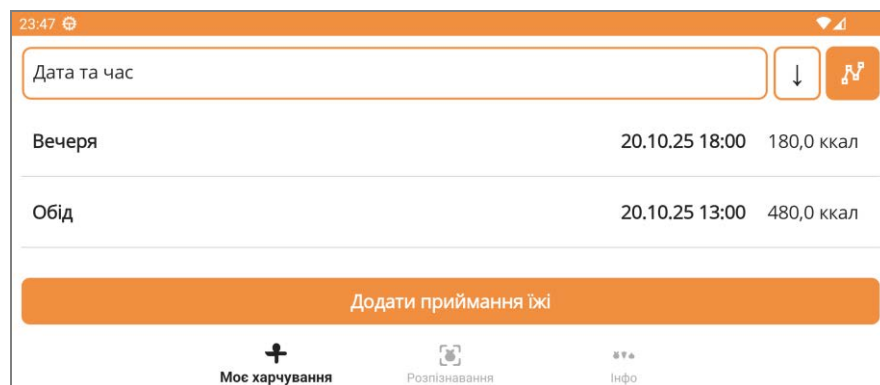


Рисунок 3.26 — Адаптація дизайну

3.9 Обробка виняткових ситуацій

Для запобігання помилок у роботі застосунку передбачено різноманітні перевірки та обмеження, а в окремих випадках — повідомлення з текстом помилки.

Крім того, у застосунку передбачено обробку виняткових ситуацій, які можуть виникати під час взаємодії з серверною частиною, відкритою базою даних Open Food Facts або розумними вагами.

3.10 Висновки за розділом 3

У цьому розділі представлено вибраний шаблон проектування та описано архітектуру створеного програмного забезпечення. Також наведено структури розроблених моделей для задачі розпізнавання їжі. Для реалізації системи розпізнавання їжі було обрано модель ResNet50V2, оскільки вона забезпечує оптимальний баланс між точністю для навчальної та тестової вибірок і демонструє стабільну здатність до узагальнення навіть за збільшення кількості класів. Ця модель показує високу точність тестування (71% для 10 класів і 48% для 101 класу) у порівнянні з іншими архітектурами, що свідчить про її ефективність для задач класифікації зображень їжі.

Було розроблено концептуальну модель бази даних із сутностями «Приймання їжі» та «Користувач», для яких визначено поля, типи та розміри даних. Також було описано реалізації взаємодій з відкритою базою даних Open Food Facts та розумними вагами Xiaomi Mi Body Composition Scale 2.

Окрім цього, розглянуто методи шифрування даних AES, RSA та їх комбінацію. Аналіз показав, що гібридне шифрування є найбільш ефективним для застосунку, забезпечуючи безпечну передачу даних.

Також представлено основні рішення щодо розробки графічного інтерфейсу користувача: описано сторінки мобільного застосунку, їх функціональне призначення, використані елементи та кольорову схему, а також вплив орієнтації пристрою на дизайн.

Крім того, зазначено потенційні виняткові ситуації та повідомлення про помилки, які можуть виникати під час роботи з програмним забезпеченням.

4 КЕРІВНИЦТВО ПРОГРАМІСТА

4.1 Призначення та умови застосування програми

4.1.1 Призначення програми

Призначенням програми є надання зручного інструменту для обліку харчування, дозволяючи користувачам зробити більш обґрунтовані рішення щодо свого раціону та підтримувати здоровий баланс калорій у повсякденному житті.

4.1.2 Функції програми

Програма виконує наступні функції:

- додавання, редагування, видалення інформації про приймання їжі;
- сортування переліку приймання їжі;
- виведення статистики калорійності приймань їжі;
- надання можливості зробити фотографію їжі, її шифрування, розшифрування та подальше розпізнавання;
- отримання даних про продукт з відкритої бази даних Open Food Facts за допомогою штрихкоду;
- редагування інформації про користувача;
- взаємодія з розумними вагами Xiaomi Mi Body Composition Scale 2 для отримання значень ваги та опору тіла, розрахунку показників тіла за даними про користувача.

4.1.3 Умови застосування програми

Мінімальні вимоги до апаратного та програмного забезпечення для стабільної роботи мобільного застосунку визначені наступним чином:

- операційна система: Android 5.0 або новіша, iOS 11 або новіша;
- процесор: двоядерний з частотою 1,5 ГГц або вище;

- вільний простір на диску: щонайменше 250 Мб;
- оперативна пам'ять: не менше 2 Гб;
- підключення до інтернету;
- наявність камери, модуля Bluetooth та GPS.

Для нормальної роботи серверної частини застосунку мінімальні вимоги такі:

- програмне забезпечення: Python 3.9, Flask 3.0.3, Tensorflow 2.15.0, Keras 2.15.0, NumPy 1.26.4, Pillow 9.4.0, Matplotlib 3.9.2, PyCryptodome 3.21.0;
- процесор: двоядерний із частотою не менше 2 ГГц;
- вільний простір на диску: не менше 5 Гб;
- оперативна пам'ять: щонайменше 4 Гб;
- підключення до інтернету.

4.2 Характеристика програми

Мобільний застосунок розроблено із застосуванням фреймворку .NET MAUI та мови програмування C# у середовищі Microsoft Visual Studio.

Серверна частина реалізована на Python із використанням бібліотек Flask, Tensorflow, Keras, NumPy, Pillow, Matplotlib та PyCryptodome також у середовищі Microsoft Visual Studio.

4.2.1 Структура програми

Архітектура мобільного застосунку містить 25 класів та 7 файлів XAML (eXtensible Application Markup Language), які пов'язані з 7 класами. Класи моделей представлень і представлень мають у назві відповідно суфікси VM та V. Всі файли організовані у теках згідно з їх функціональним призначенням. Класи, що розміщені у папці «General», відповідають за підтримку взаємодії між представленнями та моделями представлень. Структура файлів мобільного застосунку наведена на рисунку 4.1.

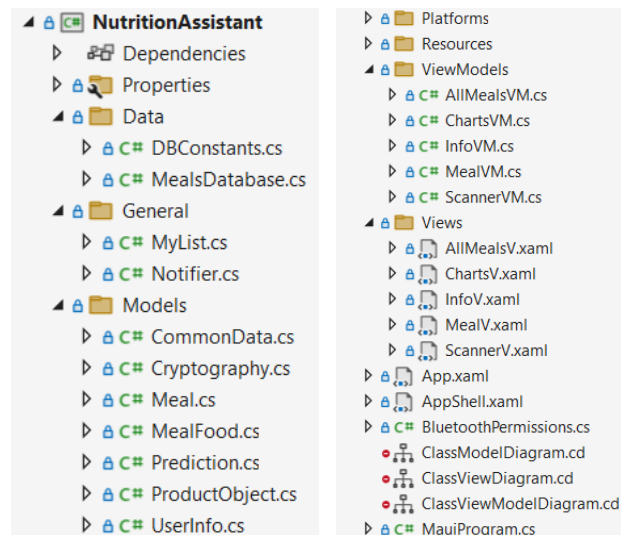


Рисунок 4.1 — Структура файлів мобільного застосунку

Серверна частина програми побудована з 6 файлів: 4 з них призначені для створення моделі розпізнавання їжі, а решта відповідають за обробку HTTP-запитів, шифрування, розшифрування та безпосереднє розпізнавання.

Структура файлів серверної частини представлена на рисунку 4.2.

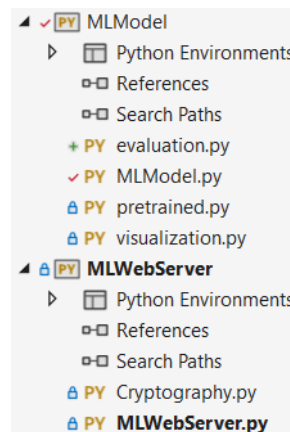


Рисунок 4.2 — Структура файлів серверної частини програми

4.3 Звертання до програми

Перед початком роботи програми необхідно переконатися, що пристрій відповідає мінімальним системним вимогам. Також слід перевірити наявність усіх необхідних файлів застосунку. Для коректного виконання

HTTP-запитів потрібно впевнитися, що середовище розробки має всі компоненти, що забезпечують роботу з інтернетом.

Щоб запустити проекти мобільного застосунку та серверної частини у Microsoft Visual Studio, необхідно відкрити відповідні файли рішення: NutritionAssistant.sln для мобільного застосунку та MLWebServer.sln для серверної частини.

Вихідними файлами мобільного застосунку та серверної частини є com.panikat.nutritionassistant-Signed.apk (для ОС Android) і MLModel.py, pretrained.py, evaluation.py, visualization.py, MLWebServer.py та Cryptography.py.

4.4 Вхідні та вихідні дані

4.4.1 Вхідні дані

Вхідні дані програми містять інформацію про приймання їжі та користувача, яка зберігається в базі даних, про їжу, яка зберігається в файлі JSON, графічні зображення для інтерфейсу користувача, модель для розпізнавання, що використовується на серверній частині, а також зображення, які застосовуються для навчання цієї моделі.

4.4.2 Вихідні дані

Вихідні дані програми містять оновлену інформацію про приймання їжі та користувача, збережену у БД, а також модель для розпізнавання.

4.5 Повідомлення програмісту

Під час роботи з мобільним застосунком програміст може побачити наступні повідомлення.

«Помилка на стороні клієнта! Будь ласка, спробуйте пізніше або зверніться до служби підтримки» — сигналізує про помилку на клієнтській частині під час надсилання запиту до сервера. Необхідно перевірити коректність налаштування зв'язку з серверною частиною або підключення до інтернету.

«Помилка на стороні сервера! Будь ласка, спробуйте пізніше або зверніться до служби підтримки» — свідчить про помилку на сервері під час обробки запиту. Програмісту слід переконатися в правильності роботи серверної частини.

«Помилка! Будь ласка, спробуйте пізніше або зверніться до служби підтримки» (вкладка «Розпізнавання») — повідомляє про збій при надсиланні або обробці запиту до серверної частини чи відкритої бази даних. Рекомендується перевірити налаштування зв'язку з сервером або базою даних, стабільність інтернет-з'єднання та працездатність серверної частини.

«Помилка! Будь ласка, спробуйте пізніше або зверніться до служби підтримки» (вкладка «Інфо») — свідчить про помилку під час підключення до розумних ваг Xiaomi Mi Body Composition Scale 2. Програмісту слід переконатися у правильності налаштувань зв'язку з вагами, увімкнених Bluetooth та GPS, а також наданні відповідних дозволів на пристрої.

«Bluetooth вимкнено. Увімкніть його для виконання операції» — повідомляє про відсутність активного Bluetooth, що унеможливорює виконання операції. Потрібно перевірити наявність увімкненого Bluetooth та його налаштування.

«GPS вимкнено. Увімкніть його для виконання операції» — сигналізує про відключений GPS, через що операція неможлива. Рекомендується переконатися, що GPS увімкнено та налаштовано правильно.

4.6 Висновки за розділом 4

У цьому розділі було описано призначення та основні функції розробленого програмного забезпечення, а також наведено мінімальні системні вимоги для його коректного функціонування. Було охарактеризовано структуру програми та наведено детальний опис вхідних і вихідних даних. Крім того, представлено перелік усіх текстових повідомлень, які можуть відображатися під час взаємодії програміста з програмним забезпеченням.

5 КЕРІВНИЦТВО ОПЕРАТОРА

5.1 Призначення програми

Призначенням програми є надання зручного інструменту для обліку харчування, дозволяючи користувачам зробити більш обґрунтовані рішення щодо свого раціону та підтримувати здоровий баланс калорій.

5.2 Умови виконання програми

5.2.1 Вимоги програми

Мінімальні вимоги до апаратного та програмного забезпечення для стабільної роботи мобільного застосунку визначені наступним чином:

- операційна система: Android 5.0 або новіша, iOS 11 або новіша;
- процесор: двоядерний з частотою 1,5 ГГц або вище;
- вільний простір на диску: щонайменше 250 Мб;
- оперативна пам'ять: не менше 2 Гб;
- підключення до інтернету та наявність камери, Bluetooth і GPS.

5.2.2 Вимоги до користувача

Основною вимогою до користувача є базове вміння працювати з мобільним пристроєм.

5.3 Виконання програми

5.3.1 Запуск програми

Спочатку необхідно завантажити мобільний застосунок за допомогою файлу com.panikat.nutritionassistant-Signed.apk для ОС Android. Після цього на пристрої з'явиться іконка застосунку з назвою «Моє харчування», яку наведено на рисунку 5.1.

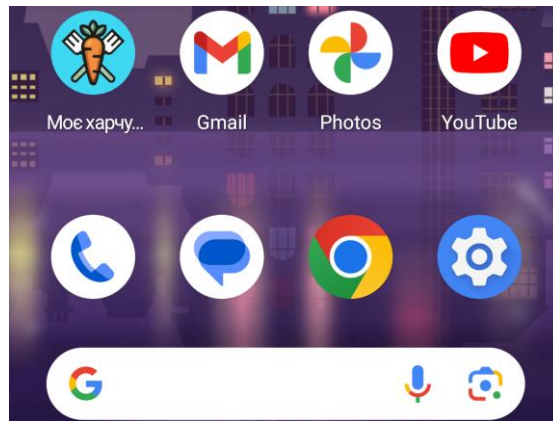


Рисунок 5.1 — Іконка встановленого мобільного застосунку

Після запуску застосунку відображається початковий екран із іконкою програми. Потім буде відкрита перша вкладка з назвою «Моє харчування», яку зображено на рисунку 5.2.

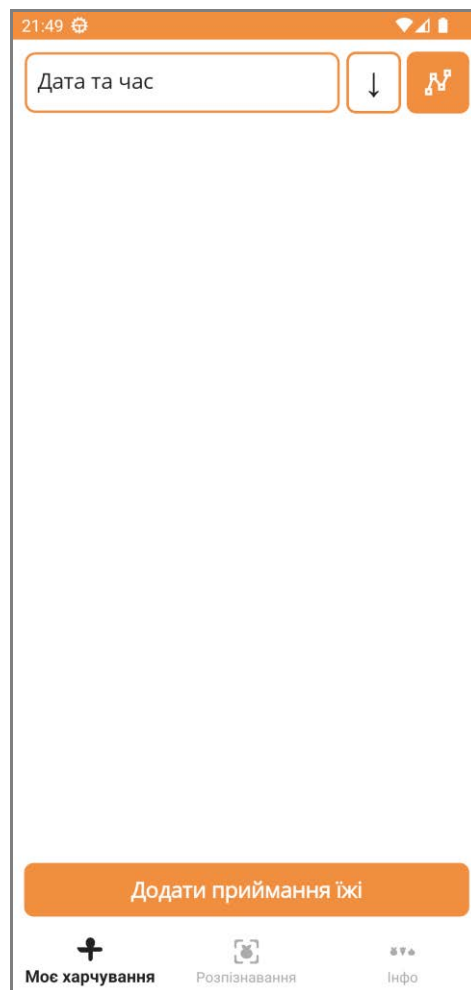


Рисунок 5.2 — Вкладка «Моє харчування»

5.3.2 Робота з програмою

Після запуску програми буде відкрита перша вкладка з назвою «Моє харчування», яку зображено на рисунку 5.2. Вона містить перелік приймання їжі, в якому будуть приймання, після їхнього додавання, список, що розкривається та містить назви полів для сортування, кнопку зі стрілкою, яка встановлює порядок сортування, кнопки «Додати приймання їжі» та «Статистика». Приклад змінених параметрів сортування наведено на рис. 5.3.



Рисунок 5.3 — Змінені параметри сортування

Після натискання на кнопку «Додати приймання їжі» з'явиться сторінка додавання приймання їжі (рис. 5.4). Вона містить наступні поля: «Приймання їжі», «Дата та час», «Перелік їжі» («Назва», «Кількість», «Калорії»), «Загальна кількість калорій». Дані в ці поля вводяться за допомогою списів, що розкриваються, календарів, годинників, лічильників, текстових полів та кнопок. Поле «Калорії» заповнюється автоматично при виборі їжі з каталогу (при натисканні на кнопку «✓»), розпізнаванні та

пошуку у відкритій базі даних. Поле «Загальна кількість калорій» розраховується автоматично.

Для додавання їжі до переліку необхідно натиснути на кнопку «+». Для видалення їжі необхідно провести вправо та натиснути на кнопку «Видалити».

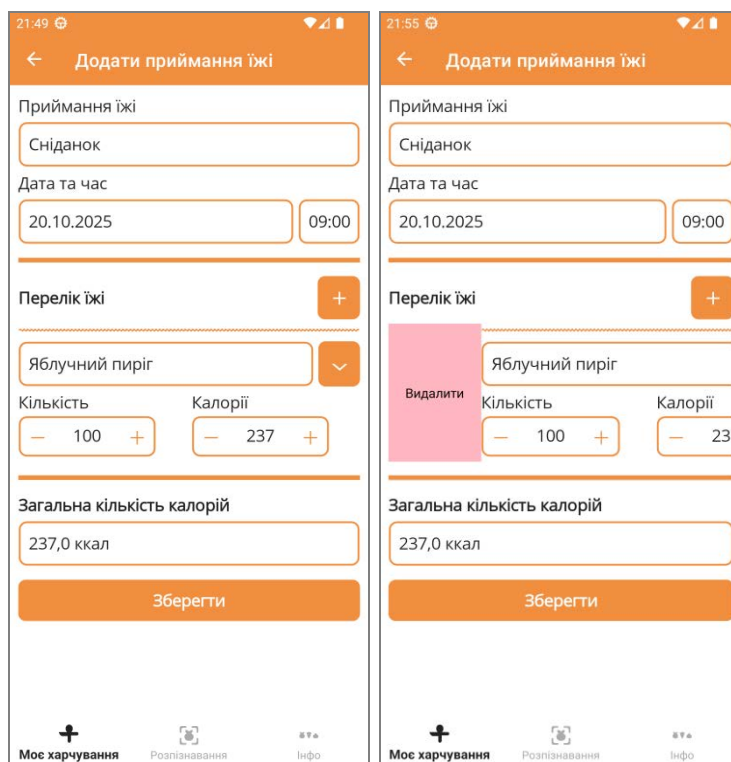


Рисунок 5.4 — Додавання приймання їжі

Після натискання кнопки «Зберегти» приймання їжі з'явиться в переліку із зазначенням назви, дати та часу, калорійності приймання їжі. Для видалення приймання зі списку необхідно утримувати значення приймання їжі та в контекстному меню натиснути на кнопку «Видалити» (рис. 5.5).

Для перегляду та редагування інформації про приймання їжі потрібно натиснути на нього в переліку приймання їжі. Відкриється сторінка перегляду (рис. 5.5), яка містить ті ж поля, що і при додаванні. Для збереження внесених змін необхідно натиснути на кнопку «Зберегти».

Після натискання на кнопку «Статистика» з'явиться сторінка «Статистика» (рис. 5.5). Вона містить поле для введення дати, кнопку

«Оновити» та графіки калорійності приймань їжі за обраний день і сумарної калорійності за відповідний тиждень.

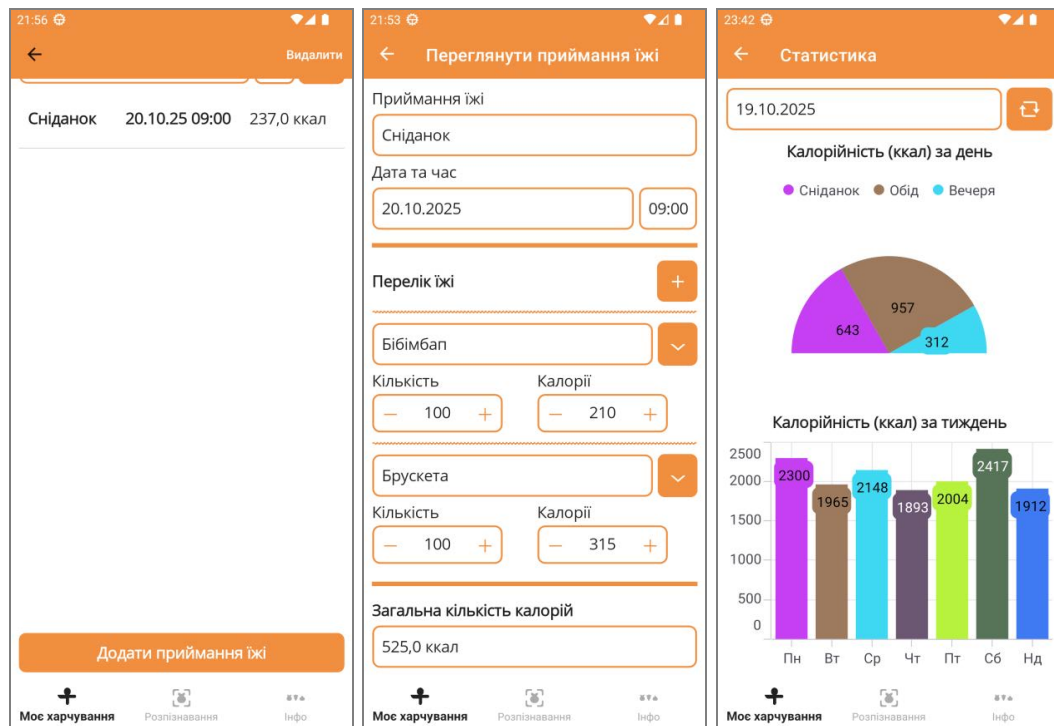


Рисунок 5.5 — Контекстне меню з кнопкою «Видалити», перегляд та редагування інформації про приймання їжі, виведення статистики

Друга вкладка «Розпізнавання» призначена для розпізнавання їжі за фотографіями та пошуку інформації про продукт в відкритій базі даних за його штрихкодом. Вона містить кнопку «Зображення їжі», текстове поле для штрихкоду, кнопку «Пошук», місце для зображення, текстове поле для результату розпізнавання або пошуку та кнопку «Додати». Після натискання кнопки «Зображення їжі» (рис. 5.6) відкриється застосунок камери для того, щоб зробити фотографію їжі. Потім зображення буде зашифровано та надіслано до серверної частини для розпізнавання. Отримана відповідь буде відображена в текстовому полі. Після введення штрихкоду та натискання кнопки «Пошук» (рис. 5.7) буде надіслано запит до відкритої бази даних Open Food Facts. Отримані зображення та результат будуть відображені у відповідних полях. Якщо результат відповідає їжі або продукту, то можна

натиснути на кнопку «Додати», після чого відкриється сторінка додавання приймання їжі з результатом розпізнавання або пошуку у полях «Назва» та «Калорії» переліку їжі.

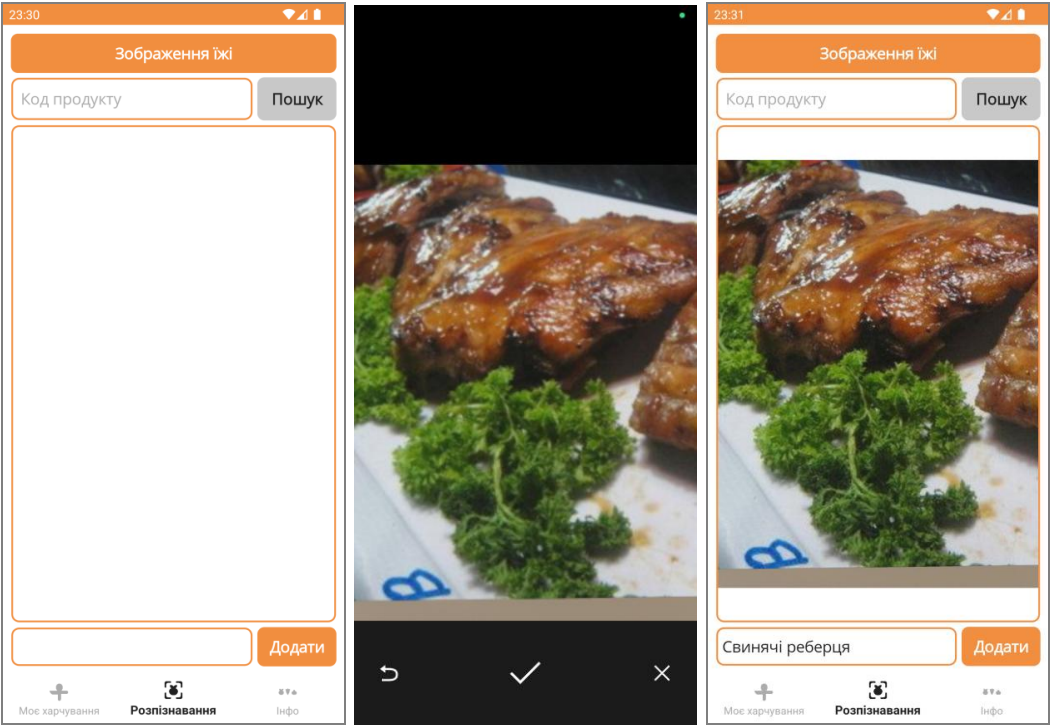


Рисунок 5.6 — Розпізнавання їжі за фотографією

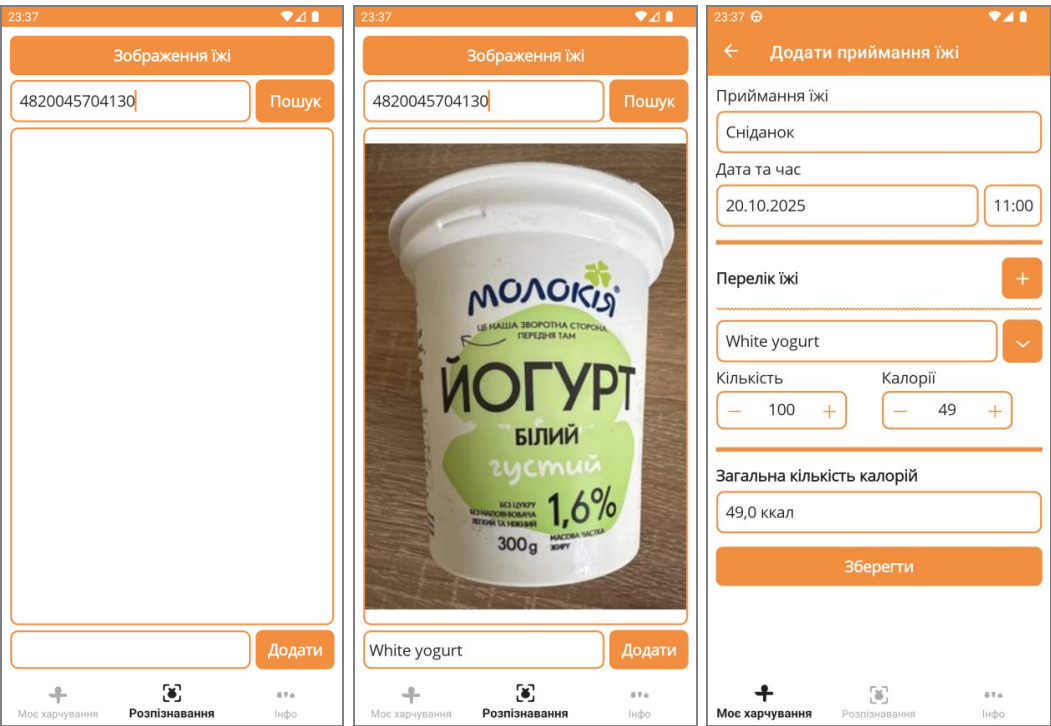


Рисунок 5.7 — Пошук продукту за штрихкодом

Третя вкладка «Інфо» (рис. 5.8) призначена для відображення інформації про користувача, застосунок і його розробника та взаємодії з розумними вагами Xiaomi Mi Body Composition Scale 2. Вона містить лічильники для зросту та віку користувача, список, що розкривається, для статі, кнопки «Під'єднатися до ваг», «Зберегти» та поле для отриманого результату. Для коректної взаємодії з розумними вагами необхідно ввімкнути Bluetooth та GPS і надати необхідні дозволи. Після натискання кнопки «Під'єднатися до ваг» застосунок намагається підключитися до ваг для отримання значень ваги та опору тіла користувача. Потім отримані значення використовуються для розрахунку інших показників тіла користувача, які буде відображено у відповідному полі. Для збереження інформації про користувача необхідно натиснути на кнопку «Зберегти».

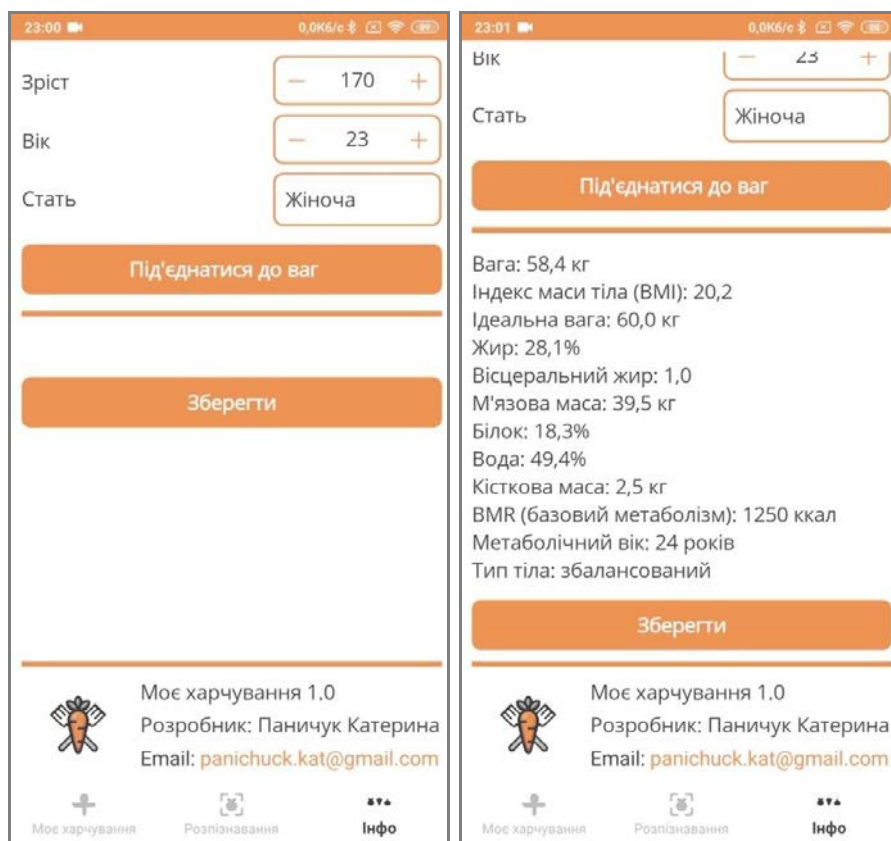


Рисунок 5.8 — Взаємодія з розумними вагами

5.4 Повідомлення оператору

Під час роботи з програмою оператор може отримати такі повідомлення.

«Помилка на стороні клієнта! Будь ласка, спробуйте пізніше або зверніться до служби підтримки» — сигналізує про помилку на стороні клієнта при надсиланні запиту до серверної частини (рис. 5.9). Рекомендується спробувати пізніше або звернутися до служби підтримки.



Рисунок 5.9 — Повідомлення про помилку на стороні клієнта

«Помилка на стороні сервера! Будь ласка, спробуйте пізніше або зверніться до служби підтримки» — вказує на помилку на сервері під час обробки запиту (рис. 5.10). Слід повторити спробу пізніше або зв'язатися зі службою підтримки.

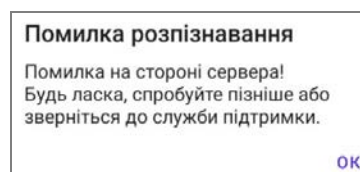


Рисунок 5.10 — Повідомлення про помилку на стороні сервера

«Помилка! Будь ласка, спробуйте пізніше або зверніться до служби підтримки» — виникає при проблемах із надсиланням запиту до серверної частини або його обробкою (рис. 5.11). Оператору слід повторити спробу або звернутися до підтримки.



Рисунок 5.11 — Повідомлення про помилку під час роботи з сервером

«Помилка! Будь ласка, спробуйте пізніше або зверніться до служби підтримки» — повідомлення про помилку при доступі до відкритої бази даних або під час обробки запиту (рис. 5.12). Рекомендується перевірити підключення до інтернету, коректність введеного штрихкоду, спробувати пізніше або звернутися до служби підтримки.

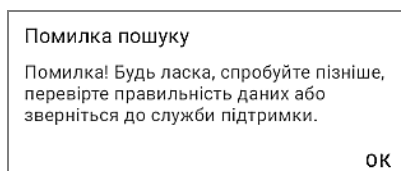


Рисунок 5.12 — Повідомлення про помилку під час роботи з відкритою БД

«Помилка! Будь ласка, спробуйте пізніше або зверніться до служби підтримки» — сигналізує про проблему при підключенні до розумних ваг Xiaomi Mi Body Composition Scale 2 (рис. 5.13). Необхідно перевірити, чи увімкнено Bluetooth і GPS, наявність відповідних дозволів на пристрої та працездатність ваг.

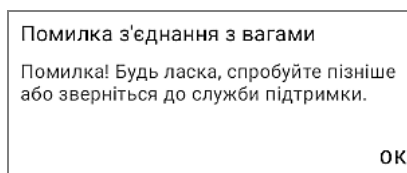


Рисунок 5.13 — Повідомлення про помилку з'єднання з розумними вагами

«Bluetooth вимкнено. Увімкніть його для виконання операції» — повідомляє про відсутність активного Bluetooth, через що операція

неможлива (рис. 5.14). Слід увімкнути Bluetooth або перевірити налаштування.

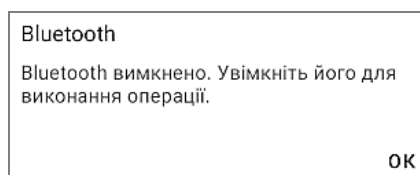


Рисунок 5.14 — Повідомлення про відсутність ввімкненого Bluetooth

«GPS вимкнено. Увімкніть його для виконання операції» — сигналізує про відключений GPS, що унеможлиблює виконання операції (рис. 5.15). Рекомендується активувати GPS або перевірити налаштування пристрою.

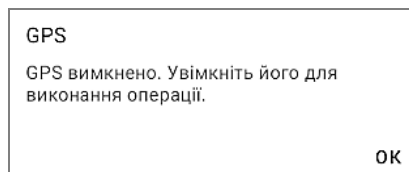


Рисунок 5.15 — Повідомлення про відсутність ввімкненого GPS

5.5 Висновки за розділом 5

У цьому розділі представлено призначення та основні функції розробленого програмного забезпечення, а також мінімальні системні вимоги для його стабільної роботи. Надано покрокову інструкцію для користувача мобільного застосунку, що включає додавання, редагування та видалення інформації про приймання їжі, перегляд статистичних даних, виконання розпізнавання їжі, пошук інформації про харчові продукти, взаємодію з розумними вагами та оновлення даних користувача. Також наведено перелік текстових повідомлень, які можуть з'являтися під час роботи з програмою. Інтерфейс застосунку розроблено таким чином, щоб він був інтуїтивно зрозумілим і не потребував додаткового навчання користувача.

6 ТЕСТУВАННЯ І РЕЗУЛЬТАТИ

6.1 Опис методики проведення тестування

Тестування буде виконано за наступним чином:

- запуск мобільного застосунку;
- додавання приймання їжі;
- редагування інформації про приймання їжі;
- виведення статистики;
- видалення приймання їжі;
- сортування переліку приймання їжі;
- створення фотографії їжі для подальшого шифрування, розшифрування та розпізнавання;
- пошук інформації про продукт за штрихкодом в відкритій базі даних Open Food Facts;
- взаємодія з розумними вагами Xiaomi Mi Body Composition Scale 2;
- оновлення інформації про користувача.

Під час тестування буде перевірено ключові функціональні можливості застосунку, зокрема правильність додавання, відображення та обробки даних тощо.

Тестування здійснюватиметься на емуляторі мобільного пристрою з ОС Android із такими характеристиками:

- модель: Google Pixel 5;
- версія ОС: Android 14.0 (API 34);
- процесор: x86_64;
- пам'ять: 1 Гб;
- діагональ екрану: 6.0 дюймів;
- роздільна здатність екрану: 1080x2340 пікселів, 440 dpi;
- доступ до мережі, фронтальної та задньої камер, Bluetooth та GPS;
- підтримка сенсорів.

6.2 Результати використання розробленого програмного забезпечення

Для тестування розробленої програми на емулятор мобільного пристрою з ОС Android було встановлено застосунок. Після його запуску була відкрита вкладка «Моє харчування» і було натиснуто кнопку «Додати приймання їжі». На відкритій сторінці «Додати приймання їжі» було заповнено наявні поля (рис. 6.1). Після натискання кнопки «Зберегти» приймання їжі було додано до БД та переліку приймання їжі на вкладці «Моє харчування». Під час повторного запуску застосунку приймання їжі відображається в списку.

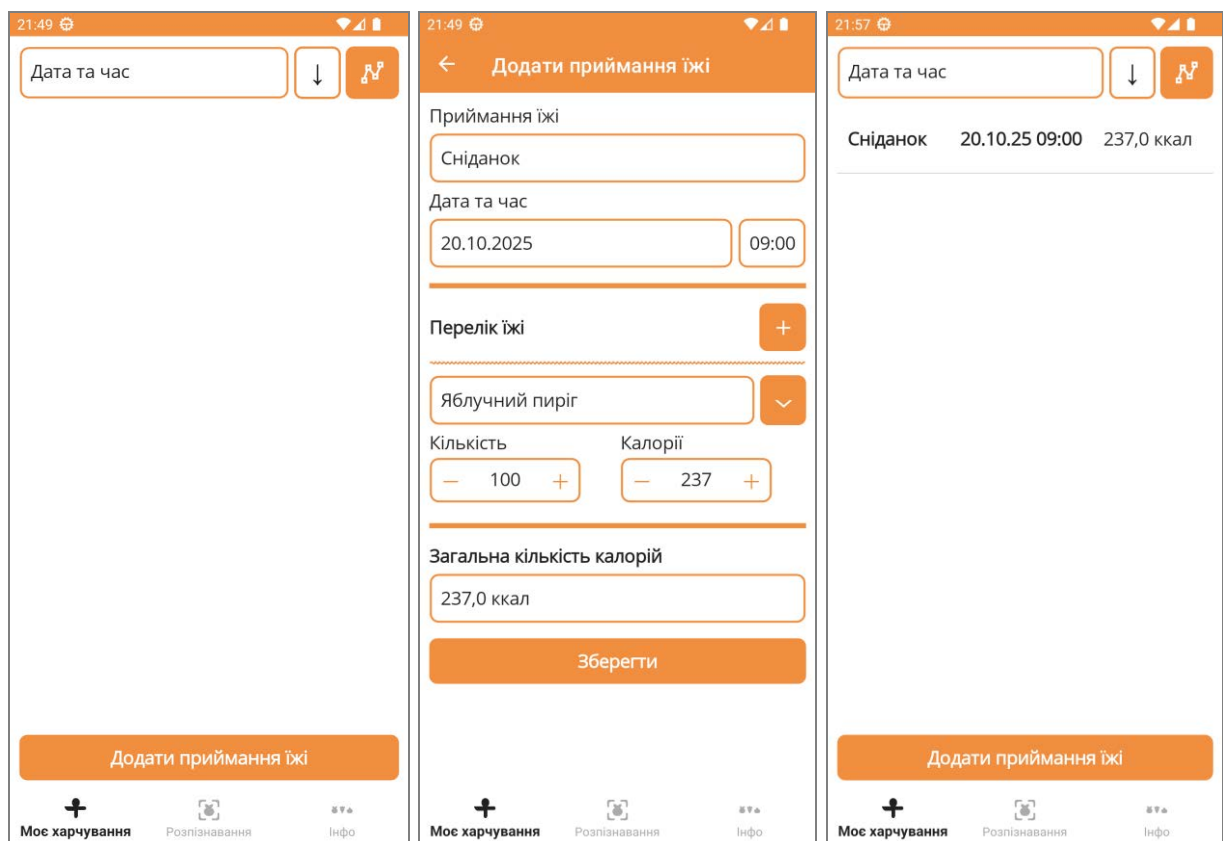


Рисунок 6.1 — Додавання приймання їжі до переліку

Після натискання на приймання їжі була відкрита сторінка «Переглянути приймання їжі», на якій було внесено зміни та натиснуто

кнопку «Зберегти» (рис. 6.2). Оновлена інформація була збережена в БД. Під час повторного запуску застосунку приймання їжі містить відредаговані дані.

Після натискання на кнопку «Статистика» була відкрита сторінка «Статистика», на якій було відображено графіки для обраного дня (рис. 6.2). Після змінення дати та натискання на кнопку «Оновити», графіки було змінено.

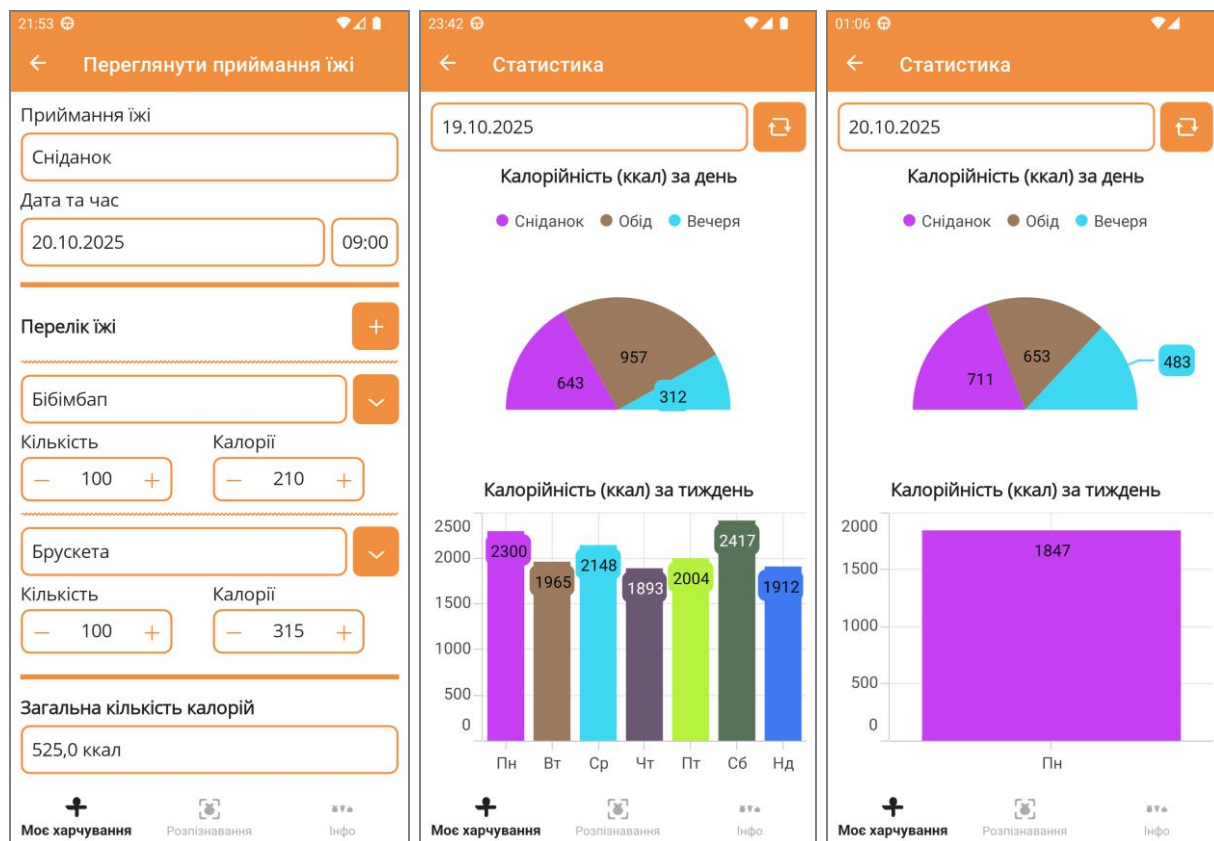


Рисунок 6.2 — Змінення інформації про приймання їжі та відображення статистики

Для видалення приймання їжі зі списку та БД було утримано значення приймання їжі та в контекстному меню натиснуто кнопку «Видалити» (рис. 6.3). Під час повторного запуску застосунку приймання їжі не було в переліку.

Над переліком є параметри сортування. Після їхнього змінення порядок розташування приймання їжі змінився (рис. 6.4).

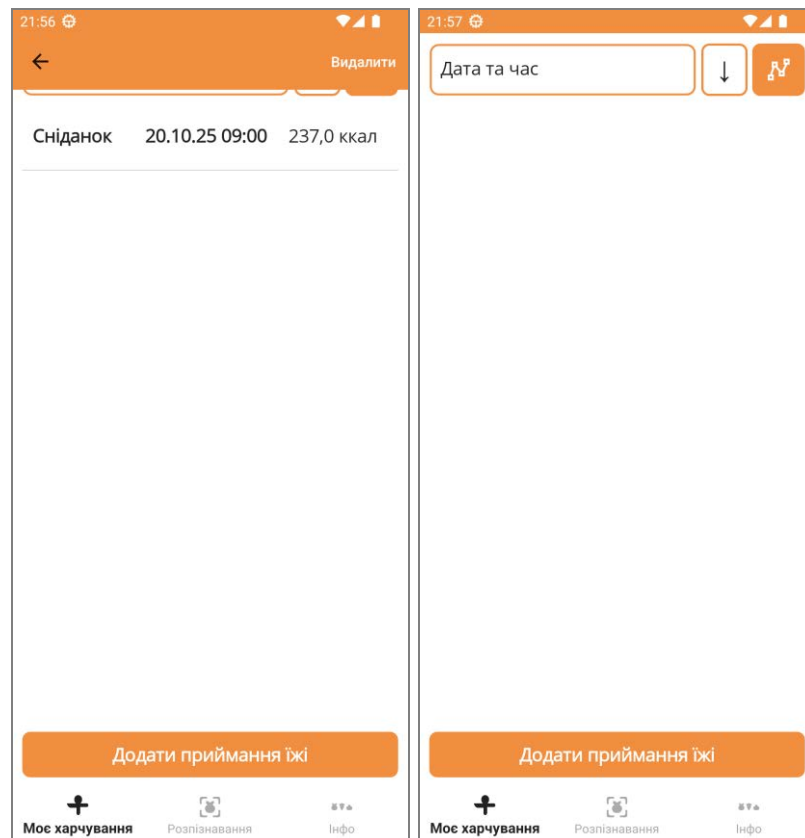


Рисунок 6.3 — Видалення приймання їжі

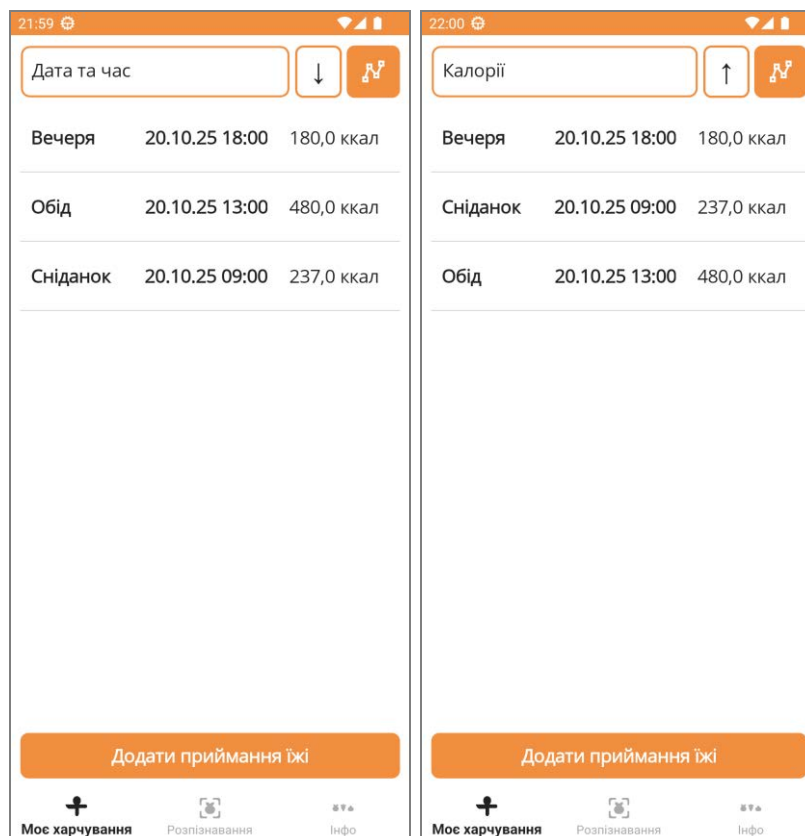


Рисунок 6.4 — Сортуння переліку приймання їжі

На другій вкладці «Розпізнавання» було натиснуто кнопку «Зображення їжі», зроблено фотографію їжі. Потім в текстовому полі з'явився результат розпізнавання з назвою їжі (рис. 6.5). Після натискання на кнопку «Додати» було відкрито сторінку «Додати приймання їжі» із заповненими полями «Назва» та «Калорії» переліку їжі.

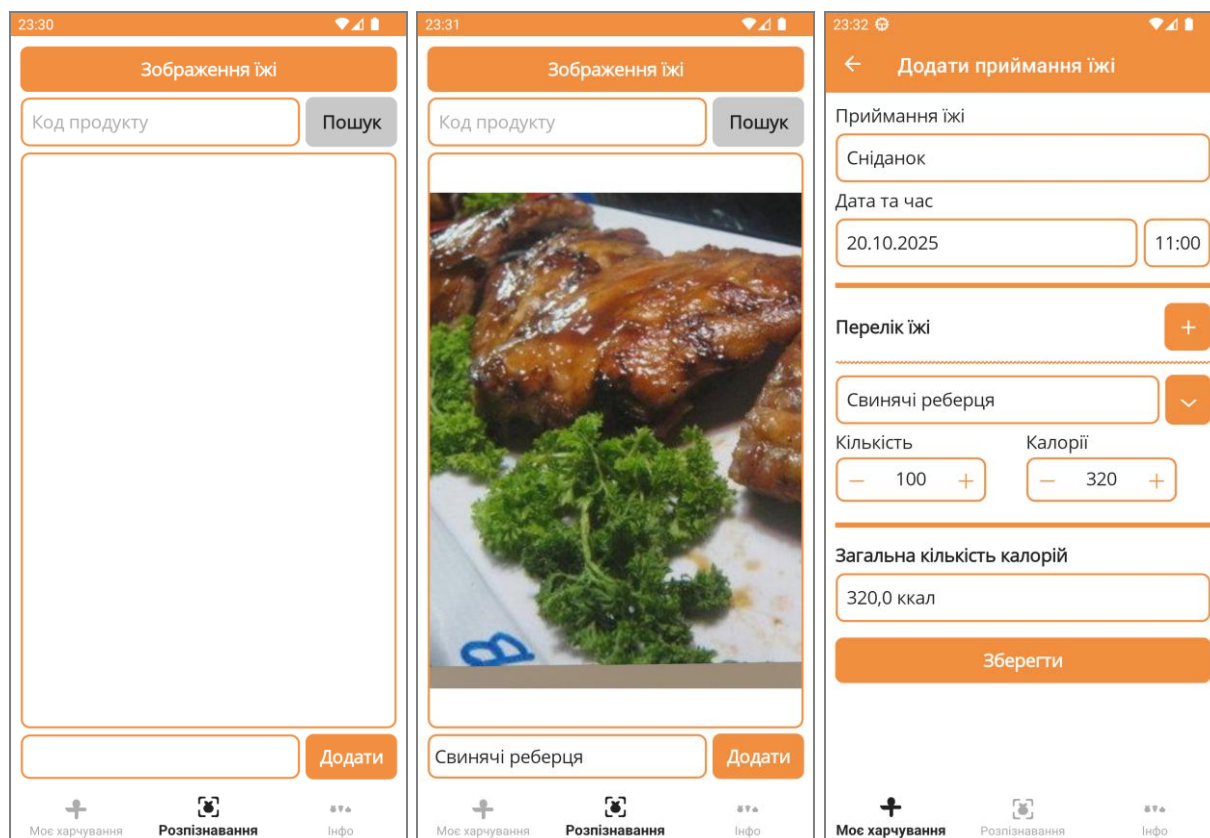


Рисунок 6.5 — Розпізнавання їжі за фотографією

На цій же вкладці було введено штрихкод продукту та натиснуто кнопку «Пошук». Потім з'явилися зображення та назва продукту в текстовому полі (рис. 6.6). Після натискання на кнопку «Додати» було відкрито сторінку «Додати приймання їжі» із заповненими полями «Назва» та «Калорії» переліку їжі.

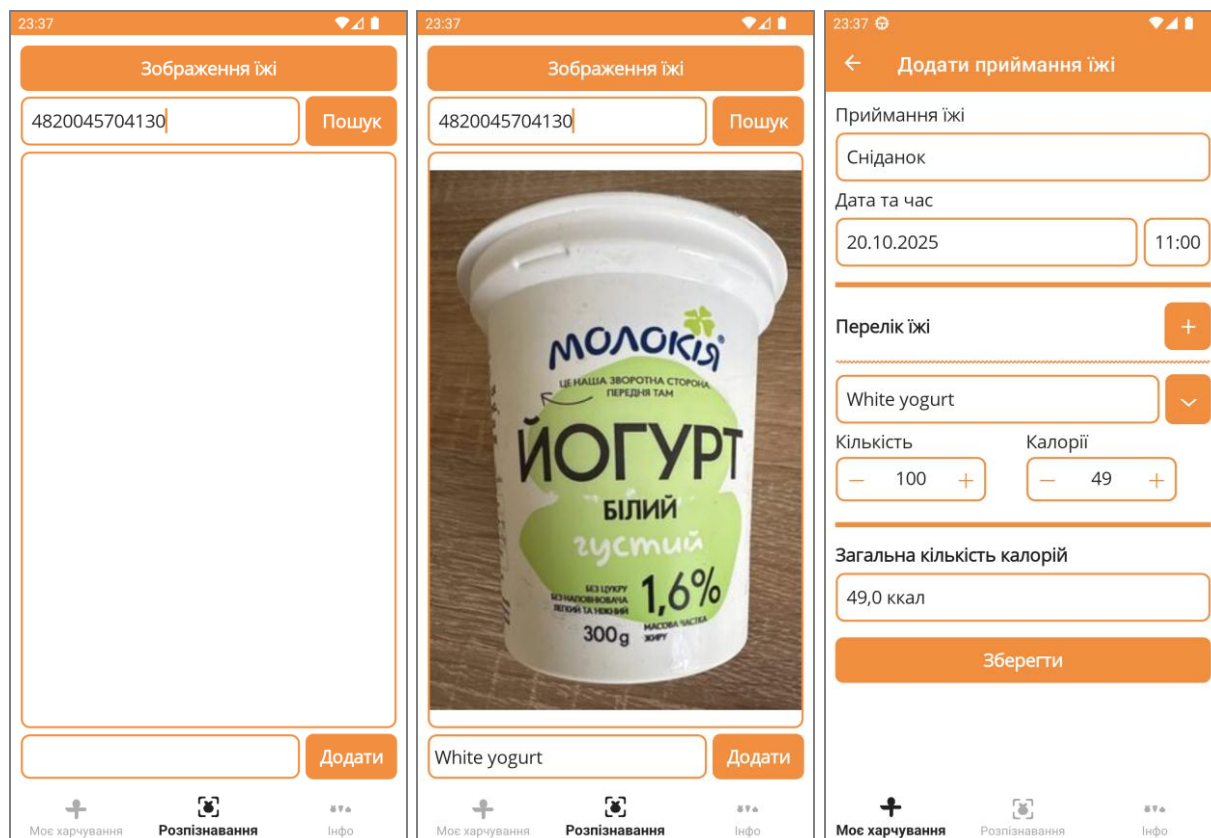


Рисунок 6.6 — Пошук продукту за штрихкодом

На третій вкладці «Інфо» було натиснуто кнопку «Під'єднатися до ваг» при ввімкнених Bluetooth і GPS та наданих дозволах. Потім в текстовому полі з'явився результат у вигляді розрахованих показників тіла користувача з отриманих ваги та опору тіла (рис. 6.7). Для збереження інформації про користувача було натиснуто кнопку «Зберегти». Під час повторного запуску застосунку оновлена інформація про користувача відображається у відповідних полях.

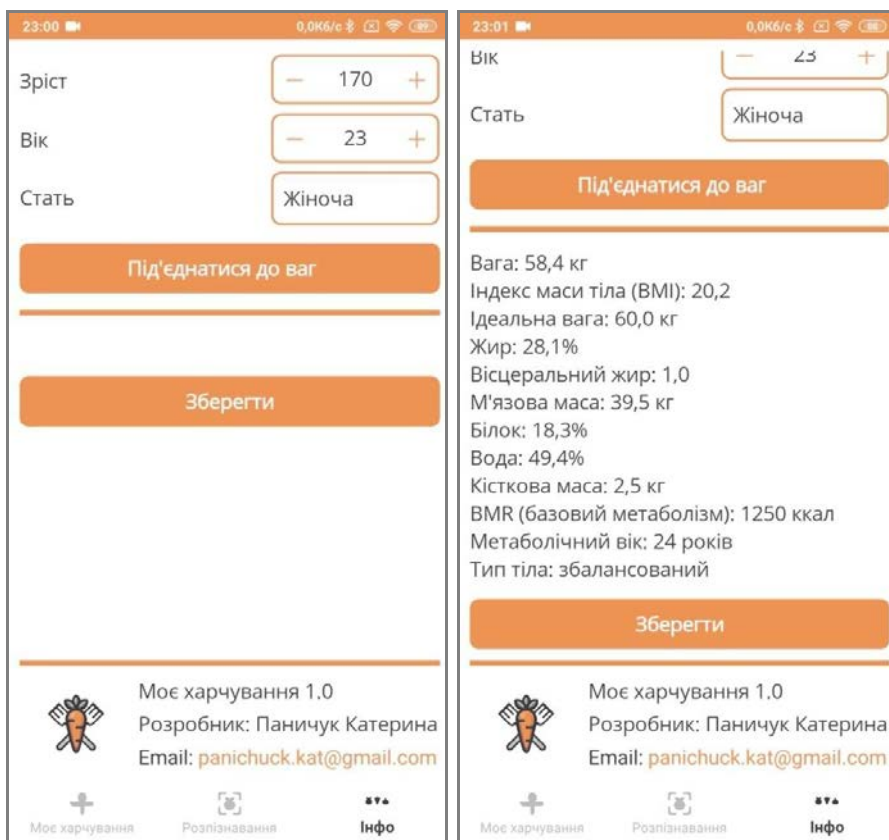


Рисунок 6.7 — Взаємодія з розумними вагами

Під час тестування було виконано низку сценаріїв, які охоплювали перевірку функціональності, стабільності роботи, взаємодії з серверною частиною, відкритою базою даних та розумними вагами, а також коректності збереження та обробки даних. Підсумки тестування наведено в таблиці 6.1.

Таблиця 6.1 — Результати тестування

Перевірка	Результат
Додавання приймання їжі	Пройдено
Змінення інформації про приймання їжі	Пройдено
Виведення статистики	Пройдено
Видалення приймання їжі	Пройдено
Сортування переліку приймання їжі	Пройдено
Надання можливості зробити фотографію їжі, її шифрування, розшифрування та подальше розпізнавання	Пройдено
Пошук інформації про продукт за штрихкодом в відкритій базі даних	Пройдено
Взаємодія з розумними вагами	Пройдено
Оновлення інформації про користувача	Пройдено

6.3 Висновки за розділом 6

У цьому розділі представлено сценарій тестування програмного забезпечення та характеристики пристрою, на якому воно проводилося. Було перевірено правильність роботи застосунку, його стабільність та відповідність заявленим вимогам.

ВИСНОВКИ

Під час виконання роботи було запропоновано модифіковану модель згорткової нейронної мережі MealNet для розпізнавання їжі, швидкість навчання якої серед 6 моделей є найкращою (741,4 с. для 10 класів, 3120,6 с. – 30, 6127,5 с. – 60, 9602,2 с. – 101).

У першому розділі було здійснено аналіз предметної області, обґрунтовано актуальність і доцільність створення програмного забезпечення. Розглянуто відомі методи виконання завдання обліку харчування, методи машинного навчання та мобільні застосунки, що забезпечують цей облік. Проведений аналіз дозволив виявити їхні переваги й недоліки та визначити методи та функціонал, які реалізовано у застосунку.

У другому розділі виконано аналіз вимог до системи, обрано інструментарій для розробки та створено діаграму прецедентів. Було визначено структуру майбутнього програмного забезпечення. Проведено порівняльний аналіз мов програмування: для мобільного застосунку обрано C# із фреймворком .NET MAUI, а для серверної частини — Python із використанням Flask, Keras, Tensorflow та PyCryptodome. Також здійснено порівняння середовищ розробки, в результаті якого Visual Studio визнано оптимальним варіантом, та систем управління базами даних, де перевага віддана SQLite.

У третьому розділі представлено вибраний шаблон проєктування та описано архітектуру створеного програмного забезпечення. Також наведено структури розроблених моделей для задачі розпізнавання їжі. Для реалізації системи розпізнавання їжі було обрано модель ResNet50V2, оскільки вона забезпечує оптимальний баланс між точністю для навчальної та тестової вибірок і демонструє стабільну здатність до узагальнення навіть за збільшення кількості класів. Ця модель показує високу точність тестування (71% для 10 класів і 48% для 101 класу) у порівнянні з іншими архітектурами, що свідчить про її ефективність для задач класифікації

зображень їжі. Було розроблено концептуальну модель бази даних із сутностями «Приймання їжі» та «Користувач», для яких визначено поля, типи та розміри даних. Також було описано реалізації взаємодій з відкритою базою даних Open Food Facts та розумними вагами Xiaomi Mi Body Composition Scale 2. Окрім цього, розглянуто методи шифрування даних AES, RSA та їх комбінацію. Аналіз показав, що гібридне шифрування є найбільш ефективним для застосунку, забезпечуючи безпечну передачу даних. Також представлено основні рішення щодо розробки графічного інтерфейсу користувача: описано сторінки мобільного застосунку, їх функціональне призначення, використані елементи та кольорову схему, а також вплив орієнтації пристрою на дизайн. Крім того, зазначено потенційні виняткові ситуації та повідомлення про помилки, які можуть виникати під час роботи з програмним забезпеченням.

Розроблене програмне забезпечення реалізує всі функції, визначені у другому розділі. Результати тестування підтвердили коректність роботи всього функціоналу.

У подальшому програмне забезпечення може бути вдосконалене шляхом додавання інформації про білки, жири та вуглеводи, групування приймання їжі за датою, використання добової норми калорій, покращення моделі розпізнавання їжі, впровадження додаткових механізмів безпеки та розширення списку розпізнаваної їжі тощо.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. The effectiveness of app-based mobile interventions on nutrition behaviours and nutrition-related health outcomes: A systematic review and meta-analysis [Text] / [K. Villinger, D.R. Wahl, H. Boeing et al.] // Obesity reviews : an official journal of the International Association for the Study of Obesity. – 2019. – Vol. 20, № 10. – P. 1465–1484.
2. Kirk, D. Food Tracking: The Good & The Bad [Electronic resource] / D. Kirk. – Access mode: <https://dankirk1995.medium.com/food-tracking-the-good-the-bad-9e3e9f844a06>.
3. Chesak, J. The 2025 Levels Guide to food tracking apps [Electronic resource] / J. Chesak. – Access mode: <https://www.levels.com/blog/food-tracking-apps>.
4. Lifesum: AI Calorie Tracker [Electronic resource]. – Access mode: <https://play.google.com/store/apps/details?id=com.sillens.shapeupclub>.
5. Calorie Counter by fatsecret [Electronic resource]. – Access mode: <https://play.google.com/store/apps/details?id=com.fatsecret.android>.
6. Calorie Counter • Planner • DietAI [Electronic resource]. – Access mode: <https://play.google.com/store/apps/details?id=dietaiplan.calorietracker.app>.
7. Archana, R. Deep learning models for digital image processing: a review [Text] / R. Archana, P.S.E. Jeevaraj // Artificial Intelligence Review. – 2024. – Vol. 57, № 11. – P. 1–33.
8. Lu, L. A Decision-Tree Classifier for Extracting Transparent Plastic-Mulched Landcover from Landsat-5 TM Images [Text] / L. Lu, L. Di, Y. Ye // IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing. – 2014. – Vol. 7, № 11. – P. 4548–4558.
9. Wu, J. Efficient HIK SVM Learning for Image Classification [Text] / J. Wu // IEEE Transactions on Image Processing. – 2012. – Vol. 21, № 10. – P. 4442–4453.

10. Taherkhani, A. Recognizing sorting algorithms with the C4.5 decision tree classifier [Text] / A. Taherkhani // IEEE 18th International Conference on Program Comprehension, Braga, Portugal. – 2010. – P. 72–75.
11. MDCT-based 3-D texture classification of emphysema and early smoking related lung pathologies [Text] / [Y. Xu, M. Sonka, G. McLennan et al.] // IEEE Transactions on Medical Imaging. – 2006. – Vol. 25, № 4. – P. 464–475.
12. Srensen, L. Quantitative Analysis of Pulmonary Emphysema Using Local Binary Patterns [Text] / L. Srensen, S.B. Shaker, M. de Bruijne // IEEE Transactions on Medical Imaging. – 2010. – Vol. 29, № 2. – P. 559–569.
13. Pineda-Jaramillo, J.D. A review of Machine Learning (ML) algorithms used for modeling travel mode choice [Text] / J.D. Pineda-Jaramillo // DYNA. – 2019. – Vol. 86, № 211. – P. 32–41.
14. Alnuaimi, A.F.A.H. An overview of machine learning classification techniques [Text] / A.F.A.H. Alnuaimi, T.H.K. Albaldawi // BIO Web of Conferences. – 2024. – Vol. 97, № 4. – P. 1–24.
15. Шевченко, Р. Рейтинг мов програмування 2025. TypeScript і Python — найпопулярніші, частки C# та Java зменшуються [Електрон. ресурс] / Р. Шевченко. – Режим доступу: <https://dou.ua/lenta/articles/language-rating-2025/>.
16. Hamaamin, R.A. Java Programming Language: Time Permanence Comparison with Other Languages: A Review [Text] / R.A. Hamaamin, O.M.A. Ali, S.W. Kareem // ITM Web of Conferences. – 2024. – Vol. 64. – P. 01012.
17. Fojtik, R. Swift a New Programming Language for Development and Education / R. Fojtik // Digital Science 2019. – Cham, 2019. – P. 284–295.
18. Mateus, B.G. On the adoption, usage and evolution of Kotlin features in Android development [Text] / B.G. Mateus, M. Martinez // ESEM '20: ACM / IEEE International Symposium on Empirical Software Engineering and Measurement, Bari, Italy. – New York : ACM, 2020. – P. 1–12.

19. Ali, S. A Pragmatic Comparison of Four Different Programming Languages [Text] / S. Ali, S. Qayyum // ScienceOpen Preprints. – 2021. – P. 1–13.
20. Comparison of Programming Languages: Review [Text] / [K.P.N. Reddy, Y. Geyavalli, D. Sujani et al.] // International Journal of Computer Science and Communication. – 2018. – Vol. 9, № 2. – P. 113–122.
21. Evolution & Trends of Programming Language [Text] / [P. Chaudhari, M. Rahate, M. Pokale et al.] // International Research Journal of Engineering and Technology. – 2025. – Vol. 12, № 3. – P. 397–402.
22. Kovács, M. Comparative Analysis of Native and Cross-Platform iOS Application Development [Text] / M. Kovács, Z.C. Johanyák // Műszaki Tudományos Közlemények. – 2021. – Vol. 15, № 1. – P. 61–64.
23. Singh, A. Android Application Development using Android Studio and PHP Framework [Text] / A. Singh, S. Sharma, S. Singh // International Journal of Computer Applications. – 2016. – P. 5–8.
24. Sotnik, S. Modern Integrated Software Development Environments [Text] / S. Sotnik, V. Lyashenko, T. Schakurova // International Journal of Academic and Applied Research. – 2021. – Vol. 5, № 10. – P. 157–161.
25. Salunke, S.V. A Performance Benchmark for the PostgreSQL and MySQL Databases [Text] / S.V. Salunke, A. Ouda // Future Internet. – 2024. – Vol. 16, № 10. – P. 382.
26. MongoDB Vs PostgreSQL: A comparative study on performance aspects [Text] / [A. Makris, K. Tserpes, G. Spiliopoulos et al.] // Geoinformatica. – 2021. – Vol. 25. – P. 243–268.
27. SQLite: Past, Present, and Future [Text] / [K.P. Gaffney, M. Prammer, L. Brasfield et al.] // Proceedings of the VLDB Endowment. – 2022. – Vol. 15, № 12. – P. 3535–3547.
28. Mohamed, V. A study on Model-View-View Model (MVVM) Design Pattern [Text] / V. Mohamed, A.S. Shaffi // International Journal of Emerging Technology and Advanced Engineering. – 2016. – Vol. 6, № 7. – P. 264–269.

29. Sruthi, M.S. Automatic Voting System Using Convolutional Neural Network [Text] / M.S. Sruthi, K. Shanjai // Journal of Physics: Conference Series. – 2021. – Vol. 1916, № 1. – P. 012074.

30. Datta, L. A Survey on Activation Functions and their relation with Xavier and He Normal Initialization [Text] / L. Datta // ArXiv. – 2020. – P. 2004.06632.

31. Kingma, D.P. Adam: A Method for Stochastic Optimization [Text] / D.P. Kingma, J.L. Ba // International Conference on Learning Representations, Banff, 14–16 April 2014. – Ithaca, NY : ArXiv, 2015. – P. 1–15.

32. Calcification Detection of Coronary Artery Disease in Intravascular Ultrasound Image: Deep Feature Learning Approach [Text] / [H. Sofian, J.C.M. Than, S. Mohammad et al.] // International Journal of Integrated Engineering. – 2018. – Vol. 10, № 7. – P. 43–57.

33. Chollet, F. Xception: Deep Learning with Depthwise Separable Convolutions [Text] / F. Chollet // 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, 21–26 July 2017. – IEEE, 2017. – P. 1800–1807.

34. MobileNetV2: Inverted Residuals and Linear Bottlenecks [Text] / [M. Sandler, A. Howard, M. Zhu et al.] // 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, 18–23 June 2018. – IEEE, 2018. – P. 4510–4520.

35. Fine-Tuned DenseNet-169 for Breast Cancer Metastasis Prediction Using FastAI and 1-Cycle Policy [Text] / [A. Vulli, P.N. Srinivasu, M.S.K. Sashank et al.] // Sensors (Basel). – 2022. – Vol. 22, № 8. – P. 2988.

36. Open Food Facts. Get information for a specific product by barcode [Electronic resource] – Access mode: <https://openfoodfacts.github.io/openfoodfacts-server/api/ref-v2/#get-/api/v2/product/-barcode->.

37. Mi Body Composition Scale [Electronic resource] – Access mode: <https://github.com/wiecosystem/Bluetooth/blob/master/doc/devices/huami.health.scale2.md>.

38. Abdulah, H.S. An Enhancement Method Based on Modifying CFB Mode for Key Generation in AES Algorithm [Text] / H.S. Abdulah, M.A.H. Al-Rawi, D.N. Hammod // Engineering and Technology Journal. – 2016. – Vol. 34, № 6B. – P. 759–768.

39. Secure Sensitive Data Sharing Using RSA and ElGamal Cryptographic Algorithms with Hash Functions [Text] / [E.A. Adeniyi, P.B. Falola, M.S. Maashi et al.] // Information. – 2022. – Vol. 13, № 10. – P. 442.

40. Chaloop, S.G. Enhancing Hybrid Security Approach Using AES and RSA Algorithms [Electronic resource] / S.G. Chaloop, M.Z. Abdullah // Journal of Engineering and Sustainable Development. – 2021. – Vol. 25, № 4. – P. 58–66.

ДОДАТОК А
Технічне завдання

Вступ

Сфера застосування мобільного застосунку «Моє харчування» охоплює індивідуальне використання для моніторингу харчування, а також професійну діяльність у сфері здоров'я та дієтології. Вона може бути корисною для людей, які прагнуть контролювати свою вагу, слідкують за калоріями, дотримуються дієт або мають специфічні потреби в харчуванні. Програма також може використовуватися фахівцями, такими як дієтологи чи лікарі, для аналізу раціону своїх пацієнтів та надання рекомендацій щодо коригування харчових звичок.

А.1 Підстави для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Дослідження та програмна реалізація мобільного застосунку для обліку харчування», затверджене наказом Національного університету «Запорізька політехніка» № 447 від 30 вересня 2025 р.

А.2 Призначення розробки

Призначенням програми є надання зручного інструменту для обліку харчування, дозволяючи користувачам зробити більш обґрунтовані рішення щодо свого раціону та підтримувати здоровий баланс калорій у повсякденному житті.

А.2.1 Функціональне призначення

Програма має надавати можливості додавати, видаляти, сортувати приймання їжі, редагувати інформацію про них та користувача, виводити статистику калорійності приймань їжі, розпізнавати деяку їжу за

фотографіями, отримувати дані про продукт за його штрихкодом, взаємодіяти з розумними вагами для розрахунку показників.

A.2.2 Експлуатаційне призначення

Застосунок призначений для використання на мобільних пристроях, які відповідають мінімальним апаратним та програмним вимогам, необхідним для його стабільної та коректної роботи.

A.3 Вимоги до програми або програмного продукту

A.3.1 Вимоги до функціональних характеристик

A.3.1.1 Користувач

Користувачем програми є широкий спектр осіб, які зацікавлені у контролі за своїм харчуванням. Це можуть бути люди, що бажають підтримувати здоровий спосіб життя, дотримуються певних дієт або намагаються скинути або набрати вагу. Програма також орієнтована на тих, хто прагне покращити свої харчові звички, а також на професіоналів — дієтологів і медичних працівників, які використовують програму для моніторингу раціону пацієнтів.

A.3.1.2 Вимоги до складу виконуваних функцій

Мобільний застосунок має реалізовувати виконання таких функцій:

- додавання, редагування, видалення інформації про приймання їжі;
- сортування переліку приймання їжі;
- виведення статистики калорійності приймань їжі;
- надання можливості зробити фотографію їжі, її шифрування, розшифрування та подальше розпізнавання;

- отримання даних про продукт з відкритої бази даних Open Food Facts за допомогою штрихкоду;
- редагування інформації про користувача;
- взаємодія з розумними вагами Xiaomi Mi Body Composition Scale 2 для отримання значень ваги та опору тіла, розрахунку показників тіла за даними про користувача.

A.3.1.3 Вимоги до організації початкових даних

Вхідними даними для програмного забезпечення є інформація, що вже міститься в базі даних на момент запуску, а також відомості, які користувач вводить під час роботи із застосунком.

Структура інформації про приймання їжі: назва, дата та час, перелік їжі (назва, кількість, калорійність), загальна кількість калорій. Структура інформації про користувача: зріст, вік, стать, склад тіла.

Перелік їжі містить об'єкти, які включають в себе назву продукту, кількість і калорійність. Назви та калорійність їжі можуть бути отримані з каталогу (файлу JSON), що містить інформацію про їжу, результатів розпізнавання або пошуку за штрихкодом, що дозволяє автоматично поповнювати базу даних актуальними даними без необхідності вручну вводити харчові характеристики їжі.

Склад тіла містить набір детальних показників тіла, розрахованих з отриманих із розумних ваг даних.

A.3.1.4 Вимоги до організації вихідних даних

Вихідні дані — це результати, що відображаються на екрані після оброблення інформації алгоритмами та представлення їх засобами інтерфейсу. До них, зокрема, належать підрахована сумарна калорійність, яка

визначається на основі внесених приймань їжі, а також параметри тіла користувача.

A.3.1.5 Вимоги до тимчасових характеристик

Для програми не встановлюються жорсткі вимоги щодо тимчасових характеристик.

A.3.2 Вимоги до надійності

Надійність програмного забезпечення повинна забезпечувати безвідмовну роботу не менше ніж у 99% часу.

A.3.2.1 Вимоги до забезпечення надійного функціонування програми

Стабільність функціонування системи гарантується виконанням низки організаційно-технічних заходів, серед яких:

- забезпечення безперебійного живлення обладнання;
- застосування ліцензійних програмних продуктів;
- регулярний контроль відповідно до стандартів інформаційної безпеки та періодична антивірусна перевірка.

A.3.2.2 Збійні ситуації

Програма повинна коректно опрацьовувати можливі нестандартні ситуації, зокрема:

- втрату підключення до бази даних;
- спробу збереження некоректної інформації;
- неможливість прочитати дані з бази даних;

- відсутність значень у полях, які є обов’язковими;
- помилки, що виникають під час взаємодії з серверною частиною, відкритою базою даних або розумними вагами.

У разі виникнення помилкових ситуацій застосунок повинен видавати зрозумілі повідомлення з рекомендаціями щодо подальших дій користувача.

A.3.2.3 Час відновлення після відмови

Вимоги щодо часу відновлення працездатності після можливих відмов не встановлюються.

A.3.3 Умови експлуатації

A.3.3.1 Кліматичні умови експлуатації

Експлуатаційні кліматичні умови для забезпечення зазначених характеристик повинні відповідати стандартам, визначеним для технічних засобів, на яких запускається програмне забезпечення.

A.3.3.2 Вимоги до видів обслуговування

Додаткового технічного обслуговування програмний продукт не потребує.

A.3.3.3 Вимоги до чисельності та кваліфікації персоналу

Не висуваються вимоги до кількості чи рівня кваліфікації персоналу, який працюватиме з програмою.

А.3.4 Вимоги до складу та параметрів технічних засобів

Мінімальні вимоги до апаратного та програмного забезпечення для стабільної роботи мобільного застосунку визначені наступним чином:

- операційна система: Android 5.0 або новіша, iOS 11 або новіша;
- процесор: двоядерний з частотою 1,5 ГГц або вище;
- вільний простір на диску: щонайменше 250 Мб;
- оперативна пам'ять: не менше 2 Гб;
- підключення до інтернету;
- наявність камери, модуля Bluetooth та GPS.

Для нормальної роботи серверної частини застосунку мінімальні вимоги такі:

- програмне забезпечення: Python 3.9, Flask 3.0.3, Tensorflow 2.15.0, Keras 2.15.0, NumPy 1.26.4, Pillow 9.4.0, Matplotlib 3.9.2, PyCryptodome 3.21.0;
- процесор: двоядерний із частотою не менше 2 ГГц;
- вільний простір на диску: не менше 5 Гб;
- оперативна пам'ять: щонайменше 4 Гб;
- підключення до інтернету.

А.3.5 Вимоги до інформаційної та програмної сумісності

Розробка клієнтської частини застосунку має здійснюватися мовою C# у середовищі Microsoft Visual Studio. Під час проектування архітектури слід використовувати шаблон проектування Model-View-ViewModel, а для побудови інтерфейсу користувача — фреймворк .NET MAUI.

Оскільки застосунок призначений для роботи на пристроях під керуванням Android та iOS, коректне функціонування залежить від стабільної роботи відповідних мобільних операційних систем.

Серверна частина повинна бути реалізована мовою Python з використанням таких бібліотек, як Flask, Keras, TensorFlow, NumPy, Pillow,

Matplotlib та PyCryptodome. Для її розроблення також рекомендовано застосовувати Microsoft Visual Studio.

Сервер може працювати на будь-якій операційній системі, що підтримує Python та перелічені бібліотеки, забезпечуючи універсальність розгортання.

A.3.6 Вимоги до маркування та пакування

Мобільний застосунок постачається у форматі APK (Android Package Kit) для пристроїв Android та у форматі IPA (iOS App Store Package) для системи iOS. Спеціальні вимоги щодо маркування відсутні.

A.3.7 Вимоги до транспортування та збереження

Не висуваються вимоги до умов транспортування чи зберігання програмного продукту.

A.3.8 Спеціальні вимоги

Взаємодія користувача із застосунком забезпечується через графічний інтерфейс, приклади якого наведено на рисунках A.1-A.2.

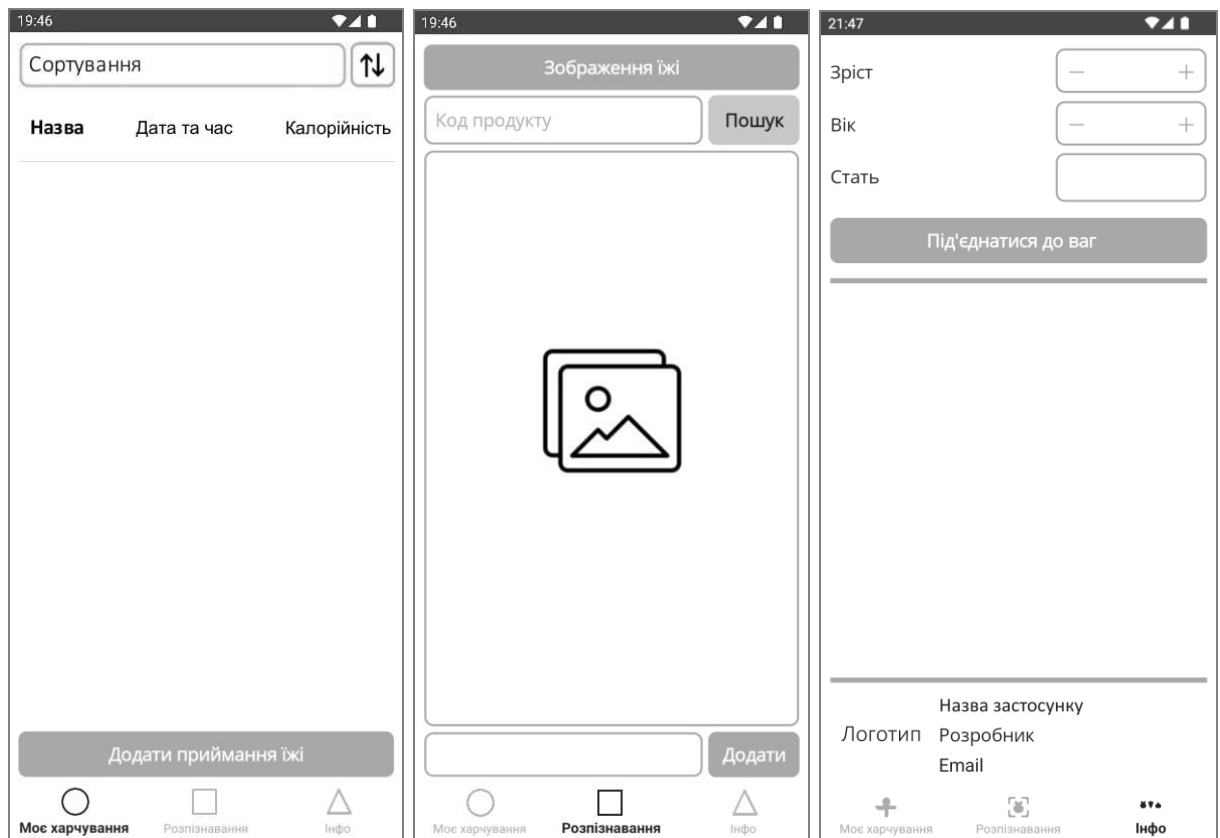


Рисунок А.1 — Сторінки з переліком, розпізнаванням та інформацією

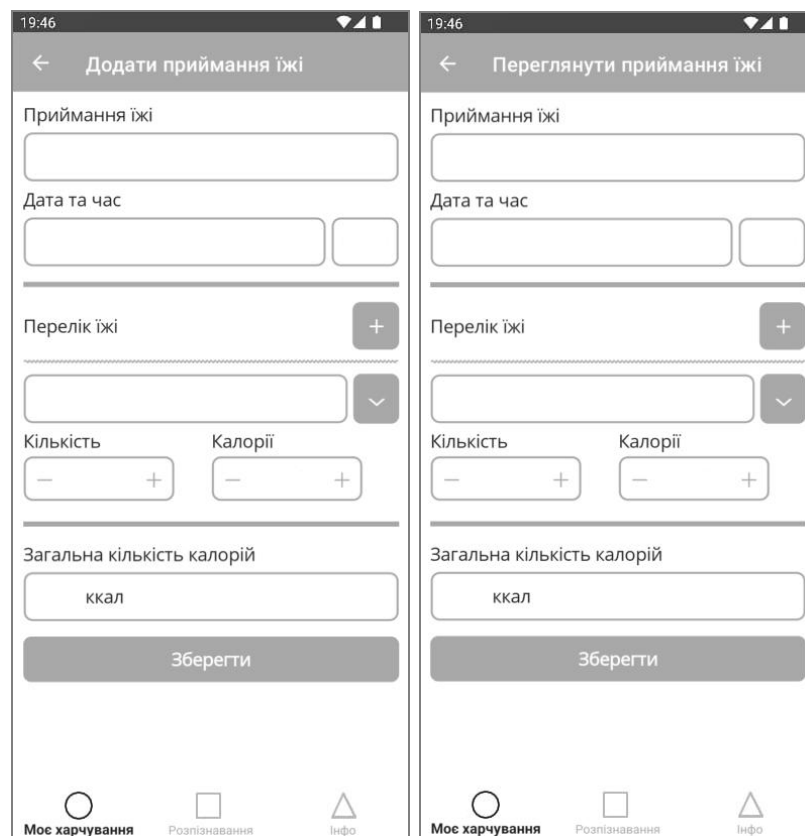


Рисунок А.2 — Сторінки додавання та перегляду приймання їжі

A.4 Вимоги до програмної документації

До комплекту програмної документації мають входити:

- технічне завдання;
- опис програми;
- керівництво оператора;
- керівництво програміста;
- програма і методика випробувань.

A.5 Техніко-економічні показники

Економічна доцільність упровадження програми досягається завдяки істотному зменшенню часу та трудових витрат, необхідних для контролю та аналізу харчування. Застосунок автоматизує облік калорій, усуваючи потребу у ручному внесенні даних і самотійному підрахунку калорійності продуктів, що зазвичай потребує значних зусиль. Для спеціалістів — зокрема дієтологів і медичних працівників — система пришвидшує аналіз харчових звичок пацієнтів і спрощує формування індивідуальних рекомендацій, що підвищує якість та оперативність надання професійних послуг.

A.6 Стадії та етапи розробки

Розроблення програмного забезпечення передбачає три основні стадії:

- підготовка технічного завдання;
- виконання технічного та робочого проєктування;
- етап впровадження.

Стадія «Технічне завдання» охоплює підготовку, узгодження та затвердження документа, що визначає вимоги до майбутньої системи.

На етапі «Технічний і робочий проєкти» виконуються такі роботи:

- створення програмного забезпечення;

- оформлення необхідної програмної документації;
- проведення тестування.

На стадії «Впровадження» здійснюється підготовка та передача програми замовнику.

Зміст робіт за етапами. Етап розроблення технічного завдання включає:

- формулювання цілей та постановки задачі;
- уточнення вимог до апаратного забезпечення;
- визначення характеристик і функціональних вимог до програмного продукту;
- вибір стадій та етапів створення програмного забезпечення, а також термінів виконання;
- узгодження та затвердження технічного завдання.

Етап створення програми передбачає її програмування (кодування) та налагодження.

Під час підготовки програмної документації формуються всі необхідні документи, що супроводжують програмний продукт.

Етап тестування охоплює:

- розробку, погодження та затвердження методики проведення випробувань;
- виконання приймально-здавальних тестів;
- оформлення звітної документації за результатами перевірок.

На етапі впровадження здійснюється передача готової програми та супровідної документації для подальшої експлуатації.

Розробник повинен створити мобільний застосунок «Моє харчування», а також передати вихідні коди та повний комплект документації не пізніше 08.12.2025.

A.7 Порядок контролю та приймання

Приймально-здавальні випробування програмного забезпечення мають виконуватися відповідно до «Програми і методики випробувань», яку розробляє виконавець та погоджує замовник.

Процес проведення цих випробувань фіксується в протоколі, який ведеться представниками обох сторін.

На основі складеного протоколу виконавець разом із замовником оформлюють та підписують акт введення програми в експлуатацію.

ДОДАТОК Б
Опис програми

Б.1 Загальні відомості

Мобільний застосунок має назву «Моє харчування». Призначенням програми є надання зручного інструменту для обліку харчування, дозволяючи користувачам зробити більш обґрунтовані рішення щодо свого раціону та підтримувати здоровий баланс калорій у повсякденному житті.

Користувачем програми є широкий спектр осіб, які зацікавлені у контролі за своїм харчуванням. Це можуть бути люди, що бажають підтримувати здоровий спосіб життя, дотримуються певних дієт або намагаються скинути або набрати вагу. Програма також орієнтована на тих, хто прагне покращити свої харчові звички, а також на професіоналів — дієтологів і медичних працівників, які використовують програму для моніторингу раціону пацієнтів.

Б.2 Функціональне призначення

Мобільний застосунок виконує наступні функції:

- додавання, редагування, видалення інформації про приймання їжі;
- сортування переліку приймання їжі;
- виведення статистики калорійності приймань їжі;
- надання можливості зробити фотографію їжі, її шифрування, розшифрування та подальше розпізнавання;
- отримання даних про продукт з відкритої бази даних Open Food Facts за допомогою штрихкоду;
- редагування інформації про користувача;
- взаємодія з розумними вагами Xiaomi Mi Body Composition Scale 2 для отримання значень ваги та опору тіла, розрахунку показників тіла за даними про користувача.

Б.3 Опис логічної структури програми

Програмне забезпечення містить два основні компоненти: клієнтську частину у вигляді мобільного застосунку та серверний модуль. Узагальнену структурну схему системи подано на рисунку Б.1.

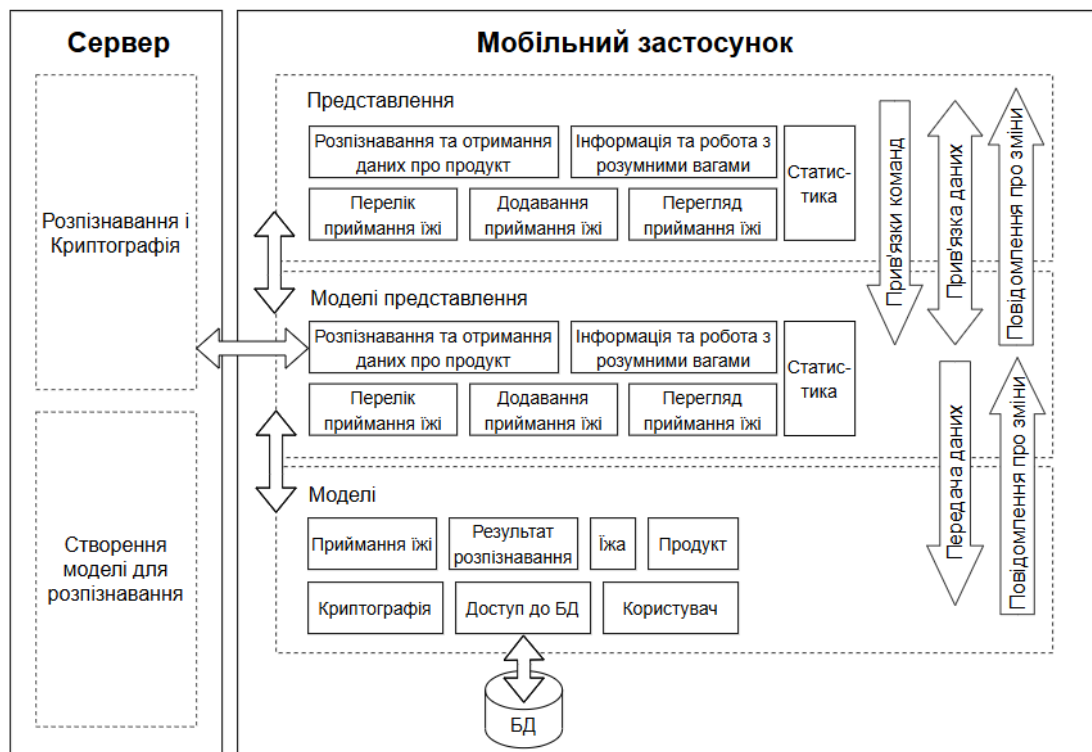


Рисунок Б.1 — Структурна схема програмного забезпечення

Мобільний застосунок складається з трьох основних компонентів: моделі, представлення та моделі представлення. Модулі частини з моделями наведено в таблиці Б.1.

Таблиця Б.1 — Опис модулів частини з моделями

Модуль	Структура	Функції
1	2	3
Доступ до БД	class DBConstants, class CommonData, class MealsDatabase	Зчитування інформації з БД та файлу JSON; Запис та оновлення даних у БД; Видалення даних з БД.
Приймання їжі	class Meal	Зберігання даних про приймання їжі; Редагування даних.

Продовження таблиці Б.1

1	2	3
Їжа	class MealFood	Зберігання даних про їжу; Редагування даних.
Продукт	class ProductRoot, class Product, class Nutriments	Зберігання даних про продукт.
Користувач	class UserInfo	Зберігання даних про користувача; Редагування даних.
Результат розпізнавання	class Prediction	Зберігання даних про результат розпізнавання.
Криптографія	class Cryptography	Шифрування зображень.

Модулі частини з моделями представлення наведено в таблиці Б.2.

Таблиця Б.2 — Опис модулів частини з моделями представлення

Модуль	Структура	Функції
Перелік приймання їжі	class AllMeals-VM	Отримання переліку приймання їжі з БД; Застосування сортування за обраними параметрами; Видалення обраного запису про приймання їжі з БД.
Статистика	class ChartsVM	Отримання дати для відображення даних; Визначення даних калорійності приймань їжі за обраний день та сумарної калорійності за днями відповідного тижня.
Додавання приймання їжі	class MealVM	Приймання даних від користувача; Створення нового запису в БД.
Перегляд приймання їжі		Отримання обраного приймання їжі; Оновлення інформації в БД після редагування.
Розпізнавання та отримання даних про продукт	class Scanner-VM	Отримання ключів шифрування від сервера; Отримання зображення їжі за допомогою камери, його шифрування та відправка на сервер; Отримання результату розпізнавання з сервера; Отримання даних про продукт з відкритої бази даних Open Food Facts за допомогою штрихкоду.
Інформація та робота з розумними вагами	class InfoVM	Отримання інформації про користувача; Взаємодія з розумними вагами Xiaomi Mi Body Composition Scale 2 для отримання значень ваги та опору тіла, розрахунку показників тіла за даними про користувача; Збереження інформації в БД.

Модулі частини з представленнями наведено в таблиці Б.3.

Таблиця Б.3 — Опис модулів частини з представленнями

Модуль	Структура	Функції
Перелік приймання їжі	class AllMealsV	Відображення списку приймання їжі з урахуванням сортування; Встановлення параметрів сортування; Виділення конкретного приймання їжі та можливість його видалення або редагування; Перехід до додавання нового приймання їжі.
Статистика	class ChartsV	Введення дати для відображення (оновлення) даних; Відображення статистики калорійності приймань їжі за обраний день та сумарної калорійності за днями відповідного тижня.
Додавання приймання їжі	class MealV	Введення даних про приймання їжі; Збереження нової інформації (додавання до списку приймання їжі).
Перегляд приймання їжі		Відображення деталей обраного приймання їжі; Редагування даних про приймання їжі; Збереження змінених даних у базі даних.
Розпізнавання та отримання даних про продукт	class ScannerV	Використання камери для створення зображення їжі; Введення штрихкоду продукту для пошуку; Відображення зображення та результату розпізнавання або пошуку; Перехід до додавання приймання їжі з використанням результатів розпізнавання або пошуку.
Інформація та робота з розумними вагами	class InfoV	Відображення інформації про користувача, застосунок та розробника; Підключення до розумних ваг через Bluetooth; Збереження інформації про користувача.

Серверна частина системи містить три ключові модулі: криптографічний, модуль розпізнавання та модуль створення моделі для розпізнавання. Опис зазначених компонентів наведено в табл. Б.4.

Таблиця Б.4 — Опис модулів серверної частини

Модуль	Структура	Функції
1	2	3
Створення моделі для розпізнавання	MLModel.py pretrained.py evaluation.py visualization.py	Попередня обробка зображень їжі; Формування структури моделей для розпізнавання; Навчання моделей на навчальній вибірці; Тестування моделей на тестовій вибірці; Формування та виведення статистичних результатів; Збереження навченої моделі для використання.

Продовження таблиці Б.4

1	2	3
Розпізнавання	MLWebServer.py	Надсилання ключів шифрування на застосунок; Отримання HTTP-запиту від застосунку, що містить зображення їжі; Розпізнавання їжі створеною моделлю; Передача результатів розпізнавання у застосунок.
Криптографія	Cryptography.py	Розшифрування отриманих зображень для обробки.

Б.4 Використані технічні засоби

Розробка клієнтської частини застосунку була здійснена мовою C# у середовищі Microsoft Visual Studio. Під час проєктування архітектури було використано шаблон проєктування Model-View-ViewModel, а для побудови інтерфейсу користувача — фреймворк .NET MAUI.

Оскільки застосунок призначений для роботи на пристроях під керуванням Android та iOS, коректне функціонування залежить від стабільної роботи відповідних мобільних операційних систем.

Серверна частина була реалізована мовою Python з використанням таких бібліотек, як Flask, Keras, TensorFlow, NumPy, Pillow, Matplotlib та PyCryptodome. Для її розроблення також було застосовано Microsoft Visual Studio.

Сервер може працювати на будь-якій операційній системі, що підтримує Python та перелічені бібліотеки, забезпечуючи універсальність розгортання.

Мінімальні вимоги до апаратного та програмного забезпечення для стабільної роботи мобільного застосунку визначені наступним чином:

- операційна система: Android 5.0 або новіша, iOS 11 або новіша;
- процесор: двоядерний з частотою 1,5 ГГц або вище;
- вільний простір на диску: щонайменше 250 Мб;
- оперативна пам'ять: не менше 2 Гб;
- підключення до інтернету;

- наявність камери, модуля Bluetooth та GPS.

Для нормальної роботи серверної частини застосунку мінімальні вимоги такі:

- програмне забезпечення: Python 3.9, Flask 3.0.3, Tensorflow 2.15.0, Keras 2.15.0, NumPy 1.26.4, Pillow 9.4.0, Matplotlib 3.9.2, PyCryptodome 3.21.0;
- процесор: двоядерний із частотою не менше 2 ГГц;
- вільний простір на диску: не менше 5 Гб;
- оперативна пам'ять: щонайменше 4 Гб;
- підключення до інтернету.

Б.5 Виклик і завантаження

Для початку роботи з програмним забезпеченням необхідно завантажити мобільний застосунок за допомогою файлу `com.panikat.nutritionassistant-Signed.apk` для ОС Android. Після цього на мобільному пристрої з'явиться іконка застосунку з назвою «Моє харчування». Натискання на іконку призводить до запуску програми.

Для роботи серверної частини на ній необхідно запустити файл `MLWebServer.py`

Б.6 Початкові та вихідні дані

Вхідні дані програми містять інформацію про приймання їжі та користувача, яка зберігається в базі даних, про їжу, яка зберігається в файлі JSON, графічні зображення для інтерфейсу користувача, модель для розпізнавання, що використовується на серверній частині, а також зображення, які застосовуються для навчання цієї моделі.

Вихідні дані програми містять оновлену інформацію про приймання їжі та користувача, збережену у БД, а також модель для розпізнавання.

ДОДАТОК В
Текст програми

B.1 Текст файла ScannerVM.cs

```

using System.Net.Http.Headers;
using System.Net.Http.Json;
using System.Security.Cryptography;
using System.Text;
using System.Text.Json;
using System.Windows.Input;
using NutritionAssistant.General;
using NutritionAssistant.Models;

namespace NutritionAssistant.ViewModels
{
    public class ScannerVM : Notifier
    {
        private string imageSrc, productCode, result;
        private static readonly string alg = "Hybrid";
        private Cryptography crypto = new Cryptography();
        private static readonly string url =
"http://10.0.2.2:8080/classify";
        private static readonly string urlKey =
"http://10.0.2.2:8080/keys";
        private static readonly string urlProduct =
"https://world.openfoodfacts.net/api/v2/product/";
        public ProductRoot ProductData { get; set; }
        public bool IsProductCode { get; set; }
        public string ImageSrc
        {
            get { return imageSrc; }
            set { SetProperty(ref imageSrc, value); }
        }
        public string ProductCode
        {
            get { return productCode; }
            set { SetProperty(ref productCode, value);
(SearchCommand as Command)?.ChangeCanExecute(); }
        }
        public string Result
        {
            get { return result; }
            set { SetProperty(ref result, value); }
        }
        public ICommand FoodCommand { get; private set; }
        public ICommand SearchCommand { get; private set; }
        public Func<string, string, Task> ErrorMessage {
get; set; }

        public ScannerVM()
        {
            FoodCommand = new Command(execute:
MakePrediction, canExecute: () => true);

```

```

        SearchCommand = new Command(execute:
SearchProduct, canExecute:
!string.IsNullOrEmpty(ProductCode));
    }

    public async void SearchProduct()
    {
        Clear();
        using HttpClient client = new HttpClient();
        try
        {
            HttpResponseMessage response = await
client.GetAsync(urlProduct + ProductCode +
"?product_type=all&fields=product_name%2Cnutriments%2Cimage_url"
);
            response.EnsureSuccessStatusCode();

            string jsonResponse = await
response.Content.ReadAsStringAsync();
            ProductData =
JsonSerializer.Deserialize<ProductRoot>(jsonResponse);

            if (ProductData?.Product != null)
            {
                Result =
ProductData.Product.ProductName;
                ImageSrc =
ProductData.Product.ImageUrl;
                IsProductCode = true;
            }
            else throw new Exception();
        }
        catch (Exception)
        {
            await ErrorMessage?.Invoke("Помилка
пошуку", "Помилка! Будь ласка, спробуйте пізніше, перевірте
правильність даних або зверніться до служби підтримки.");
            Result = "Помилка!";
        }
    }

    public async void MakePrediction()
    {
        try
        {
            await TakePhoto();
            if (string.IsNullOrEmpty(ImageSrc)) return;
            byte[] key = null;
            using var client = new HttpClient();
            using var algStrContent = new
StringContent(alg, Encoding.UTF8, "text/txt");

```

```

        using var content = new
MultipartFormDataContent { { algStrContent, "alg", "alg.txt" }
};
        if (alg == "Hybrid")
        {
            crypto.AlgRSA = new
RSACryptoServiceProvider(2048);
            byte[] publickey =
crypto.AlgRSA.ExportRSAPublicKey();
            var keyContent = new
ByteArrayContent(publickey);
            keyContent.Headers.ContentType =
MediaTypeHeaderValue.Parse("text/txt");
            content.Add(keyContent, "key",
"key.txt");
        }
        using var response = await
client.PostAsync(urlKey, content);
        if (response.IsSuccessStatusCode)
            key = await
response.Content.ReadAsByteArrayAsync();
        if (key != null)
        {
            crypto.SetKeys(key, alg);
            byte[] origImgBytes =
File.ReadAllBytes(ImageSrc);
            byte[] encrImgBytes =
crypto.Encryption(alg, origImgBytes);

            using var client2 = new HttpClient();
            using var encrImgContent = new
ByteArrayContent(encrImgBytes);
            encrImgContent.Headers.ContentType =
MediaTypeHeaderValue.Parse("image/jpeg");
            using var content2 = new
MultipartFormDataContent
            {
                { encrImgContent, "encrImage",
"encrImage.jpg" },
                { algStrContent, "alg", "alg.txt" }
            };
            using var response2 = await
client2.PostAsync(url, content2);
            if (response2.IsSuccessStatusCode)
            {
                var prediction = await
response2.Content.ReadFromJsonAsync<Prediction>();
                Result = prediction?.Class ??
"Помилка!";
            }
            else
            {

```

```

        if (ErrorMessage != null)
        {
            var error =
(int)response2.StatusCode;
            if (error > 399 && error < 500)
                await ErrorMessage?.Invoke("Помилка розпізнавання", "Помилка на
                стороні клієнта! Будь ласка, спробуйте пізніше або зверніться до
                служби підтримки.");
            else if (error > 499 && error <
600)
                await ErrorMessage?.Invoke("Помилка розпізнавання",
                "Помилка на стороні сервера! Будь ласка, спробуйте пізніше або
                зверніться до служби підтримки.");
            else
                await
ErrorMessage?.Invoke("Помилка розпізнавання", "Помилка! Будь
                ласка, спробуйте пізніше або зверніться до служби підтримки.");
        }
        Result = "Помилка!";
    }
    }
    else throw new Exception();
}
catch (Exception)
{
    await ErrorMessage?.Invoke("Помилка
розпізнавання", "Помилка! Будь ласка, спробуйте пізніше або
зверніться до служби підтримки.");
    Result = "Помилка!";
}
}

private async Task TakePhoto()
{
    Clear();
    ProductCode = string.Empty;
    if (MediaPicker.Default.IsCaptureSupported)
    {
        FileResult photo = await
MediaPicker.Default.CapturePhotoAsync();
        if (photo != null)
        {
            string localFilePath =
Path.Combine(FileSystem.CacheDirectory, photo.FileName);

            using Stream sourceStream = await
photo.OpenReadAsync();
            using FileStream localFileStream =
File.OpenWrite(localFilePath);

            await
sourceStream.CopyToAsync(localFileStream);
            ImageSrc = localFilePath;
        }
    }
}

```

```

    }

    public void Clear()
    {
        Result = string.Empty;
        ImageSrc = string.Empty;
        IsProductCode = false;
    }
}

```

B.2 Текст файла MLModel.py

```

import visualization
import pretrained
import tensorflow as tf
from pathlib import Path
from keras.models import Sequential
from keras import regularizers
from keras.callbacks import Callback
from keras.optimizers import Adam
from keras.layers import Rescaling, Flatten, Dense,
Dropout, RandomFlip, RandomRotation, RandomZoom, Conv2D,
MaxPooling2D, BatchNormalization, GlobalAveragePooling2D
from timeit import default_timer as timer

# For time count
class TimingCallback(Callback):
    def __init__(self, logs={}):
        self.logs=[]
    def on_train_begin(self, logs={}):
        self.logs=[]
    def on_epoch_begin(self, epoch, logs={}):
        self.starttime = timer()
    def on_epoch_end(self, epoch, logs={}):
        self.logs.append(timer()-self.starttime)

if __name__ == '__main__':
    # Start data
    path = Path().absolute().as_posix()
    epochs, batch_size = 60, 25
    image_size = (512, 512)
    num_classes = 30
    models = dict()
    use_pretrained = False
    cb = TimingCallback()
    ds_train = tf.data.Dataset.load(f'{path}/ds_train')
    ds_val = tf.data.Dataset.load(f'{path}/ds_val')

    def preprocess_image(image, label):

```

```

        image = tf.image.resize(image, image_size)
        return image, label

    ds_train =
ds_train.map(preprocess_image).batch(batch_size).cache().prefetch(
h(tf.data.AUTOTUNE)
    ds_val =
ds_val.map(preprocess_image).batch(batch_size).cache().prefetch(
tf.data.AUTOTUNE)

    # Ask for using pretrained models
    print('Use pretrained models? (Y/N):')
    x = input()
    if x == 'y' or x == 'Y':
        use_pretrained = True

    # Data augmentation and normalization
    data_augmentation = Sequential([
        RandomFlip('horizontal', input_shape=(*image_size,
3)),
        RandomRotation(0.1),
        RandomZoom(0.1),
        Rescaling(1./255)
    ])

    # Create models
    models['MealNet'] = Sequential([
        data_augmentation,
        Conv2D(16, 3, padding='same', activation='relu'),
        MaxPooling2D(2, strides=2),
        Conv2D(32, 3, padding='same', activation='relu'),
        MaxPooling2D(2, strides=2),
        Conv2D(32, 3, padding='same', activation='relu'),
        BatchNormalization(),
        MaxPooling2D(2, strides=2),
        Conv2D(64, 3, padding='same', activation='relu'),
        MaxPooling2D(2, strides=2),
        Conv2D(128, 3, padding='same', activation='relu'),
        MaxPooling2D(2, strides=2),
        Flatten(),
        Dense(256, activation='relu',
kernel_regularizer=regularizers.l2(0.01)),
        Dropout(0.1),
        Dense(128, activation='relu',
kernel_regularizer=regularizers.l2(0.01)),
        Dense(num_classes, activation='softmax')
    ])

    if use_pretrained == True:
        pretrained.pretrained_models(models, image_size,
num_classes, data_augmentation)

    for name, model in models.items():

```

```

        # Compile the model
        model.compile(optimizer=Adam(learning_rate=0.0001),
loss='sparse_categorical_crossentropy', metrics=['accuracy'])
        model.summary()

        # Train the model
        history = model.fit(
            ds_train,
            validation_data=ds_val,
            epochs=epochs,
            batch_size=batch_size,
            callbacks=[cb]
        )

        # Evaluate the model
        score = model.evaluate(ds_val, verbose=0)

        # Visualize training results
        visualization.show_results(history, name, cb,
score, num_classes)

        # Ask for saving
        print('Save the model? (Y/N):')
        x = input()
        if x == 'y' or x == 'Y':
            model.save(f'ML_{name}_{num_classes}.h5')

```

B.3 Текст файлы `evaluation.py`

```

# -*- coding: latin-1 -*-
import tensorflow as tf
import numpy as np
from pathlib import Path
from sklearn.metrics import accuracy_score,
precision_score, recall_score, f1_score, confusion_matrix

if __name__ == '__main__':
    # Start data
    num = 4
    batch_size = 25
    image_size = (512, 512)
    num_classes = [10, 30, 60, 101]
    path = Path().absolute().as_posix()
    ds_val = tf.data.Dataset.load(f'{path}/ds_val')
    models =
[tf.keras.models.load_model('D:/ML_Own1_3_10.h5')]

models.append(tf.keras.models.load_model('D:/ML_Own1_3_30.h5'))

models.append(tf.keras.models.load_model('D:/ML_Own1_3_60.h5'))

```

```

models.append(tf.keras.models.load_model('D:/ML_Own1_3_101.h5'))

models.append(tf.keras.models.load_model('D:/ML_ResNet50V2_10.h5'))

models.append(tf.keras.models.load_model('D:/ML_ResNet50V2_30.h5'))

models.append(tf.keras.models.load_model('D:/ML_ResNet50V2_60.h5'))

models.append(tf.keras.models.load_model('D:/ML_ResNet50V2_101.h5'))

def filter_classes(_, y, i): return y.numpy() <
num_classes[i]

def preprocess_image(image, label):
    image = tf.image.resize(image, image_size)
    return image, label

ds_val_arr, y_true = [], []
ds_val_arr.append(ds_val.filter(lambda x, y:
tf.py_function(func=filter_classes, inp=[x, y, 0],
Tout=tf.bool)).map(preprocess_image).batch(batch_size).cache().p
refetch(tf.data.AUTOTUNE))
ds_val_arr.append(ds_val.filter(lambda x, y:
tf.py_function(func=filter_classes, inp=[x, y, 1],
Tout=tf.bool)).map(preprocess_image).batch(batch_size).cache().p
refetch(tf.data.AUTOTUNE))
ds_val_arr.append(ds_val.filter(lambda x, y:
tf.py_function(func=filter_classes, inp=[x, y, 2],
Tout=tf.bool)).map(preprocess_image).batch(batch_size).cache().p
refetch(tf.data.AUTOTUNE))

ds_val_arr.append(ds_val.map(preprocess_image).batch(batch_size)
.cache().prefetch(tf.data.AUTOTUNE))
for i in range(num):
    y_true.append(np.concatenate([y.numpy() for _, y in
ds_val_arr[i]], axis=0))

def eval(j, k):
    pred_probs = models[k].predict(ds_val_arr[j])
    y_pred = np.argmax(pred_probs, axis=1)
    score = models[k].evaluate(ds_val_arr[j],
verbose=0)

    accuracy = accuracy_score(y_true[j], y_pred)
    precision = precision_score(y_true[j], y_pred,
average='macro')
    recall = recall_score(y_true[j], y_pred,
average='macro')

```

```

f1 = f1_score(y_true[j], y_pred, average='macro')
cm = confusion_matrix(y_true[j], y_pred)

print(f"\n-----Model {k + 1}-----")

print(f"Accuracy: {accuracy:.3f}; {score[1]:.3f}")
print(f"Precision (macro): {precision:.3f}")
print(f"Recall (macro): {recall:.3f}")
print(f"F1-score (macro): {f1:.3f}")
print("Confusion Matrix:")
print(cm)
for m in range(len(cm)):
    TP = cm[m, m]
    FP = cm[:, m].sum() - TP
    FN = cm[m, :].sum() - TP
    TN = cm.sum() - (TP + FP + FN)
    print(f"Class {m}: TP={TP}, FP={FP}, FN={FN},
TN={TN}")

for i in range(num):
    eval(i, i);
    eval(i, i + 4);

```

В.4 Текст файла pretrained.py

```

from keras.models import Sequential
from keras.regularizers import l2
from keras.applications import Xception, InceptionV3,
ResNet50V2, MobileNetV2, DenseNet169
from keras.layers import Dense, GlobalAveragePooling2D

def pretrained_models(models, image_size, num_classes,
data_augmentation):
    pretrained_models = dict()
    pretrained_models['Xception'] =
Xception(weights='imagenet', include_top=False,
input_shape=(*image_size, 3))
    pretrained_models['InceptionV3'] =
InceptionV3(weights='imagenet', include_top=False,
input_shape=(*image_size, 3))
    pretrained_models['ResNet50V2'] =
ResNet50V2(weights='imagenet', include_top=False,
input_shape=(*image_size, 3))
    pretrained_models['MobileNetV2'] =
MobileNetV2(weights='imagenet', include_top=False,
input_shape=(*image_size, 3))
    pretrained_models['DenseNet169'] =
DenseNet169(weights='imagenet', include_top=False,
input_shape=(*image_size, 3))
    for pname, pmodel in pretrained_models.items():
        pretrained_models[pname].trainable = False

```

```

        models[pname] = Sequential([
            data_augmentation,
            pmodel,
            GlobalAveragePooling2D(),
            Dense(256, activation='relu',
kernel_regularizer=l2(0.01)),
            Dense(128, activation='relu',
kernel_regularizer=l2(0.01)),
            Dense(num_classes, activation='softmax')
        ])

```

B.5 Текст файла MealVM.cs

```

using NutritionAssistant.General;
using NutritionAssistant.Models;
using NutritionAssistant.Views;
using System.Windows.Input;

namespace NutritionAssistant.ViewModels
{
    public class MealVM : Notifier
    {
        private Meal selectedMeal = new();
        private TimeSpan time = DateTime.Now.TimeOfDay;
        public int Index { get; set; } = -1;
        public MealV Parent { get; set; }
        public DateTime MaxDate { get; set; } =
DateTime.Today;
        public Meal SelectedMeal
        {
            get { return selectedMeal; }
            set
            {
                if (value != null) Time =
value.MealDateTime.TimeOfDay;
                SetProperty(ref selectedMeal, value);
                (SaveCommand as
Command)?.ChangeCanExecute();
            }
        }
        public TimeSpan Time
        {
            get { return time; }
            set { SetProperty(ref time, value); }
        }
        public ICommand SaveCommand { get; protected set; }
        public ICommand AddFoodCommand { get; private set; }
    }

    public ICommand RemoveFoodCommand { get; private
set; }

```

```

public MealVM()
{
    SaveCommand = new Command(
        execute: async () =>
        {
            SelectedMeal.MealDateTime =
SelectedMeal.MealDateTime.Date + Time;
            var result = await
CommonData.Database.SaveItemAsync(SelectedMeal);
            if (result)
            {
                if (Index != -1)
CommonData.Meals[Index] = SelectedMeal;
                else
CommonData.Meals.Add(SelectedMeal);
            }
            Parent?.GoBack();
        },
        canExecute: () => { return
SelectedMeal.Food.Count != 0; });

    AddFoodCommand = new Command(
        execute: () =>
        {
            var foodTmp = new MealFood();
            foodTmp.PropertyChanged +=
SelectedMeal.TotalCaloriesUpdateEvent;
            SelectedMeal.Food.Add(foodTmp);
        },
        canExecute: () => true);

    RemoveFoodCommand = new Command(
        execute: (object food) =>
        {
            var foodTmp = (MealFood)food;
            foodTmp.PropertyChanged -=
SelectedMeal.TotalCaloriesUpdateEvent;
            SelectedMeal.Food.Remove(foodTmp);
            (SaveCommand as
Command)?.ChangeCanExecute();
        },
        canExecute: (object food) => true);
    }
}

```

ДОДАТОК Г
Слайди презентації



Рисунок Г.1 — Слайд № 1

Мета, об'єкт, предмет дослідження

- **Об'єкт дослідження** – процес розробки програмного забезпечення для обліку харчування.
- **Предмет дослідження** – програмне забезпечення для обліку харчування.
- **Мета роботи** – створення програмного забезпечення, яке дозволяє користувачеві вести облік харчування, отримувати дані про спожиту їжу, здійснювати розпізнавання їжі, зчитувати фізіологічні дані з розумних ваг.
- **Наукова новизна** полягає у комплексному поєднанні сучасних методів комп'ютерного зору, мобільної розробки та засобів криптографічного захисту даних для забезпечення процесу обліку харчування.
- **Сфера застосування** – будь-яка сфера, де користувач має потребу в контролі харчування та калорійності, а саме: особисте використання, дієтологія, спортивне харчування та медичні послуги.

2

Рисунок Г.2 — Слайд № 2

Огляд наявних програмних засобів

Можливості	«Lifesum»	«FatSecret»	«DietAI»	Власний застосунок
Багатофункціональність	+	+	–	+
Українська локалізація	–	+	+	+
Робота з інформацією про приймання їжі	+	+	+	+
Робота з інформацією про користувача	+	+	+	+
Розпізнавання їжі за зображенням	–	–	+	+
Використання баз даних	+	+	+	+
Використання штрихкодів продуктів	+	+	+	+
Взаємодія з розумними вагами	–	–	–	+
Підтримка дієт і планів харчування	+	+	+	–
Інтеграція з фітнес-трекерами	+	+	+	–
Трекер вживання води	+	+	+	–
Доступ до базової функціональності безкоштовно	+	+	–	+

3

Рисунок Г.3 — Слайд № 3

Вибір мови програмування

Для мобільного застосунку

Критерії порівняння	Java	Swift	Kotlin	C#
Зрозумілість синтаксису	—	+	+	+
Підвищена продуктивність	+	+	+	+
Підтримка багатопоточності	+	+	+	+
Підтримка IDE	+	+	+	+
Простота вивчення	—	+	+	+
Підтримка спільноти	+	+	+	+
Бібліотеки сторонніх розробників	+	+	+	+
Інструментарій та екосистема	+	+	+	+
Розробка під Android	+	—	+	+
Розробка під iOS	—	+	—	+
Кросплатформна розробка	—	—	+	+

Для серверної частини

Критерії порівняння	Java	C++	C#	Python
Зрозумілість синтаксису	—	—	+	+
Підвищена продуктивність	+	+	+	±
Підтримка багатопоточності	+	+	+	+
Підтримка IDE	+	+	+	+
Простота вивчення	—	—	+	+
Підтримка спільноти	+	+	+	+
Бібліотеки сторонніх розробників	+	+	+	+
Інструментарій та екосистема	+	+	+	+
Бібліотеки машинного навчання	—	+	+	+
Вебфреймворки	+	—	+	+
Кросплатформна розробка	+	+	+	+

4

Рисунок Г.4 — Слайд № 4

Вибір середовища розробки

Критерії порівняння	Xcode	Android Studio	Eclipse	Visual Studio
Кросплатформність	—	—	+	+
Підсвічування синтаксису	+	+	+	+
Автозавершення коду	+	+	+	+
Швидкий та зручний пошук	+	+	+	+
Перевірка та аналіз коду	+	+	+	+
Швидкий та безпечний рефакторинг коду	+	+	+	+
Простота встановлення	+	+	—	+
Інтуїтивний інтерфейс	+	+	—	+
Підтримка пакетів	+	+	+	+
Підтримка репозиторіїв	+	+	+	+
Підтримка C#	—	—	+	+
Підтримка Python	—	—	+	+

5

Рисунок Г.5 — Слайд № 5

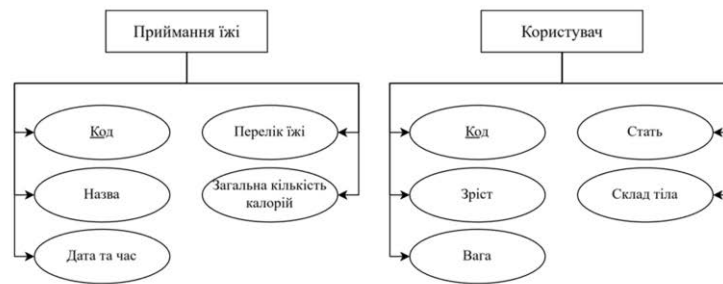
Вибір системи управління базами даних

Критерії порівняння	PostgreSQL	MySQL	MongoDB	SQLite
Швидкодія	+	+	+	+
Функціональність	+	+	+	+
Надійність	+	+	—	+
Масштабованість	+	+	+	—
Підтримка SQL	+	+	—	+
Зручність використання	+	+	—	+
Відкритість та спільнота	+	+	+	+
Сумісність з мобільними платформами	+	+	+	+
Вбудованість	—	—	—	+

6

Рисунок Г.6 — Слайд № 6

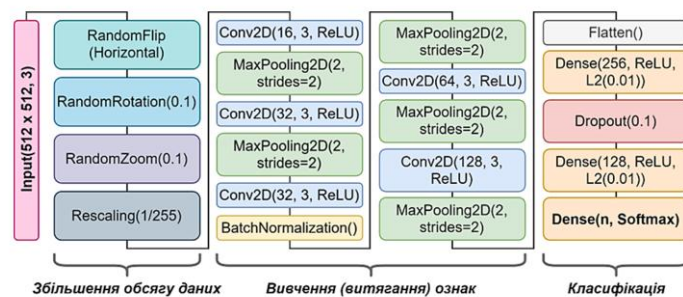
Концептуальна модель бази даних



10

Рисунок Г.10 — Слайд № 10

Модифікована модель ЗНМ MealNet



11

Рисунок Г.11 — Слайд № 11

Створені моделі для розпізнавання

Модель	Час навч. (с)	Втрата навч. (%)	Точн. навч. (%)	Втрата тест. (%)	Точн. тест. (%)	Розмір (Мб)
10 класів						
MealNet	741,4	1,86	96	4,71	35	97,7
InceptionV3	3725,9	3,19	95	3,53	72	90,1
Xception	4618,0	2,45	96	2,67	81	86,2
ResNet50V2	4191,0	2,61	100	3,16	71	96,6
MobileNetV2	1536,7	3,22	98	3,86	62	13,1
DenseNet169	5407,3	2,50	97	2,94	76	54,5
30 класів						
MealNet	3120,6	1,28	92	6,02	20	97,7
InceptionV3	7853,2	2,59	83	2,96	63	90,1
Xception	13628,2	2,40	78	2,70	63	86,3
ResNet50V2	10915,3	2,00	98	2,85	64	96,6
MobileNetV2	3415,2	2,67	87	3,42	57	13,1
DenseNet169	13947,9	2,37	80	2,87	58	54,6
60 класів						
MealNet	6127,5	1,44	88	7,73	9	97,7
InceptionV3	17200,9	2,78	61	3,04	52	90,2
Xception	25708,8	2,88	57	3,03	51	86,3
ResNet50V2	22869,0	2,17	87	3,08	54	96,7
MobileNetV2	7738,0	2,85	68	3,46	45	13,1
DenseNet169	26831,0	2,75	65	3,23	47	54,6
101 клас						
MealNet	9602,2	1,70	82	9,41	7	97,8
InceptionV3	28426,0	3,23	46	3,38	39	90,2
Xception	44037,7	3,34	41	3,44	37	86,4
ResNet50V2	35668,5	2,56	72	3,21	48	96,7
MobileNetV2	13798,4	3,27	50	3,70	37	13,2
DenseNet169	46003,3	3,15	49	3,45	38	54,7

12

Рисунок Г.12 — Слайд № 12

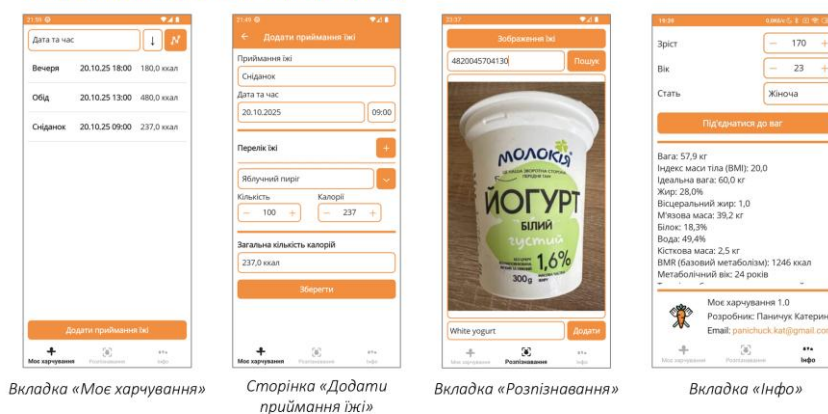
Впроваджені методи шифрування

Критерії порівняння	AES	RSA	<u>AES та RSA</u>
Ключі	Один спільний ключ	Приватний і публічний ключі	Комбінація ключів
Розмір ключа	256 біт	2048 біт	256 та 2048 біт
Швидкодія	2,06 с.	34,51 с.	1,01 с.
Обчислювальні витрати	Менші	Більші	Середні
Передача ключів	Складна	Легка	Легка
Складність управління ключами	Висока	Низька	Низька

13

Рисунок Г.13 — Слайд № 13

Приклади інтерфейсу



14

Рисунок Г.14 — Слайд № 14

Висновки

- В створеному програмному забезпеченні було реалізовано усі заплановані функції. Тестування програми довело, що увесь розроблений функціонал працює коректно.
- В подальшому можливо покращити створене програмне забезпечення за допомогою:
 - додавання інформації про білки, жири та вуглеводи їжі;
 - групування приймання їжі за датою;
 - використання норми вживання калорій на день;
 - покращення моделі розпізнавання їжі, механізмів забезпечення безпеки та розширення переліку розпізнавання тощо.

15

Рисунок Г.15 — Слайд № 15