МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

НОВИЦЬКИЙ МАКСИМ ОЛЕКСАНДРОВИЧ

Допускається до захисту:
завідувач кафедри
інформаційних технологій,
доктор технічних наук, доцент
_____Т. В. Нескородєва
«____» _____ 20__ р.

РОЗРОБКА МОБІЛЬНОГО ЗАСТОСУНКУ АНАЛІЗУ ОСОБИСТОГО РАЦІОНУ ЛЮДИНИ

Спеціальність 122 «Комп'ютерні науки»
Кваліфікаційна (бакалаврська) робота

Керівник:
Бабаков Р. М., доцент кафедри
інформаційних технологій,
д.т.н, доцент

Оцінка: ___/___/_____
(бали за шкалою ЄКТС/за національною шкалою)
Голова ЕК: _____
(підпис)

Вінниця – 2022

АНОТАЦІЯ

Новицький М. О. Розробка мобільного застосунку аналізу особистого раціону людини. Спеціальність 122 «Комп’ютерні науки», освітня програма «Сучасні інформаційні технології та програмування». Донецький національний університет імені Василя Стуса, Вінниця, 2022.

У кваліфікаційній (бакалаврській) роботі досліджено забезпечення можливості персонального нагляду за здоров’ям користувача смартфона, а також можливості полегшення нагляду за своїм харчуванням, приділяючи цьому незначну частину часу порівняно з ручним пошуком і збереженням необхідної інформації. Показано процес розробки мобільного застосунку для аналізу раціону. Розроблено персональний мобільний додаток для нагляду за раціоном.

Ключові слова: мобільний додаток, Android, Flutter, Dart, здоров’я, дієта.

57 с., 36 рис., 30 джерел.

ABSTRACT

Novytskyi M. Development of a mobile application for the analysis of a person’s personal diet. Specialty 122 “Computer Science”, educational program “Modern Information Technology and Programming”. Vasyl’ Stus Donetsk National University, Vinnytsia, 2022.

The qualification (bachelor's) work explored the possibility of personal supervision of the health of the smartphone user, as well as the possibility of facilitating the supervision of their food, devoting a small amount of time compared to manually searching and storing the necessary information. The process of developing a mobile application for diet analysis has been shown. A personal mobile application for diet monitoring has been developed.

Keywords: mobile application, Android, Flutter, Dart, health, diet.

57 p., 36 pict., 30 sources.

ЗМІСТ

ВСТУП

З кожним роком значення мобільних пристроїв зростає у нашому повсякденному житті. Людство все більше і частіше користується смартфонами, смарт-годинниками, планшетами, та іншими компактними девайсами. Вдома, на роботі, на відпочинку значна частка людей на Землі перевіряє нові повідомлення в месенджерах, дзвонить по мобільному зв'язку та з використанням інтернету, або просто грає в ігри. Світова спільнота дедалі більше інтегрується з переносними девайсами, популярність яких зростає, продажі збільшуються, ринки зростають. Також не можна не відзначити кількість розробок та інновацій у користувацькому сегменті ринку переносних девайсів.

Значну частину вищезазначених пристроїв стали представляти переносні девайси на кшталт фітнес-годинників та смарт-браслетів, що вже навчились визначати кількість пройдених кроків, темп, частоту серцевих скорочень та навіть сатурацію. Популярність таких наручних засобів наочно демонструє нам, що дедалі більша частина користувачів хочуть не просто оточити себе якнайбільшою кількістю електронних пристроїв, але й впевнитись, що вони приносять користь для свого власника. Зрештою, слідкувати за своїм здоров'ям стало просто трендом, а компанії-виробники девайсів досить успішно задовольняють цей немалий попит, надаючи змогу робити це в стилі сучасного життя.

Їжа – один з найважливіших факторів, що впливають на наше здоров'я, нарівні з фізичною активністю. Неправильне харчування – дуже поширена проблема, зокрема і серед користувачів смартфонів. Значна кількість людей, які бажають підтримувати себе у гарній формі, але не хочуть витрачати кошти на особистого дієтолога, або не хочуть витрачати свій час на пошук всієї необхідної інформації, стикаються з проблемою вибору зручного інструменту для задоволення своїх потреб у моніторингу харчування. Таким інструментом може

стати мобільний застосунок. Автор вважає, що все вищезазначене складає актуальність даного дослідження.

Мета даного дослідження: розробити мобільний застосунок, що може використовуватись для аналізу особистого раціону людини.

Завдання цієї роботи:

- проаналізувати наявні на ринку та в житті засоби та можливості нагляду за своїм раціоном;
- зробити висновки з отриманого аналізу та обрати необхідні засоби та інструменти для розробки нового продукту;
- розробити програму для смартфонів, що може успішно та в необхідному обсязі задовольнити потребу в моніторингу власного раціону в зручному вигляді.

Об’єктом дослідження є забезпечення можливості персонального нагляду за здоров’ям користувача смартфона, а також можливості полегшення нагляду за своїм харчуванням, приділяючи цьому незначну частину часу порівняно з ручним пошуком і збереженням необхідної інформації.

Предметом дослідження є процес розробки мобільного застосунку для аналізу раціону.

Знання та розробки, отримані здобувачем у ході виконання бакалаврської роботи, представляють собою прикладний програмний продукт, що стане в нагоді людям, які потребують засіб для моніторингу свого раціону прямо у смартфоні. Отримані теоретичні та практичні знання допоможуть здобувачу отримати досить детальне уявлення про ринок програмних продуктів, що відповідають темі дослідження, а також продовжити розробку та покращення наявного застосунку, що, в свою чергу, має перспективу принести ще більше користі для тих, хто потребує такий продукт.

Бакалаврська робота складається з трьох розділів. В першому з них проведений аналіз конкурентів на ринку програмного забезпечення та в житті. В другому проведене дослідження з метою обрати найбільш підходящі засоби та інструменти розробки. В третьому розділі описано процес розробки застосунку. Використано 30 посилань, наведено 36 рисунків.

РОЗДІЛ 1
ОГЛЯД ІСНУЮЧИХ АНАЛОГІВ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Огляд аналогів

Цей розділ присвячений огляду існуючих аналогів, що поширюються на таких майданчиках-магазинах для застосунків, як Google Play та App Store. Перший доступний для користувачів операційної системи Android, другий – для користувачів екосистеми Apple, включаючи IOS та iPad OS.

1.1.1. HealthifyMe

HealthifyMe – достатньо популярний додаток, що доступний як на пристроях Apple, так і на пристроях з операційною системою Android. Продукт розробників з Індії, де є найпопулярнішим застосунком такого типу. Має як безкоштовний, так і преміум функціонал.

HealthifyMe відрізняється тим, що в додатку доступні різні способи використання, і функціонал не обмежується тільки дієтами. В HealthifyMe існує також трекінг тренувань користувача, допомога у досягненні певної ваги, платні плани, що включають в себе як харчування, так і тренування. Є також достатньо унікальна функція студій, що дозволяють займатися спортом по відео зв’язку в групі людей та з тренером. Звісно, багато з цього не безкоштовне, і в безкоштовному користуванні є достатньо обмежень. Для того, щоб досягти мети скинути вагу, користуючись цим застосунком, радше за все треба буде заплатити. Також є огляд трансформацій користувачів HealthifyMe (фото змін, що зазнало тіло після тренувань та харчування, рекомендованого HealthifyMe), і блог, де користувачі діляться рецептами або своїм досвідом. На думку автора, блог може дратувати або заважати деяким користувачам. Що стосується інтерфейсу, він тут реалізований не кращим чином, так що загальний стиль,

реклама власних функцій і кількість рухомих елементів можуть відштовхнути від користування. Окремо слід виділити невисоку швидкодію. Інтерфейс наведено на рис. 1.1.

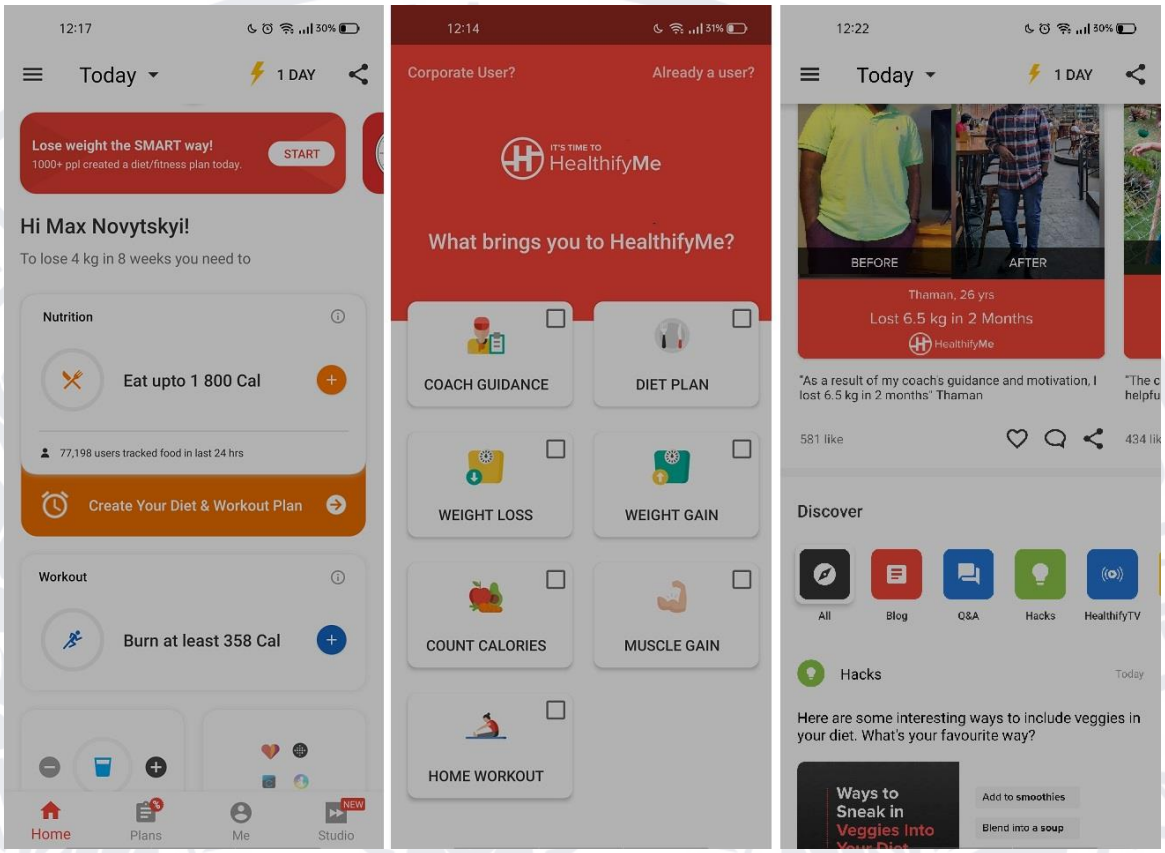


Рисунок 1.1 – Інтерфейс HealthifyMe

Серед переваг можна зазначити:

- широкий функціонал;
- функція онлайн-тренувань з тренерами;
- тісне поєднання догляду за раціоном та фізичною активністю;
- широкі можливості в допомозі схуднути;
- існує безкоштовний план користування;
- велика база їжі.

Але існують також і недоліки:

- поганий рівень технічної підтримки;
- низька швидкодія;

- недостатньо відпрацьована система онлайн-тренувань;
- агресивна політика сповіщень;
- неможливо придбати платну підписку тільки за один місяць;
- неоднозначний дизайн.

1.1.2. Lifesum

Lifesum – популярний застосунок, який можна завантажити на будь-яку з двох найпопулярніших операційних систем. Безкоштовний, але деякий, не критичний функціонал потрібно придбати, тому що він включений до платної підписки.

Lifesum одразу вирізняється серед інших приємним та зручним дизайном, що є дійсно важливим і допомагає захопити аудиторію. Розробники пропонують відслідковувати кількість калорій в спожитій їжі, надавати рекомендації, слідкувати за питтям води, додавати фізичну активність. Є можливість додавати спожиту їжу скануванням штрих-коду. Рейтинг їжі та вміст мікроелементів недоступні без придбання підписки, як і внесення детальної інформації про себе для формування релевантних цілей та рекомендацій. На додачу, застосунок надає можливість обрати дієту або ознайомитись з рецептами, але це – платний функціонал. Інтерфейс наведено на рис. 1.2.

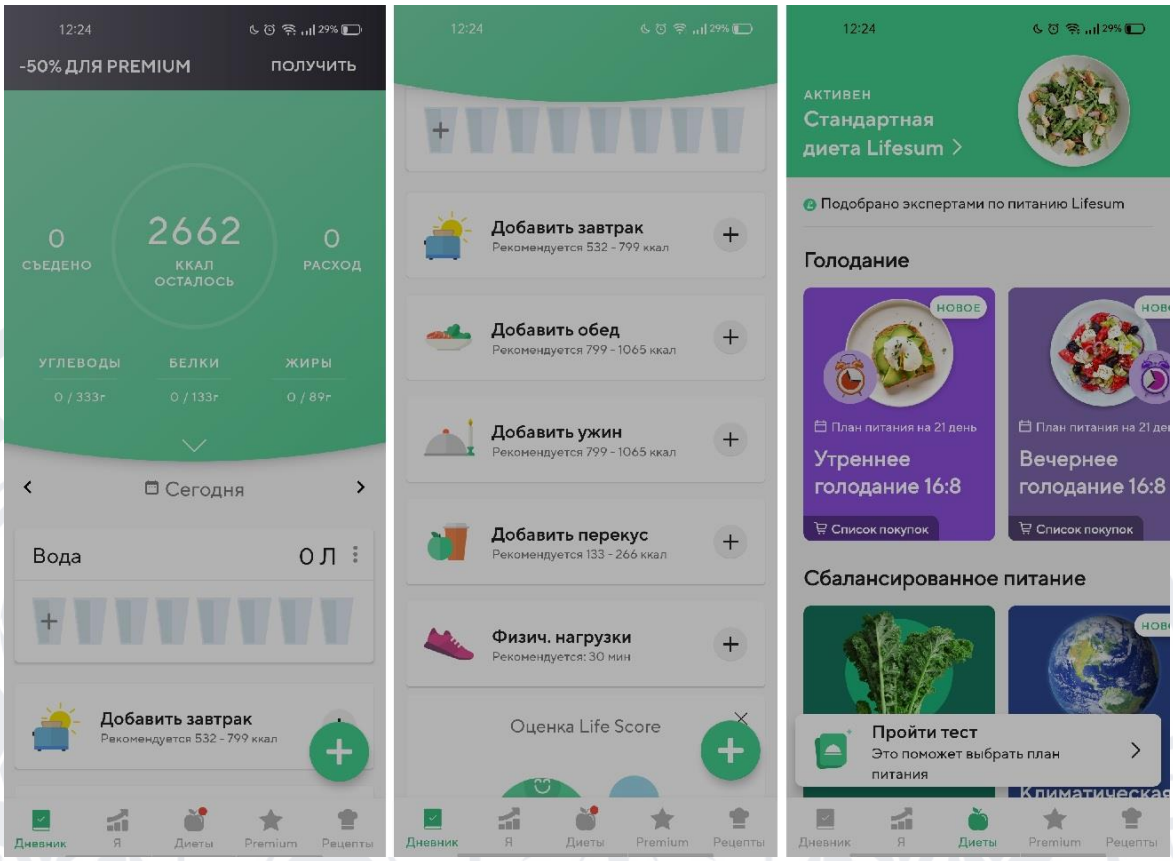


Рисунок 1.2 – Інтерфейс Lifesum

Отже, Lifesum в безкоштовній версії виступає непоганим засобом для відслідковування того, що їси, і з врахуванням фізичної активності вибудувати для себе приблизний план харчування. В платній версії це потужніший інструмент, але все-таки без якихось унікальних особливостей, що здатні справити враження.

Плюси Lifesum:

- можливості інтеграції з фітнес-браслетами та годинниками: Wear OS, Google Fit, S Health (платна підписка розширює ці можливості та дозволяє підключитися до Moves, Nokia Health, FitBit, Jawbone, Runkeeper);
- існує безкоштовна версія;
- чудовий дизайн;
- функція додавання своєї їжі, якщо не знайшов потрібну в базі;

- дієти та рецепти;
- можливість сканування штрих-коду;
- тестування для отримання рейтингу здоров’я.

Мінуси Lifesum:

- іноді неможливо підібрати потрібну або близьку до неї їжу в базі даних (не вистачає складних страв з декількома інгредієнтами та не американських страв);
- трапляються некоректні показники калорій або макро/мікро-нутрієнтів;
- неможливо відредагувати тренування для запису своїх показників часу або втрачених калорій;
- існують проблеми з інтеграцією фітнес-браслетів та годинників.

1.1.3. Foodvisor

Foodvisor – менш популярний застосунок, що доступний на обох платформах-конкурентах. Причиною його меншої популярності може бути те, що стратегія платної підписки в цьому випадку суттєво відрізняється від конкурентів.

Foodvisor пропонує схему, запозичену в багатьох онлайн-сервісів, коли користувачу пропонується або одразу придбати платну підписку, або почати пробний період платного функціоналу тривалістю 14 днів. І з цим пов’язані дві проблеми, що потенційно лякають користувачів. По-перше, вищеописаний вибір дають лише після того, як користувач пройде через декілька десятків екранів з запитаннями про показники, вподобання, та стиль життя. Середній користувач, здивований такою кількістю запитань, потім може бути розчарований тим, що для оформлення пробного періоду необхідно прив’язати свою банківську картку до застосунку. Це і є друга проблема для користувача. Звісно, автоматичне

списання можна в будь-який момент відмінити. Але далеко не кожен буде мати бажання додавати свою картку, з якої згодом, після пробного періоду, можливе списання коштів.

Загалом, розробники пропонують нам відслідковування спожитої їжі (калорій, поживних речовин та мікроелементів), а також споживання води, фізичної активності. Отже, базовий функціонал для подібного роду програм. Foodvisor вирізняється сучасним гарним дизайном та декількома особливостями, як-то сканер не тільки штрих-кодів, але і безпосередньо розпізнаванням їжі на фото; а також глибокими налаштуваннями своїх вподобань та образу життя, що може бути використано персональним дієтологом. І так, дієтологи разом з планами дієт, планами харчуваннями та рецептами доступні лише там, хто платить. Інтерфейс наведено на рис. 1.3.

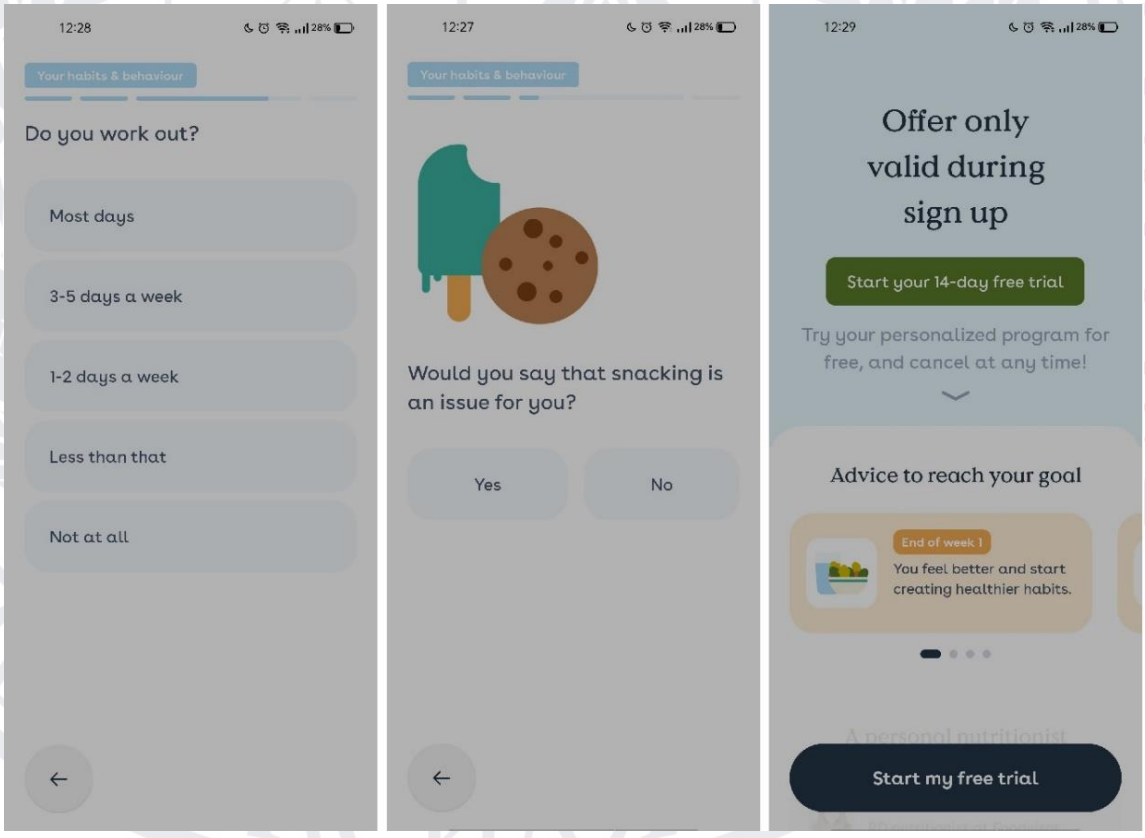


Рисунок 1.3 – Інтерфейс Foodvisor

Узагальнюючи, виділимо такі переваги Foodvisor, що базуються на аналітиці та відгуках користувачів:

- сучасний приємний дизайн;
- можливість вказати стільки відомостей про себе, скільки потрібно для формування якісного плану харчування або дієти;
- персональні дієтологи;
- функція розпізнавання їжі на фото;
- дієти та рецепти;
- функція сканування штрих-коду.

Враховуючи контраверсійність цього програмного рішення, згадаємо про недоліки:

- відсутність безкоштовної версії;
- висока плата за користування;
- проблеми в роботі інструмента розпізнавання їжі;
- необхідність відповісти на декілька десятків запитань перед тим, як застосунок повідомляє, що для продовження потрібно прив'язати банківську картку.

1.1.4. FatSecret

FatSecret – популярний застосунок, що доступний на системах і Apple, і Google. Існує щонайменше з 2010 року. Хоч і виглядає простішим, але його сильний рейтинг не дасть ввести в оману.

Перший запуск одразу породжує питання «Чому ця програма настільки популярна?». І хоча FatSecret не може похизуватись сучасним дизайном, він залишається міцним гравцем серед конкурентів. Безкоштовна версія надає увесь потрібний функціонал (окрім відслідковування спожитої води), зокрема повний розбір їжі на макроелементи та поживні речовини (рис. 1.4). Нажаль, розширення кількості прийомів їжі, дієти, плани харчування та рецепти залишаються платними.

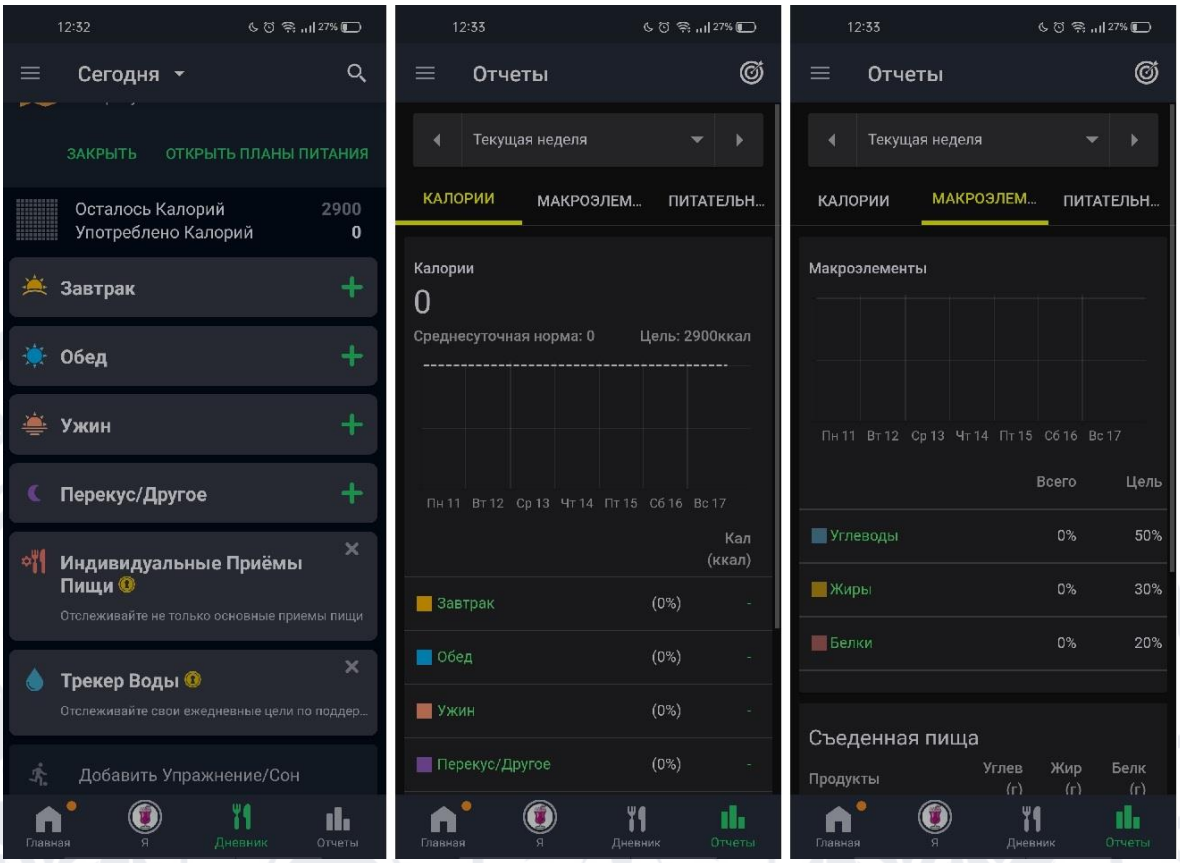


Рисунок 1.4 – Вигляд функціоналу FatSecret

Заслуговує на увагу блог для користувачів, які викладають туди фото спожитої їжі або навіть діляться своїми проблемами або досвідом, а інші користувачі можуть ставити вподобання і підписуватись. При цьому цей блог/соцмережа підлаштовується під регіон, вказаний при стартовому налаштуванні. Автор вважає, що останнє рішення значно вплинуло на зацікавленість користувачів до такої ідеї. Блог можна оглянути на рис. 1.5.

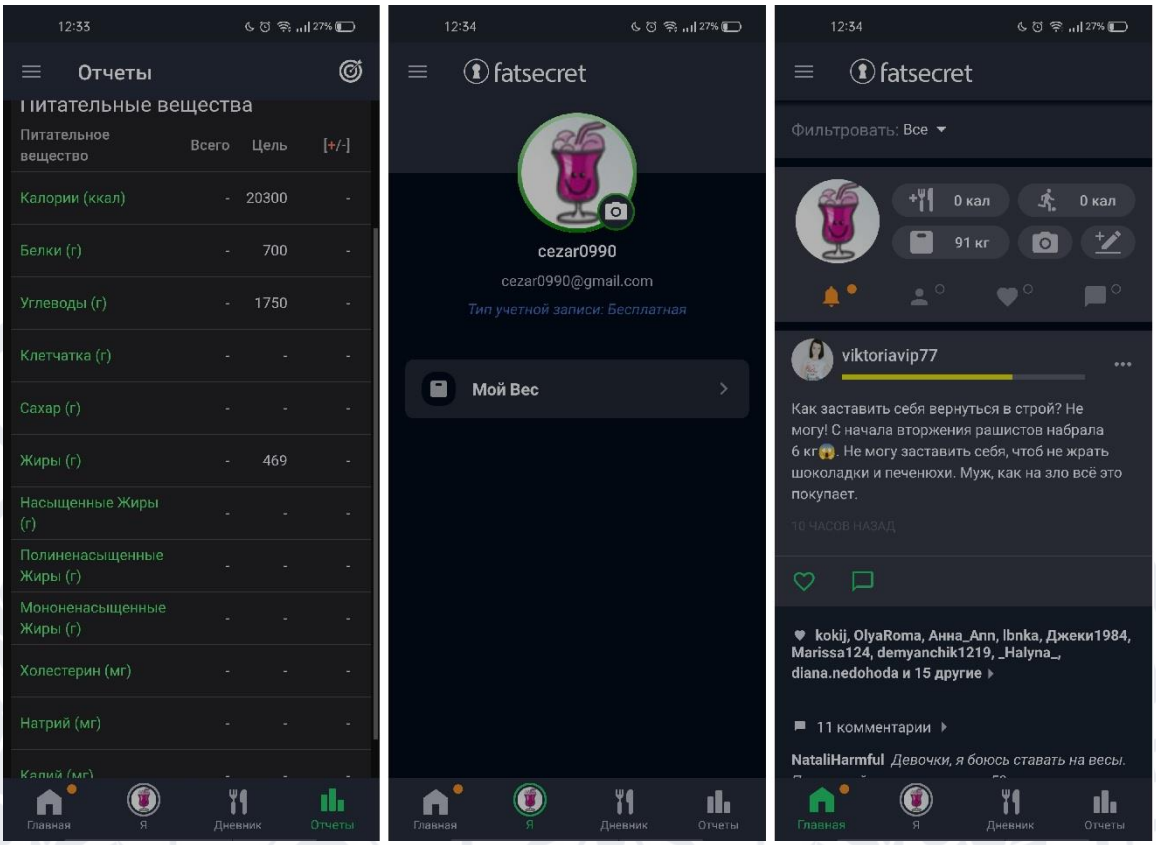


Рисунок 1.5 – Функціонал та блог FatSecret

Переваги FatSecret:

- велика база даних їжі та страв;
- цікава реалізація блогу;
- кількість мікроелементів для кожної страви;
- зручні графіки та діаграми.

Аспекти, які можна вважати недоліками застосунку:

- не найзручніший дизайн, який складно назвати сучасним;
- відсутність моніторингу споживання води в безкоштовній версії;
- інтеграція тільки з Google Fit з-поміж інших подібних програм;
- дивна назва (по повідомленням користувачів).

1.2. Постановка задачі

Аналіз усіх програмних продуктів у попередньому розділі продемонстрував, що серед мобільних програм для слідкування за харчуванням немає однозначного лідера. Такі висновки призвели до формування чітких вимог, що мають бути виконані при розробці нового додатку для успішної конкуренції з іншими подібними застосунками:

- привабливий дизайн;
- універсальність;
- обов’язкова присутність даних про поживні речовини для усіх позицій їжі;
- безкоштовне розповсюдження.

Слід зазначити, що розглянуті в попередньому розділі програми також включають в себе функцію трекінгу фізичної активності (включно з інтеграцією фітнес-браслетів та смарт-годинників) на додачу до догляду за харчуванням. Таке рішення виглядає органічним, але ускладнює застосунок, вимагаючи витрачати на розробку більше часу, а якщо розглядати застосунок як комерційний проект (або перспективу його перетворення у такий), то час розробника – це додаткові витрати. Виключення моніторингу фізичної активності дасть змогу заочно делегувати таку функцію програмам для фітнес-браслетів та смарт-годинників, або спеціалізованим спортивним застосункам, які можуть бути встановлені на смартфоні користувача. Також це усуває зайвий для деяких користувачів мікро контроль за раціоном.

Слід також сказати, що в перспективі додаток орієнтований на менш вимогливих користувачів, що зможуть користуватися додатком на безоплатній основі. Це значно розширить коло цільової аудиторії та допоможе слугувати цілі практичної користі для людей, дозволить будь-кому зі смартфоном завантажити застосунок і слідкувати за своїм харчуванням, а отже і за здоров’ям. Це означає,

що створення цього додатку покликане принести користь у вигляді збільшення частки здорового населення, незважаючи на кількість аудиторії.

РОЗДІЛ 2
АНАЛІЗ ТЕХНОЛОГІЙ ТА ВИБІР ІНСТРУМЕНТАРІЮ РОЗРОБКИ

2.1. Аналіз та вибір засобів розробки

В цьому розділі проведено невелике дослідження з метою обрати інструменти та технології, що стануть в нагоді при розробці програмного продукту. В якості таких згадані та описані: Dart, Flutter, Android Studio, Firebase та інші.

2.1.1. Dart

Dart – молода мова програмування загального призначення, розроблена компанією Google, представлена в 2011 р., а випущена вже в 2013 р. [1]. Спроектована для розробки серверних та клієнтських застосунків, зокрема у програмуванні для мобільних пристроїв та веб-програмуванні. Може використовуватись для розробки програм для систем Windows, MacOS та Linux, тобто для побудови desktop програм. Спочатку Dart позиціонувався як альтернатива JavaScript, але від цієї стратегії відмовилися.

Загалом представляє собою об’єктно-орієнтовану мову, що зазнала впливу інших, старіших мов: Smalltalk, Java, JavaScript, C, C#, Erlang, Strongtalk. Користувацькі пакунки, яких велика кількість, поширюються в репозиторії pub. Вони додаються в програму в якості бібліотек.

Синтаксис мови добре знайомий новачкам, що переходять з інших мов програмування, через його подібність на JavaScript, Java, C. Цікавою особливістю є підтримка як статичної, так і динамічної типізації. В процесі розробки тип змінної можливо вказати як var або dynamic, і це не буде помилкою, хоча в багатьох випадках в готових програмах слід уникати цього і вказувати

статичний тип. Підтримуються числові, рядкові типи, різноманітні хеш-коди, масиви і списки, типи дати і часу, а також регулярні вирази. Створення своїх типів також можливе. Важливим нюансом мови є те, що починаючи з версії 2.12 в Dart за замовчуванням діє null-safety. Тобто, типам, що не можуть приймати значення null, має бути призначене якесь значення до використання змінної, інакше при компіляції програми ми отримаємо помилку. Для ситуацій з null-безпекою використовується оператор ?, що повідомляє про те, що змінна даного типу може приймати null. Наприклад, тип String? є по суті nullable-двійником звичайного String. Також для null перевірок використовується оператор ?? (val1 ?? val2). Якщо значення val1 не null, оператор поверне саме значення val1. Якщо ж null, то val2. Існує і оператор !, що використовується для тих випадків, коли розробник впевнений, що змінна nullable-типу в певний момент часу точно не буде зберігати null (int? a = 1; int b = a!;). Даним оператором краще не зловживати. Широко використовується тернарний оператор ? : ([val1 - умова] ? [val2]: [val3]) [2]. Тернарна операція повертає другий або третій операнд: якщо умова дорівнює true, то повертається val2, а якщо умова дорівнює false, то val3 (рис. 2.1).

```dart
void main() {
  var bob = User('Bob', 40, true, ['Football', 'Skate']);
    //..name = 'Bob'
    //..age = 40;
  bob.info();

  var alex = User('Alex', 25, false, ['Basketball']);
  alex.info();
}

class User{
  String? name;
  int? age;
  bool? isHappy;
  List<String>? hobbies;

  User(this.name, [this.age, this.isHappy, this.hobbies]);

  void info(){
    var happy = isHappy! ? 'happy' : 'not happy';
    print('User $name is $age years old. He is $happy. His hobbies:');
    for (var el in hobbies!) {
      print(el);
    }
  }
}
```

Рисунок 2.1 – Приклад Dart коду

Dart швидко розвивається та набув значної популярності в 2019-2021 рр. Плагіни з впровадженням підтримки цієї мови існують в Visual Studio Code, IntelliJ Idea, Android Studio, Emacs.

2.1.2. Flutter

Але Dart ніколи не отримав би свою частку розробників, якби в 2017 році не була анонсована перша версія Flutter. Flutter – це фреймворк з відкритим кодом, що використовує Dart для побудови мобільних додатків (Android та IOS), а також веб-додатків [3]. Розробка компанії Google.

Особливість Flutter полягає у тому, що він дозволяє створювати native додатки для кожної з мобільних платформ, у веб-середовищі, і, у якості незвичного варіанту, навіть для десктоп-платформ. Рушій Flutter написаний на C++, а базова бібліотека – на Dart. Відрізняється від інших подібних фреймворків тим, що використовує Dart для побудови інтерфейсу, а не HTML, CSS чи JavaScript.

Увесь інтерфейс користувача складається з віджетів. Віджет у Flutter – це своєрідний будівельний блок, які можна поєднувати для отримання складніших віджетів (рис. 2.2). Це стосується і тексту, і графіки, і анімацій. При цьому Flutter дозволяє використовувати як віджети дизайну Google, так і дизайну Apple. Це сприяє написанню програми для обох платформ за один раз.

```dart
import 'package:flutter/material.dart';

void main() => runApp(MyApp());

class MyApp extends StatelessWidget{

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      theme: ThemeData(primarySwatch: Colors.amber),
      home: Scaffold(
        appBar: AppBar(
          title: Text('Learning App'),
          centerTitle: true,
        ),
        body: Row(
          mainAxisAlignment: MainAxisAlignment.spaceAround,
          children: [
            Column(
              mainAxisAlignment: MainAxisAlignment.end,
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                Text('Hello'),
                TextButton(onPressed: () {}, child: Text('Hello')),
                Text('Hello'),
                Text('Hello')
              ],
            ),
            Column(
              children: [
                Text('Hello'),
                TextButton(onPressed: () {}, child: Text('Hello'))
              ],
            )
          ],
```

Рисунок 2.2 – Приклад коду в Flutter

Варто додати, що Flutter, на рівні з Dart, має дружню до розробника документацію, офіційні відеоматеріали, що доповнюють документацію, та немалу спільноту. Значною перевагою можна вважати своєрідний симбіоз Flutter та Android Studio. При розробці на цьому фреймворку зручно використовувати функції Hot Restart/Hot Reload, що швидко перевантажують проект або окрему частину коду у випадку внесення змін [4].

Flutter був обраний як основний фреймворк для даного проекту через легке засвоєння та відгуки розробників, які підтверджують, що Flutter реально опанувати в стислі сроки. Так само, розробка на Flutter проходить швидше, у порівнянні з іншими засобами.

У Flutter є багато конкурентів, коли справа стосується розробки мобільних застосунків, найпопулярніші з них на даний час: Swift, Java, Kotlin, React Native [5]. Але для кожного з цих варіантів знайдеться свій контраргумент. Swift створений спеціально для розробки під IOS; Java та Kotlin прекрасні, але

вимагають тривалого опанування; React Native – схоже до Flutter рішення, але вимагає досвіду роботи з JavaScript, і цей варіант був відхилений автором [6]. У висновку, було прийняти рішення обрати Flutter і здобути нові практичні навички та досвід роботи з фреймворком.

З цієї причини вибір пав і на мову Dart, як невід'ємну частину Flutter. Використовувати іншу мову при роботі з Flutter неможливо. Не варто також забувати про свої переваги використання цієї мови програмування, що випливають з її особливостей та рис.

2.1.3. Android Studio

Android Studio – інтегроване середовище розробки, розроблене Google та випущене у 2014 р [7]. Один з найпопулярніших IDE для розробки мобільних додатків для операційної системи Android, а також її версій та дочірніх систем для смарт-годинників, телевізорів, окулярів і автомобільних систем. Побудований на базі іншого популярного середовища розробки IntelliJ Idea, і багато чим його нагадує.

Емулятор Android будь-якої версії API та різних моделей смартфонів, включаючи пристрої з різною роздільною здатністю та відношенням сторін екрану, значно прискорює розробку та тестування. Є також альтернативний спосіб тестування, коли проект запускається в браузері Chrome.

Android Studio пропонує:

- Android Virtual Device емулятор для запуску та тестування застосунків;
- обширну підтримку мов програмування;
- редактор макетів, в якому можна вільно перетаскувати елементи дизайну;
- збірку проектів за допомогою Gradle;

- швидкі виправлення та орієнтований на Android рефакторинг;
- знайомий інтерфейс для тих, хто вже працював з продуктами компанії JetBrains.

Стабільний, потужний продукт досить швидко привернув увагу мобільних розробників і став одним з основних IDE для них [8]. А популярність середовища розробки зумовило легкий пошук інформації по початковим налаштуванням та допомозі в роботі в Android Studio, що є вагомим плюсом. З огляду на вимоги до розробки проекту, Android Studio був визнаний оптимальним варіантом для даної роботи.

2.1.4. Cloud Firestore

Firestore – це основний сервіс компанії Firebase, придбаної Google в 2014 р. Firebase є платформою для розробки мобільних та веб-застосунків [9]. Надає декілька служб і рішень для розробки, серед яких Firebase Analytics, Firebase Cloud Messaging, Firebase Auth, Realtime Database (Cloud Firestore є її наступницею), Firebase Storage, Firebase Hosting, ML Kit.

Cloud Firestore – це гнучка хмарна NoSQL система керування базами даних, для серверних, веб- та мобільних розробок. Cloud Firestore синхронізує дані між клієнтськими додатками у реальному часі та навіть пропонує підтримку автономних операцій, зменшуючи залежність від наявності інтернету та його швидкості [10].

Структура даних Cloud Firestore складається з гнучких ієрархічних структур [11]. Інформація зберігається в документах, що в свою чергу організовані в колекції (рис. 2.3). Документи можуть містити складні вкладені об’єкти і колекції нижчого рівня. Вони підтримують різні типи даних, від рядків і чисел до посилань та вкладених об’єктів [12].

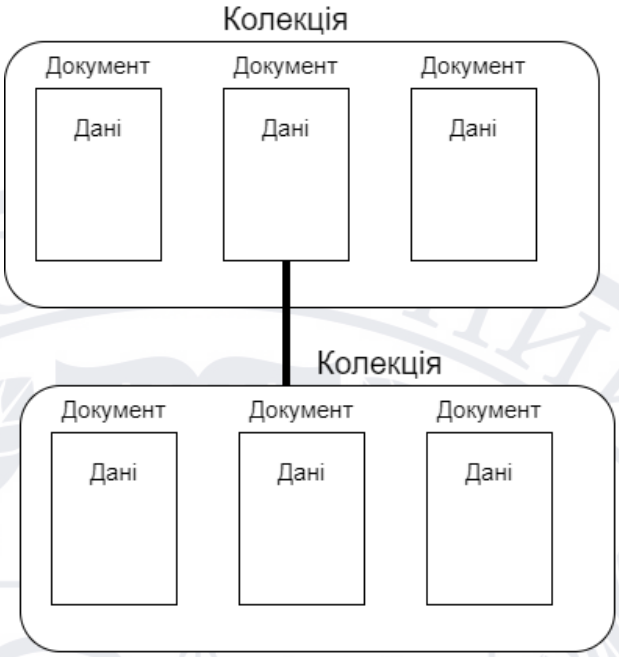


Рисунок 2.3 – Структура організації даних в Cloud Firestore

Доступ до Firestore можна отримати прямо через SDK, або REST чи RPC API. За допомогою запитів можна отримати дані на рівні документа або будь-якої вкладеної колекції. Підтримуються сортування, фільтрація та пагінація через курсори. А обробники у реальному часі допомагають витягнути та оновити лише ті дані, що потребують оновлення.

Пояснення вибору саме цієї платформи досить прості – це нескладна в опануванні хмарна NoSQL система керування базами даних, що легко інтегрується з Flutter SDK, і виконує усі поставлені задачі зі зберігання даних. Підходить як для нових проектів на етапі розробки, так і для готових продуктів з посиленими вимогами стабільності та захисту даних.

Інтеграції Flutter з рішеннями Firebase присвячений цілий сайт, що містить інформацію про офіційні Firebase пакунки для Flutter, огляд, початкову допомогу [13]. І найголовніше, присутність повної, дружньої до розробника, документації, присвяченій лише цій інтеграції. Кількість інформації та корисних посилань вражає. Такий вдалий і добре описаний симбіоз і став причиною використання Cloud Firestore в цьому проекті.

2.1.5. Figma

Було б несправедливо оминути увагою і Figma, що приніс багато користі цьому проекту. Figma – це графічний редактор, що доступний в інтернеті та як настільна програма [14]. Він надає можливість використовувати вектори для побудови користувацького дизайну. Широко застосовується як засіб для створення прототипів та дизайну веб-сторінок, веб-сервісів і програмного забезпечення. Запущений в 2016 році.

Figma виділяється також можливістю роботи в реальному часі з командою. Figma Community дозволяє дизайнерам публікувати свої роботи, напрацювання та елементи дизайну, надаючи іншим користувачам можливість переглядати та використовувати такі доробки з будь-якими змінами та без обмежень в своїх дизайнах [15].

Підтримує як імпорт файлів та зображень різного формату, так і експорт. Доробки зберігаються в хмарі, а кожна зміна зберігається миттєво, без потреби натискати на якісь кнопки.

Для створенню дизайну мобільного застосунку в рамках цієї роботи був обраний саме Figma, тому що він добре підходить під цілі розробки користувацького інтерфейсу, а також надає зручні фрейми для задання співвідношення сторін сучасних смартфонів. Додатково дуже корисною є функція перегляду дизайну в режимі презентації, використовуючи реальний вигляд смартфону, включаючи різноманітні вирізи екрану та заокруглення.

РОЗДІЛ 3
ОПИС ПРОЦЕСУ РОЗРОБКИ

3.1. Розробка дизайну

3.1.1. Вибір основних кольорів

Розробка будь-якого програмного забезпечення, а особливо мобільного застосунку, починається з уяви про те, як це буде виглядати. Разом з тим, необхідна міцна уявна прив’язка до якогось основного кольору. Автору одразу спав на думку рожевий колір. І для його використання в застосунку для слідкування за раціоном знайшлись одразу декілька вагомих причин, які не можна обійти стороною.

Перша з них – при огляді можливих конкурентів для додатку в жодного з них не мав значне місце рожевий колір. Використання кольору, що не прив’язаний до інших продуктів, дає змогу уникнути небажаних асоціацій та, на противагу, створити окрему когнітивну прив’язку, що, в теорії, позитивно позначиться на бажанні користуватись додатком або хоча б запам’ятати його інтерфейс краще [16].

Друга річ – цей колір добре поєднується з білим, що є найпопулярнішим кольором для читання та сприйняття інформації. Таке поєднання дасть змогу отримати декілька переваг, враховуючи екзотичність та яскравість розового та практичність білого. Рожевий колір століттями використовувався, щоб привернути увагу [17]. Багато з асоціацій з цим кольором пов’язані з екзотичними мотивами [18].

Третє – Існує декілька асоціацій, пов’язаних з рожевим, що пояснюють доцільність його використання в програмі, пов’язаній з харчуванням, їжею та

здоров'ям. Традиційно ми уявляємо здорових людей рожевощокими. Таке мислення спричинене тим фактом, що крізь білу шкіру видно червону кров у людей, в яких спостерігається швидкий кровообіг. Наочно це можна побачити, до прикладу, спостерігаючи за спортсменами під час тренувань. Або у людей в холодну погоду, коли організм задіює доступні природні механізми для підвищення температури тіла. Між тим, рожевий, у класичному представленні, якраз і є поєднаннями між білим та червоним [19, с.372]. Таким чином, використання рожевого кольору в інтерфейсі додатку буде спонукати користувача, простіше кажучи, думати про здоров'я, або про те, що даний додаток допоможе користувачу покращити своє здоров'я. На додачу, існують декілька англомовних виразів, що сприяють позитивному враженню про рожевий колір. Наприклад, вираз «in the pink» означає «бути в гарній формі, в доброму здоров'ї, в гарних кондиціях». Або «tickled pink», що має значення «бути надзвичайно задоволеним».

Рожевий може асоціюватись з кондитерськими виробами та солодкими напоями. Це теж грає на руку функціям цього кольору в нашому випадку, оскільки дозволяє використати «солодкий» колір в програмі, де на першому плані стоїть здорове харчування, традиційно не асоційоване з солодким смаком. Користувач тим самим розглядає концепцію здорового харчування як щось більш привабливе та солодке, смачне.

Врешті-решт, було прийнято рішення застосувати рожевий колір, що має HEX-значення #E8909C. Такий відтінок близький до «морського рожевого» та має невелику контрастність, що добре впливає на комфорт зору. Як додаткові кольори було обрано білий та сірий, що непогано поєднуються з рожевим та використовувались як кольори фону та тексту. В окремих елементах присутній сіро-рожевий колір – там, де автор бажає виділити елемент від як від рожевого, так і від білого.

Слід уточнити, що таку увагу кольорам в цій роботі було приділено через бажання збільшити її теоретичну та практичну цінність. Автор вважає, що

індикатором практичної цінності таких прикладних досліджень є їх наближеність до реального життя та орієнтація на кінцевого споживача; в загальному значенні – на людину, а у вузькому, в цьому випадку, – на користувача смартфону, що ставить за мету покращити якість свого життя через оптимізацію раціону. Приймаючи до уваги той факт, що людина більшу частину інформації сприймає саме візуально, стає очевидним велике значення оформлення користувацького інтерфейсу та, відповідно, уважний підхід до вибору належної кольорової гамми.

3.1.2. Розробка макету

Як вже зазначалося в розділі 2, для побудови макету та визначення загальних рис дизайну був використаний сервіс Figma. Будування прототипів у Figma зазвичай поділяється на розробку дизайну окремих сторінок або екранів. Такий підхід є зручним як для дизайнера, так і для розробника. Автор виконує обидві функції.

В якості певної рамки для дизайну, що визначила відношення сторін, був узятий iPhone 11 Pro Max, як представник популярної лінійки смартфонів від Apple. Було прийнято рішення почати розробку дизайну з головної сторінки, яка, за попереднім рішенням, буде відкриватись одразу після запуску застосунку (рис. 3.1).

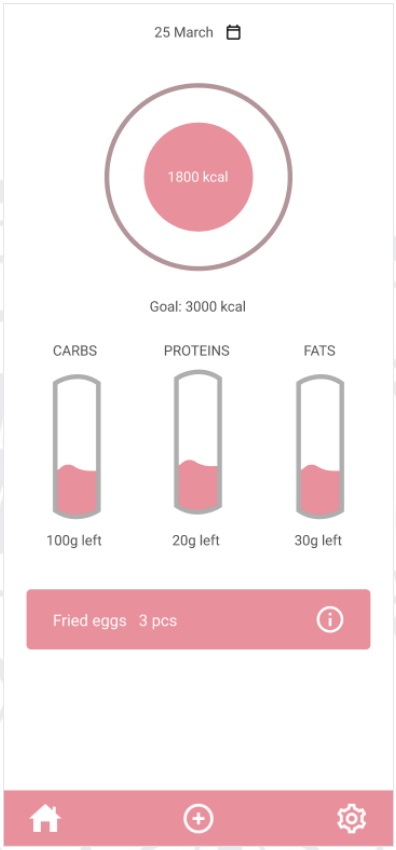


Рисунок 3.1 – Дизайн головної сторінки застосунку

На рисунку бачимо головні структурні особливості не тільки першого екрану, а й всього додатку. Слід відзначити:

1. Використання в якості меню навігації нижнього меню. Це знайомий для мобільних користувачів прийом, до якого забезпечений мінімальний час адаптації. На рис. 3.2 можна побачити, що перша іконка заповнена білим кольором, інші – ні. Це спеціальне рішення, що дає зрозуміти користувачу, на якій саме сторінці він зараз знаходиться. Відповідно, заповнена кольором іконка є індикатором поточної сторінки.

Рисунок 3.2 – Нижнє меню навігації

2. Іконка календаря у верхній частині та відображення поточної дати – слугує цілі відображення інформації на сторінці саме за поточний

день (рис. 3.3). В перспективі при натисканні на іконку можна буде обрати дату.

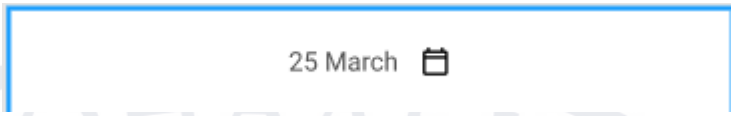


Рисунок 3.3 – Елемент дизайну: поточна дата та іконка календаря

3. Велике коло у центральній частині – це абстрактна ціль, що представляє ціль споживання певної кількості енергії як добову норму для користувача (в цьому випадку 3000 ккал). Коло меншого розміру всередині більшого – представляє собою поточну спожиту енергіжю (1800 ккал на малюнку). Таким чином, дві фігури співвідносяться між собою графічно (рис. 3.4). Поки розмір більшого кола не змінюється, менше буде збільшуватись (починаючи від повної відсутності на екрані) пропорційно до вжитих ккал.

Рисунок 3.4 – Елемент дизайну: графічне представлення кількості спожитих ккал

4. Далі в центральній частині – графічне зображення кількості спожитих поживних речовин (вуглеводнів, протеїну та жирів) за поточну добу (рис. 3.5). Така концепція схожа на концепцію діграми, коли в пропорційній залежності від вжитих елементів буде збільшуватись висота заповнення відповідного стовпця.

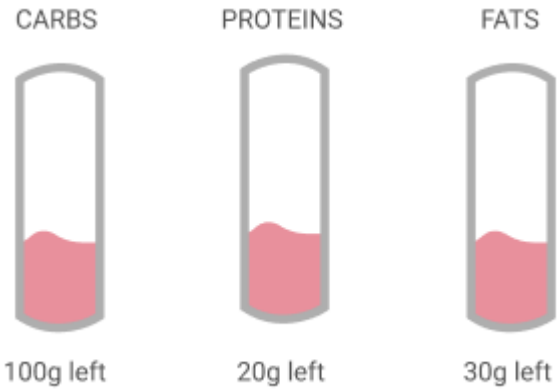


Рисунок 3.5 – Елемент дизайну: графічне представлення кількості спожитих поживних речовин

5. Останній елемент, що знаходиться в нижній частині екрані, та який можна пролистувати – це список з вжитих за поточну добу страв, що складається з карточок з назвою, кількістю та кнопкою отримання інформації (рис. 3.6).

Рисунок 3.6 – Елемент дизайну: картка однієї вжитої страви. З таких карток формується список, який можна пролистувати

В процесі розробки дизайну виникли декілька ідей для побудови другого варіанту головної сторінки. Такі ідеї відображені у наступному макеті (рис. 3.7):

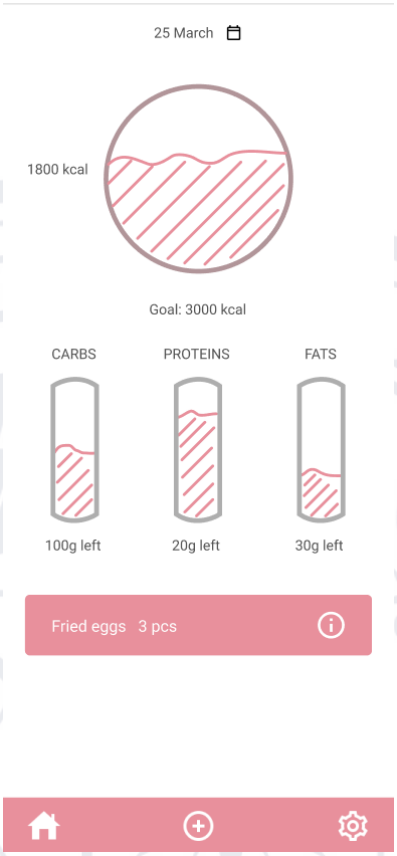


Рисунок 3.7 – Другий варіант дизайну головної сторінки застосунку

Зі змін можна відзначити:

1. Інший дизайн елементу, що відповідає за вивід інформації про добове поточне споживання кілокалорій: тепер це лише зовнішнє велике коло, а висота заповнення вмісту буде пропорційно залежати від відношення поточного споживання до цільового показника за добу.
2. Схожим чином змінений зовнішній вигляд елементу, що відповідає за вивід інформації про споживання поживних речовин за поточну добу.

Розробку дизайну було продовжено з розробки другої сторінки застосунку, яка доступна при натисканні на значок «+» в нижньому меню навігації. Головною функцією цією сторінки є пошук їжі в базі. Додатковою функцією є відображення вжитих за поточну добу позицій, що вже були внесені на поточний момент (рис. 3.8).

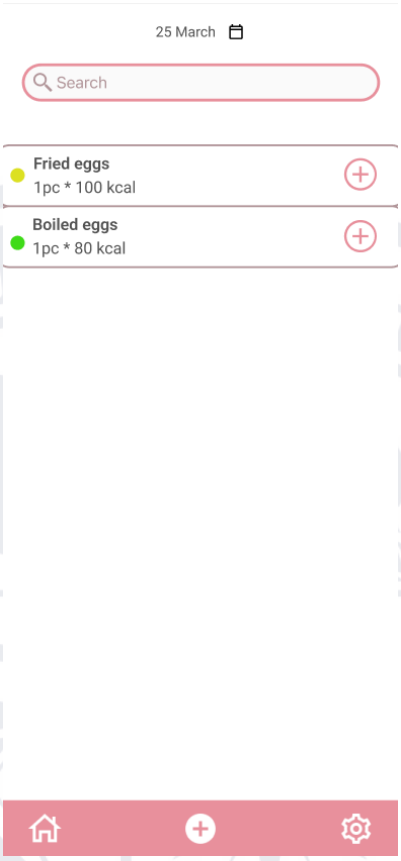


Рисунок 3.8 – Дизайн сторінки застосунку, яка відповідає за додавання нової їжі до переліку вжитої

Сторінка розбита на наступні елементи:

1. Зверху ми бачимо поточну дату разом з іконкою календаря. Цей елемент виконує таку ж саму функцію, що і відповідний елемент з головної сторінки, зокрема відповідає за відображення вжитої їжі за добу, що обрана.
2. Одразу після розташована стрічка – в ній можна шукати їжу або страву для внесення (рис. 3.9). Внесена їжа додається до переліку вжитих нижче.

Рисунок 3.9 – Елемент дизайну: пошукова стрічка

3. Далі йде перелік вжитої їжі. Його можна гортати. Складається з карток, де вказана така інформація: назва, кількість та вміст кілокалорій. При натисканні на кнопку «+» з'являється віджет, що дозволяє користувачу зменшити або збільшити кількість. Віджет містить інформацію про назву їжі, кількість та вміст кілокалорій (рис. 3.10). Кнопка «+» додає одну штуку, порцію або 100 грам (в залежності від їжі), кнопка «–» віднімає. Інший вміст сторінки стає темнішим для досягнення ефекту фокусу, а тіні сигналізують, що спливаючий віджет ніби знаходиться над усією сторінкою (рис. 3.11).

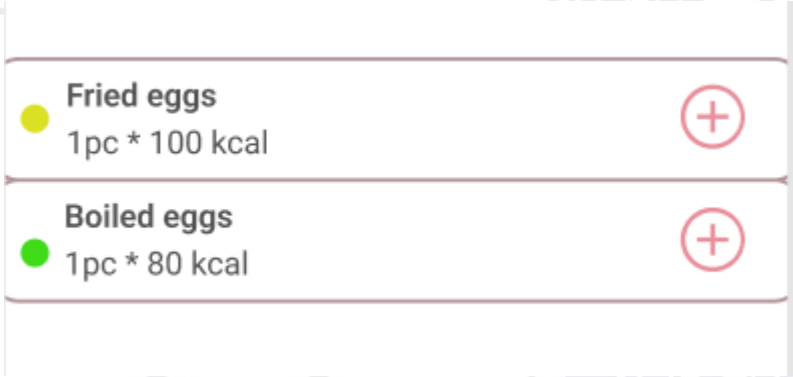


Рисунок 3.10 – Елемент дизайну: список вжитих позицій їжі з можливістю регулювати кількість

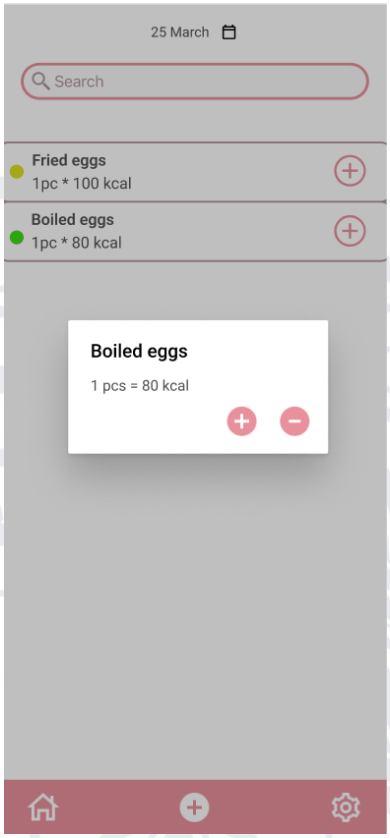


Рисунок 3.11 – Елемент дизайну: спливаючий віджет з назвою страви та можливістю регулювання кількості

Розробка дизайну застосунку продовжилась з третьої сторінки, що відкривається при натисканні на шестерню в нижньому меню навігації. Ця сторінка відображає доступні категорії налаштувань і відомості про додаток, а також містить навігацію для доступу до профіля користувача (рис. 3.12).

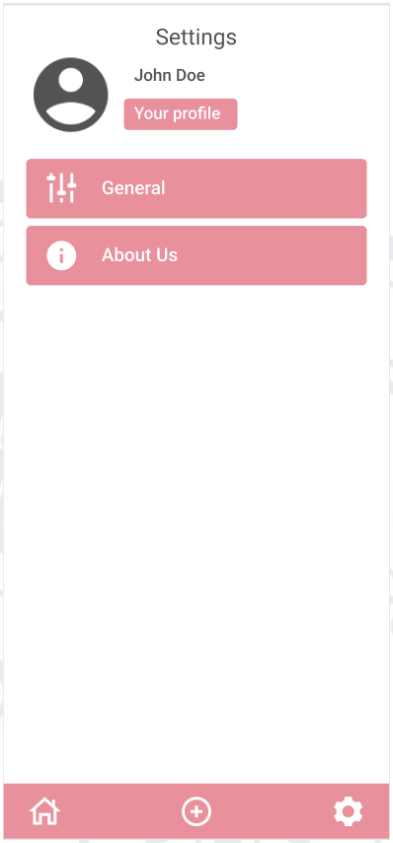


Рисунок 3.12 – Дизайн сторінки застосунку, яка містить навігацію до налаштувань та відомості про профіль і застосунок

Елементи сторінки:

1. Зверху можна зустріти іконку аватару користувача. Користувач у перспективі зможе її змінити на будь-яке зображення. Справа від аватару – ім’я користувача (що теж редагується), і кнопку, при натисканні якої відбувається перехід до відомостей про профіль (рис. 3.13).

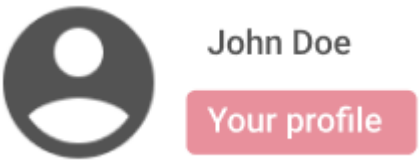


Рисунок 3.13 – Елемент дизайну: аватар користувача, ім’я користувача і кнопка переходу до інформації про користувача

2. Далі розмістились кнопки, що ведуть до відповідних сторінок. «General» веде до загальних налаштувань, а «About Us» – до

інформації про додаток (рис. 3.14). У разі необхідності кількість кнопок буде розширена в процесі розробки.

Рисунок 3.14 – Елемент дизайну: навігація на сторінці налаштувань

На цьому розробку дизайну завершено. Однак це лише прототип, який має направляти розробника, натомість певні елементи можуть бути змінені, видалені або додані в силу різних причин, в тому числі на користь поліпшення користувацького інтерфейсу, зміни складності розробки або оптимізації швидкодії.

3.2. Розробка програми

Розробка програми традиційно починається зі створення проекту в інтегрованому середовищі розробки, в цьому випадку – Android Studio. Після вказання імені проекту та необхідних попередніх налаштувань формуються файли проекту. До них входять різноманітні файли конфігурації, тека для написання тестів, окремі папки для спеціальних файлів під платформи IOS або Android [20]. Також в таких папках можна зберігати, до прикладу, зображення, якщо ми хочемо конкретне зображення для конкретної платформи. Це може бути доречним, оскільки різні платформи використовують дещо різні концепції дизайну. В папці «lib» зберігається головний файл програми «main.dart» (рис. 3.15).

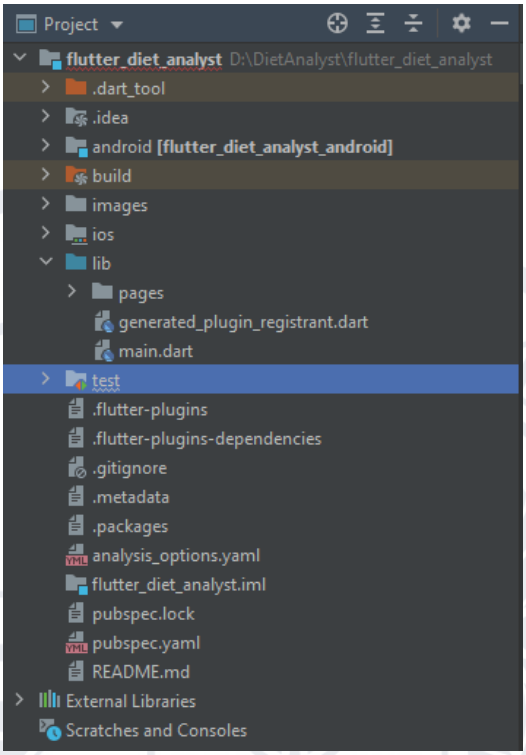


Рисунок 3.15 – Список файлів проекту в Android Studio

Заздалегідь створимо папку «pages», в якій будуть зберігатися майбутні сторінки програми. В загальній папці «images» (для обох платформ) будуть зберігатись зображення, якщо такі знадобляться в процесі розробки. Приготування завершено, тепер проект готовий для початку роботи.

На початку кожного .dart файлу імпортуються необхідні пакунки за допомогою ключового слова import. Користувацькі пакунки для правильного підключення та завантаження вимагають ще невеличкої зміни файлу «pubspec.yaml». Там же можна контролювати, яку версію пакунку завантажувати. Якщо перед назвою пакунку версія не вказана, автоматично завантажується остання версія. Також є можливість здійснити ще декілька налаштувань, зокрема підключити необхідні шрифти. Загалом, «pubspec.yaml» містить метадані про проект, які необхідно знати інструментам Dart і Flutter (рис. 3.16) [21]. Але треба бути обережним, тому що файл чуттєвий до пробілів.

```
# Dependencies specify other packages that your package needs in order to work.
# To automatically upgrade your package dependencies to the latest versions
# consider running `flutter pub upgrade --major-versions`. Alternatively,
# dependencies can be manually updated by changing the version numbers below to
# the latest version available on pub.dev. To see which dependencies have newer
# versions available, run `flutter pub outdated`.
dependencies:
  settings_ui:
  textfield_search:
  cloud_firestore:
  firebase_core:
  flutter:
    sdk: flutter
```

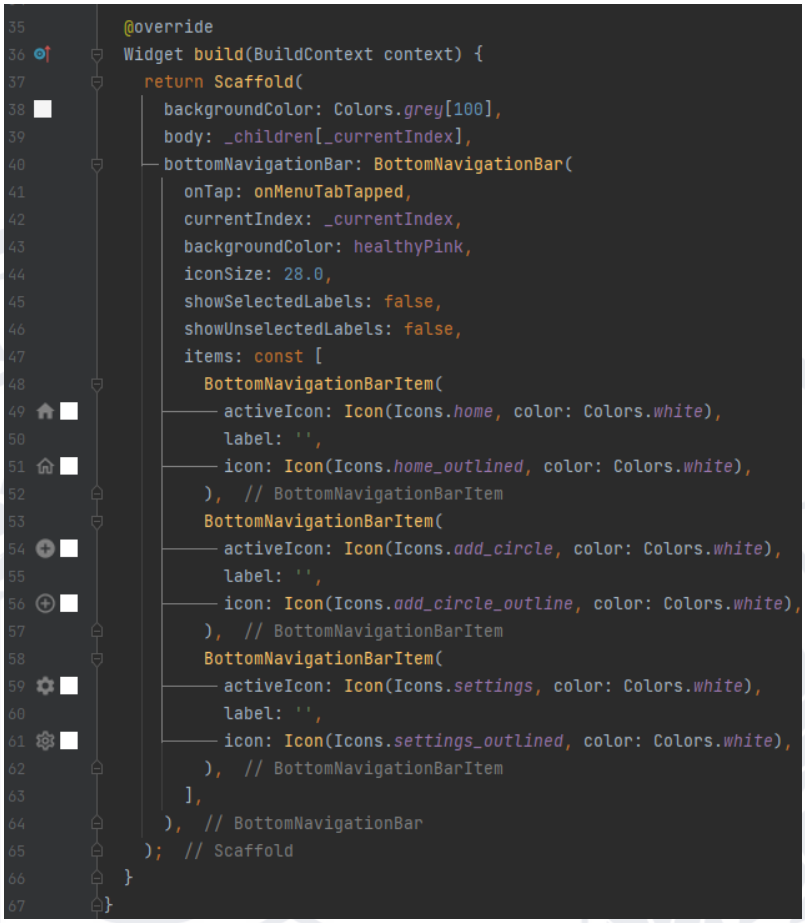
Рисунок 3.16 – фрагмент файлу «pubspec.yaml»

Найважливішим пакунком є «material.dart». Він дозволяє використовувати віджети стилю «Material Design» [22]. Це основний стиль системи Android. В головному файлі відбувається також підключення усіх сторінок програми (рис. 3.17). Окремо слід виділити підключення пакунку з ядром Firebase – це необхідно для ініціалізації Firebase на цій сторінці. Функція «main()» – асинхронна (позначена словом «async»). Це означає, що деякі процеси в функції відбуваються синхронно, а деякі – асинхронно. Для позначення асинхронних процесів використовується «await». Всього у функції три дії: ми впевняємось в тому, що WidgetsFlutterBinding (клей між шаром віджетів і рушієм Flutter) запущений або запускаємо його, далі відбувається ініціалізація Firebase, використовуючи вручну введені дані, і прикріплюємо наші віджети до екрану, використовуючи функцію «runApp()». В цій же функції визначаємо шляхи до інших сторінок програми. Початковою сторінкою буде сторінка «home.dart».

```dart
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';
import 'package:flutter_diet_analyst/pages/home.dart';
import 'package:flutter_diet_analyst/pages/main_screen.dart';
import 'package:flutter_diet_analyst/pages/settings.dart';
import 'package:flutter_diet_analyst/pages/add_meal.dart';


void main() async {
  WidgetsFlutterBinding.ensureInitialized();

  await Firebase.initializeApp(
    options: const FirebaseOptions(
      apiKey: "[redacted]5mI6DyhOd0KSzS55pbBfEcuuiFjJE",
      appId: "1:12639479614",
      messagingSenderId: "12639479614",
      projectId: "dietanalyst-fa6ba",
    ),
  );

  runApp(MaterialApp(
    title: 'Diet Analyst',
    theme: ThemeData(
    ),  // ThemeData
    initialRoute: '/',
    routes: {
      '/settings': (context) => SettingsMenu(),
      '/': (context) => Home(),
      '/mainscreen': (context) => MainScreen(),
      '/addmeal': (context) => AddMeal(),
    },
  ));  // MaterialApp
}

```

Рисунок 3.17 – Код файлу «main.dart»

Перейдемо до файлу «home.dart». В цьому файлі було реалізовано нижнє меню навігацію через віджет «BottomNavigationBar» (рис. 3.18). За допомогою цього віджету можна легко побудувати нижнє навігаційне меню та вказати шляхи до потрібних сторінок. Кожному елементу меню відповідає індекс (починаючи від 0), і перехід до іншої сторінки відбувається через індекс в масиві. «BottomNavigationBar» відноситься до готових рішень, які полегшують та пришвидшують розробку програми через надання зручних інструментів для створення елементів користувацького інтерфейсу [23].



```
@override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: Colors.grey[100],
    body: _children[_currentIndex],
    bottomNavigationBar: BottomNavigationBar(
      onTap: onMenuTabTapped,
      currentIndex: _currentIndex,
      backgroundColor: healthyPink,
      iconSize: 28.0,
      showSelectedLabels: false,
      showUnselectedLabels: false,
      items: const [
        BottomNavigationBarItem(
          activeIcon: Icon(Icons.home, color: Colors.white),
          label: '',
          icon: Icon(Icons.home_outlined, color: Colors.white),
        ),  // BottomNavigationBarItem
        BottomNavigationBarItem(
          activeIcon: Icon(Icons.add_circle, color: Colors.white),
          label: '',
          icon: Icon(Icons.add_circle_outline, color: Colors.white),
        ),  // BottomNavigationBarItem
        BottomNavigationBarItem(
          activeIcon: Icon(Icons.settings, color: Colors.white),
          label: '',
          icon: Icon(Icons.settings_outlined, color: Colors.white),
        ),  // BottomNavigationBarItem
      ],
    ),  // BottomNavigationBar
  );  // Scaffold
}
}
```

Рисунок 3.18 – Фрагмент файлу «home.dart»

Файл «main_screen.dart» являє собою безпосередньо сторінку, яку побачить на екрані користувач, при цьому сторінка буде першою, що завантажується при запуску програми. Найскладніші та найцікавіші моменти, що виникли при розробці сторінки «main_screen.dart»:

1. програмна реалізація стовпців для відображення кількості вжитих поживних речовин;
2. забезпечення читання з бази даних списку вжитої їжі за день;
3. читання показників суми вжитих речовин та калорій за день.

Окремо слід пояснити, як були структуровані дані в базі Cloud Firestore від Firebase. Оскільки дана база типу NoSQL, користувач бази майже не обмежений у виборі структури [24]. Нормалізація даних для NoSQL бази – поняття не обов’язкове, тому що всі дані зберігаються за принципом документів всередині колекцій. Дані можуть зберігатися практично як завгодно, первинних та вторинних ключів просто не існує, дані можуть дублюватись. Вагомою умовою

є і те, що користування сервісом хмарного зберігання даних стає платним, якщо вичерпуються певні ліміти на кількість читань і записів [25]. Автор витратив певну кількість часу на пошук та розробку правильної структури, що відповідала б таким вимогам:

- невелика кількість запитів на читання з бази даних в процесі користування додатком;
- невелика кількість запитів на запис в базу даних в процесі користування додатком;
- стабільна робота та стійкість до дій, що в перспективі порушують правильну роботу бази.

Очевидно, що у випадку демонстрації NoSQL-структури табличне представлення даних неприйнятне. Але повну інформативність щодо структури може надати схема (рис. 3.19).

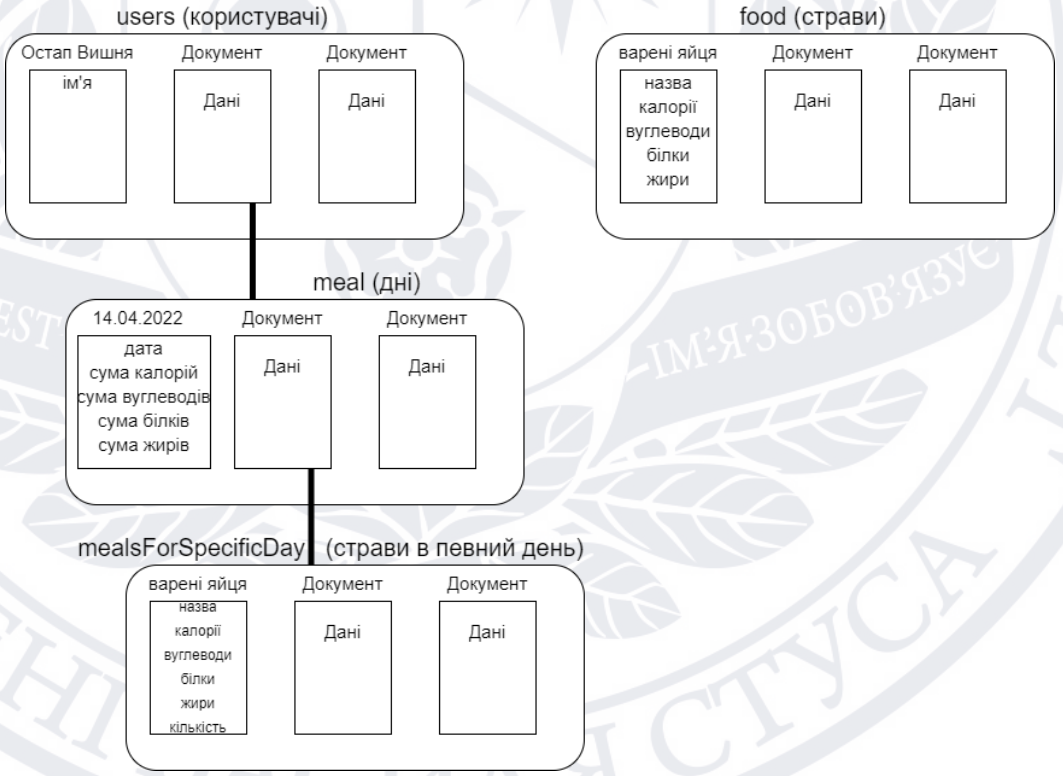


Рисунок 3.19 – Схема структури даних в Firebase

Таким чином, існують дві колекції верхнього рівня: users (користувачі) та food (страви). В колекції з користувачами зберігаються користувачі та їх імена. Кожен документ має унікальну назву. В колекції зі стравами зберігаються страви

та їжа. Кожен такий документ містить інформацію про назву, кількість калорій (1шт або 100 грам), кількість вуглеводів, кількість жирів, кількість білків.

В кожному документі з колекції користувачів зберігається вкладена колекція meal, що відповідає за окремі дні. Тобто кожен документ в цій вкладеній колекції відповідає за один день. Такий день містить інформацію про дату, суму калорій за день, а також білків, жирів і вуглеводів.

Документи з колекції днів містять вкладені колекції mealsForSpecificDay (страви в певний день). Така колекція створена з огляду на те, що потрібно зберігати спожиті за день користувачем страви в зручному вигляді, і як найзручніший було обрано саме колекцію страв у певний день. Документи в такій колекції зберігають інформацію, що дуже схожа на ту, що зберігається у документах колекції food (страви), але на додачу містить відомості про кількість спожитих порцій.

Після роботи з Cloud Firestore в Firebase загалом лишається гарне враження про якість, комфорт, та функціональність. Зручно не тільки працювати з хмарною базою, але і мати поруч такий функціонал як Cloud Functions та Cloud Authentication, що легко під’єднуються до інших рішень в сімействі Firebase. Враховуючи гнучку та зрозумілу систему надання платних послуг, можна користуватись сервісом безкоштовно протягом всього періоду розробки. Отже, Firebase можна порекомендувати розробникам як власних проектів, так і професійних рішень [26]. До того ж, завдяки зрозумілому інтерфейсу опанувати його швидко може як новачок, так і досвідчений розробник (рис. 3.20).

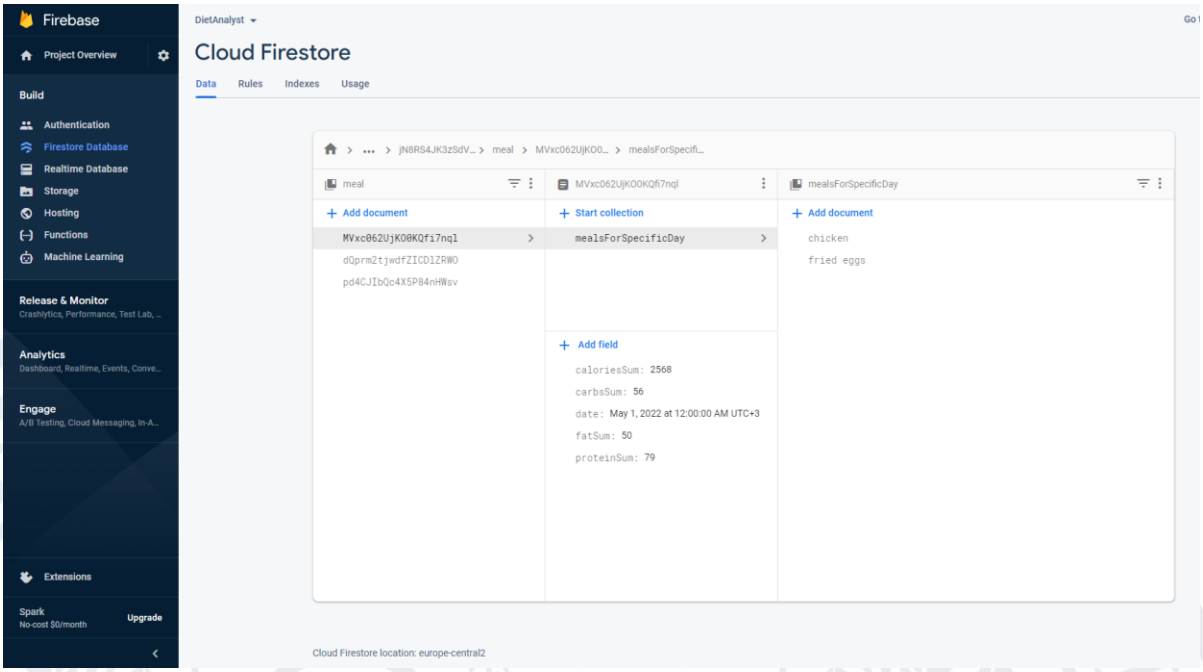


Рисунок 3.20 – Вигляд бази даних на веб-сайті

Повертаючись до розробки файлу «main_screen.dart», стовпці для відображення кількості вжитих поживних речовин, розроблені етапі дизайну, виявились занадто складними для реалізації. Спроби намалювати їх за допомогою класу CustomPainter [27], перед цим розробивши фігури візуально в програмі FlutterShapeMaker [28] та експортувавши звідти необхідний код, виявилися невдалими. Складність фігур та необхідність контролювати їх висоту в подальшому в коді та правильно відображати зумовили зависоку складність реалізації. Було прийнято важливе рішення досягти більшої простоти цих фігур та зберегти привабливість, не забуваючи про необхідність відповідати загальним засадам дизайну.

Рішенням стало відмовитися від сірої зовнішньої границі та вигляду «речовини в пробірці», і просто малювати прямокутник із заокругленими кутами, змінюючи висоту відповідно до показників з бази даних.

Наступним моментом, на якому варто зупинитися, є забезпечення читання з бази даних списку вжитої їжі за день. Це було реалізовано за допомогою віджету FutureBuilder [29]. FutureBuilder приймає дані від об’єкту Future (властивість future), та створюється на основі цих даних (властивість builder). В даному випадку віджет приймав знімок даних про колекцію страв за поточний день

(mealsForSpecificDay) і будував карточку (віджет Card) з назвою їжі, показником калорій та кількістю (рис. 3.21). Кількість карточок дорівнює кількості позицій їжі.

```
FutureBuilder<QuerySnapshot>(
  future:
  getTheUsersSpecificDay().then((value) {
    var day = value.docs.first.reference.id;
    return firestore
        .collection('users')
        .doc('jN8RS4JK3zSdVJNoReLR')
        .collection('meal')
        .doc(day)
        .collection('mealsForSpecificDay')
        .get();
  }),
  builder:
      (BuildContext context, AsyncSnapshot<QuerySnapshot> snapshot) {
    if (snapshot.hasError) return Text('Error: ${snapshot.error}');
    if (snapshot.connectionState == ConnectionState.waiting) {
      return const Center(child: Text('Loading...'));
    }
    else {
      return Column(
        children:
        snapshot.data!.docs.map((DocumentSnapshot document) {
          return Card(
            elevation: 3.0,
            shadowColor: Color(0xFFB19599),
            shape: RoundedRectangleBorder(
              borderRadius: BorderRadius.circular(10.0),
            ),  // RoundedRectangleBorder
            margin: EdgeInsets.fromLTRB(15.0, 0, 15.0, 10.0),
            child: ListTile(
              title: Text(document['name'],
                style: TextStyle(fontSize: 17, fontWeight: FontWeight.w500),),  /
              subtitle: Text('${document['quantity']} pcs = '
                  '${document['calories'.toString()]*document['quantity']} kcal',
                style: TextStyle(fontSize: 15),),  // Text
              dense: true,
```

Рисунок 3.21 – Фрагмент коду, що використовувався для отримання списку їжі за поточний день

FutureBuilder, на якому варто зупинитись окремо, є одним з найважливіших віджетів для отримання даних з бази, дозволяючи одним запитом отримати дані в асинхронному режимі. Робота з асинхронністю в Dart вимагає деяких навиків, а також підключення пакунку «dart:async». Крім властивостей future та builder, існує initialData, яка надає можливість віджету будуватись на основі знімку даних, визначеного заздалегідь. Якщо код builder будується швидше, ніж отримуються дані у future, використовується initialData.

Далі потрібно було зчитати дані для отримання показників вжитих калорій та поживних речовин за день. Поставлене завдання було виконано схожим чином з попереднім за допомогою FutureBuilder, в який передавались дані про поточний день в базі даних. Згодом, на етапі будування віджету, отримані показники

записувались у змінні, які використовувались для відображення на екрані текстових і візуальних даних (рис. 3.22).

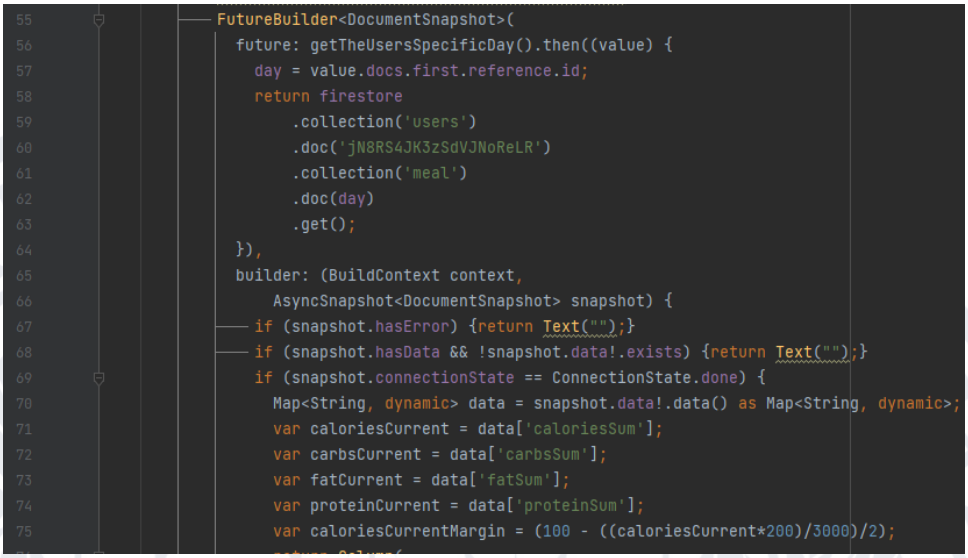


Рисунок 3.22 – Фрагмент коду, що використовувався для отримання сумарних показників спожитих речовин та енергії за поточний день

Перед будуванням вищезазначеного віджета також виконувалась функція getTheUsersSpecificDay() (рис. 3.23). В цій функції виконується запит до бази даних з пошуком документів в колекції meal для користувача з певним id. Документи мають відповідати поточній даті. Слід сказати, що кожному дню має відповідати лише один документ. Після виконання повертається об’єкт типу Future<QuerySnapshot<Map<String, dynamic>>>, що є об’єктом Future списку знімків даних, які відповідають знайденим документам.

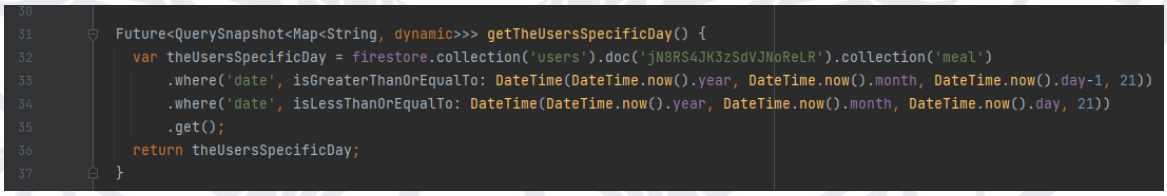


Рисунок 3.23 – Код функції getTheUsersSpecificDay()

В результаті отримано такий зовнішній вигляд головної сторінки (рис. 3.24):

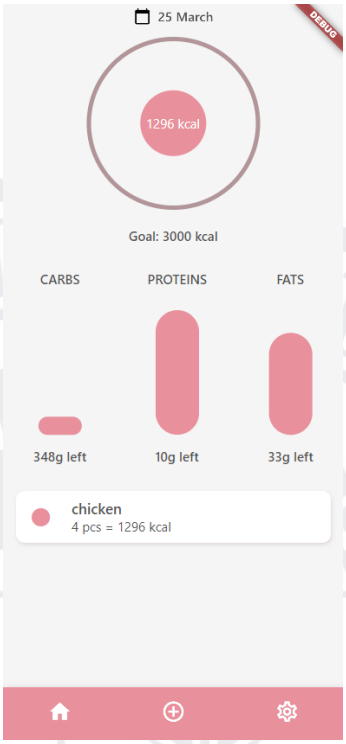


Рисунок 3.24 – Зовнішній вигляд сторінки «main_screen.dart»

Що стосується сторінки «add_meal.dart» (далі – іноді буде використовуватись «сторінка додавання їжі»), яка доступна по натисканню іконки плюса в нижньому меню, головними завданнями були:

1. забезпечення читання з бази даних списку вжитої їжі за день (аналогічна вимога, якщо порівняти з головною сторінкою);
2. поле пошуку з функцією пошуку страв в базі даних;
3. додавання їжі в список вжитої за день.

Перша задача була виконана з використанням коду, аналогічного тому, що міститься на головній сторінці. Цілі цих фрагментів користувацького інтерфейсу та їх смислове навантаження повністю співпадають.

Вирішення другої задачі заслуговує детальнішого опису. Запит до бази даних був побудований таким чином, щоб забезпечити можливість пошуку в реальному часі, «живого пошуку». При вводі часткового текстового запиту відображаються результати, що відповідають цілим словам. Загалом такий миттєвий пошук доцільно реалізовувати за допомогою віджета StreamBuilder [30].

StreamBuilder являє собою рішення, близьке до FutureBuilder, тільки в якості даних використовується не об’єкт Future, а моментальні знімки даних – об'єкт Stream (рис. 3.25). Відповідно, замість властивості future присутня властивість stream. Варто додати, що об'єкти Stream – теж результати асинхронних обчислень.

Рисунок 3.25 – Фрагмент коду, що використовувався для пошуку страв у базі даних

Варто уваги, що якщо поле пошуку не пусте, програма повертає віджети включно зі StreamBuilder. Якщо поле пусте, повертаються віджети разом з FutureBuilder, який забезпечує виведення на екран списку їжі за сьогодні, вже доданої на бази. Таким чином, сторінка додавання їжі виграла у функціональності, адже можна не тільки додавати, а й переглядати додані позиції.

І, наостанок, одна з найважливіших функцій – додавання їжі до бази. Запис відбувається із залученням методу runTransaction() класу FirebaseFirestore. Цей метод виконує запис та читання з бази даних атомарно. Тобто, в наборі операцій або всі виконуються успішно, або не виконується жоден. Транзакція і є таким набором операцій, причому може застосовуватись до одного або більше документів.

В нашому випадку виконуються дві транзакції. Перша транзакція записує до списку вжитої їжі позицію (користувач обирає кількість), або оновлює її. Друга транзакція розраховує сумарні показники по калоріям і по речовинам, та оновлює ці дані в документі, що відповідає певному дню (рис. 3.26).

```
      firestore.runTransaction((transaction) async {
        DocumentSnapshot foodSnap = await transaction.get(document.reference);
        transaction.set(rofl!.doc('${foodSnap['name']}'), {
          'quantity': _mealCount,
          'calories': foodSnap['calories'],
          'carbs': foodSnap['carbs'],
          'fat': foodSnap['fat'],
          'protein': foodSnap['protein'],
          'name': foodSnap['name'],
        });
      });
    }).then((value) {
      firestore.runTransaction((transaction) async {
        DocumentSnapshot daySnap = await transaction.get(dayRef!);
        DocumentSnapshot foodSnap = await transaction.get(document.reference);
        DocumentSnapshot mealSnap = await transaction.get(rofl!.doc('${foodSnap['name']}'));

        final cal = mealSnap['calories'];
        final carbs = mealSnap['carbs'];
        final fat = mealSnap['fat'];
        final protein = mealSnap['protein'];
        final qty = mealSnap['quantity'];

        final calSum = (cal*qty) + daySnap['caloriesSum'];
        final carbsSum = (carbs*qty) + daySnap['carbsSum'];
        final fatSum = (fat*qty) + daySnap['fatSum'];
        final proteinSum = (protein*qty) + daySnap['proteinSum'];

        await transaction.update(dayRef!, {
          'caloriesSum': calSum,
          'carbsSum': carbsSum,
          'fatSum': fatSum,
          'proteinSum': proteinSum,
        });
      });
```

Рисунок 3.26 – Фрагмент коду, що використовувався для запису страв у базу даних

До речі, користувач додає або віднімає їжу (в штуках або грамах) в спливаючому вікні. В результаті отримано сторінку такого вигляду (рис. 3.27):

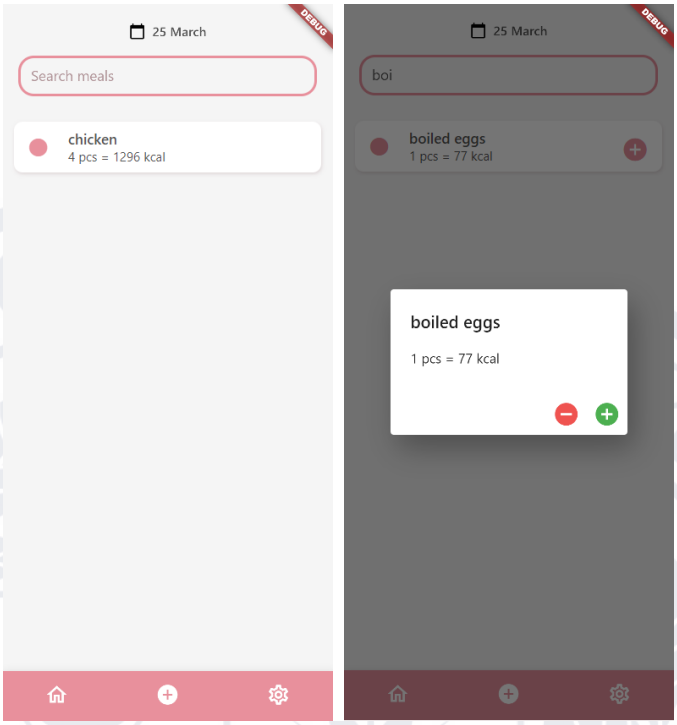


Рисунок 3.27 – Зовнішній вигляд сторінки додавання їжі

Як незамінну частину усіх мобільних додатків, було створено сторінку налаштувань під назвою «settings.dart». На цій сторінці було розміщено навігаційні елементи для переходу на інші сторінки, що містять певні категорії налаштувань. Також було розташовано кнопку для переходу у профіль користувача і його аватар (рис. 3.28).

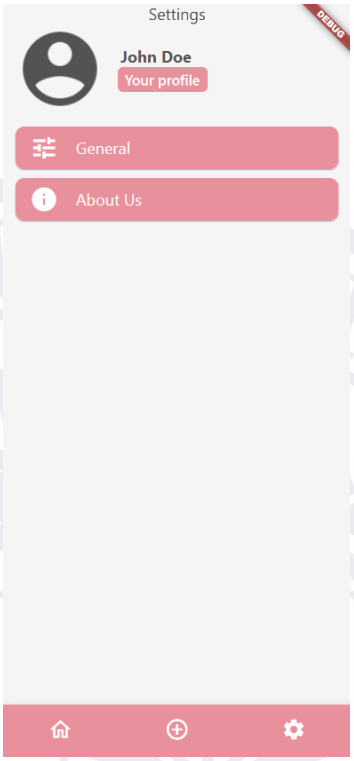


Рисунок 3.28 – Зовнішній вигляд сторінки налаштувань

ВИСНОВКИ

В ході даної роботи було виконано такі завдання:

- проаналізовано існуючі застосунки, що надають засоби та можливості для персонального нагляду за раціоном (HealthifyMe, Lifesum, Foodvisor, FatSecret); отримано детальне уявлення про загальний стан ринку мобільних додатків в сфері підтримки здоров'я і харчування; уточнено постановку задачі та сформульовано вимоги до програмного продукту;
- зроблено висновки з отриманого аналізу та обрано необхідні засоби та інструменти для розробки нового продукту (Dart, Flutter, Cloud Firestore, Android Studio); обґрунтовано доцільність їх використання у відповідності до вимог;
- розроблено макети дизайну та на їх основі програму для смартфонів, що може успішно та в необхідному обсязі задовольнити потребу в моніторингу власного раціону в зручному вигляді.

Знання та розробки, отримані здобувачем у ході виконання бакалаврської роботи, можуть бути використані для покращення програмного продукту, що розглядався, а також для створення інших додатків, що несуть користь суспільству. Автор вважає, що враховуючи орієнтацію дослідження на практичні розробки для створення користувацького додатку на підтримку здоров'я і харчування, цінність дослідження виявляється вищою, як і цінність отриманих знань і розробок, якщо вони будуть направлені на покращення програмного продукту.

Розроблений застосунок як результат дослідження може бути рекомендований для використання людьми як помічник у смартфоні для збереження інформації про спожиту їжу й автоматичних обчислень показників спожитої їжі (калорій та поживних речовин) у зручному дизайні. Доцільність

використання пояснюється збереженням часу у порівнянні з ручним пошуком і обчисленнями.

СПИСОК ВИКОРИСТАНИХ ПОСИЛАНЬ

1. Dart (programming language). Автор оригіналу: Wikipedia URL: https://en.wikipedia.org/wiki/Dart_(programming_language)
2. A tour of the Dart language. Автор оригіналу: dart.dev URL: https://dart.dev/guides/language/language-tour
3. Flutter (software). Автор оригіналу: Wikipedia URL: https://en.wikipedia.org/wiki/Flutter_(software)
4. Will Flutter and Dart be the Next Big Thing in Application Development? Автор оригіналу: Attila Vágó URL: https://attilavago.medium.com/will-flutter-and-dart-be-the-next-big-thing-in-application-development-e8e218599c89
5. Android Development: Java vs Kotlin vs React Native vs Flutter. Автор оригіналу: Arman Hossain URL: https://levelup.gitconnected.com/begin-learning-android-in-2019-a-freshers-approach-443b3f436e7d
6. Дослідження кросплатформенного фреймворку Flutter. чи означатиме розквіт цієї технології зникнення нативної розробки на Android та iOS? Комп'ютерні науки та кібербезпека (2). Харків, 2019. С. 19-25.
7. Android Studio. Автор оригіналу: Wikipedia URL: https://en.wikipedia.org/wiki/Android_Studio
8. Огляд інструментальних середовищ для розробки мобільних додатків в освітніх цілях. Молодь і ринок, 2017. № 1. Дрогобич, 2017. 28 с.
9. Firebase. Автор оригіналу: Wikipedia URL: https://en.wikipedia.org/wiki/Firebase
10. Особливості розробки мобільного додатку з використанням хмарної бази даних. «Молодий вчений» № 11 (51) (листопад 2017 р., Черкаси). Черкаси, 2017. 1079 с.
11. Cloud Firestore. Автор оригіналу: firebase.google.com URL: https://firebase.google.com/docs/firestore

12. Cloud Firestore Data model. Автор оригіналу: firebase.google.com URL: https://firebase.google.com/docs/firestore/data-model
13. FlutterFire Overview. Автор оригіналу: firebase.flutter.dev URL: https://firebase.flutter.dev/docs/overview/
14. Figma. Автор оригіналу: Wikipedia URL: https://uk.wikipedia.org/wiki/Figma
15. Карабін О. Й., Петрів Х. Б. Деякі особливості розробки хмарного навчального середовища засобами Figma. *Сучасні цифрові технології та інноваційні методики навчання: досвід, тенденції, перспективи*. (Тернопіль,11-12 лист. 2021 р.), № 8. Тернопіль, 2021. С. 21-24.
16. Литвиненко О. Л., Скляренко О. В., Колодінська Я. О. Логіка побудови інтерфейсів у програмному забезпеченні. *Вісник Національного технічного університету "ХПІ". Сер. : Нові рішення в сучасних технологіях : зб. наук. пр.* (Харків: НТУ "ХПІ", 2020. № 1 (3)). Харків, 2020. С. 54-59.
17. Pink. Автор оригіналу: Wikipedia URL: https://en.wikipedia.org/wiki/Pink
18. Eva Heller. Psychologie de la couleur: effets et symboliques. Pyramyd, 2009. С. 179 -184.
19. Культура як феномен людського духу (багатогранність і наукове осмислення). Збірник тез доповідей IV Міжнародної наукової конференції курсантів і студентів. (Львів, 16-17 листопада 2017 року). Львів: ЛДУ БЖД, 2017. 432 с.
20. An Overview of the generated files and folders in the new flutter project. Автор оригіналу: ujjwal bansal URL: https://towardsdev.com/an-overview-of-the-generated-files-and-folders-in-the-new-flutter-project-61c596ca81c6
21. The pubspec file. Автор оригіналу: dart.dev URL: https://dart.dev/tools/pub/pubspec
22. material library. Автор оригіналу: api.flutter.dev URL: https://api.flutter.dev/flutter/material/material-library.html

23. Flutter BottomNavigationBar Tutorial with Examples. Автор оригіналу: o7planning.org URL: https://o7planning.org/12849/flutter-bottomnavigationbar
24. Using Firebase for NoSQL DB. Автор оригіналу: Juan Manuel Guerrero URL: https://medium.com/@juans_gro/using-firebase-for-nosql-db-779cb71f9c3c
25. Choose a data structure. Автор оригіналу: firebase.google.com URL: https://firebase.google.com/docs/firestore/manage-data/structure-data
26. What Is Google Firebase and Why Should You Use It? Автор оригіналу: Idowu Omisola URL: https://www.makeuseof.com/what-is-google-firebase-why-use-it/
27. CustomPainter class. Автор оригіналу: api.flutter.dev URL: https://api.flutter.dev/flutter/rendering/CustomPainter-class.html
28. Flutter Custom Paint - Made Easy with Flutter Shape Maker. Автор оригіналу: Paras Jain URL: https://retroportalstudio.medium.com/auto-generate-flutter-custom-paint-code-flutter-shape-maker-be51e41daf89
29. FutureBuilder<T> class. Автор оригіналу: api.flutter.dev URL: https://api.flutter.dev/flutter/widgets/FutureBuilder-class.html
30. StreamBuilder<T> class. Автор оригіналу: api.flutter.dev URL: https://api.flutter.dev/flutter/widgets/StreamBuilder-class.html

Декларація щодо унікальності текстів роботи
та невикористання матеріалів інших авторів без посилань

Новицький Максим Олександрович
Прізвище, ім’я, по батькові

Факультет інформаційних і прикладних технологій
Факультет

122 Комп’ютерні науки
Шифр і назва спеціальності

Сучасні інформаційні технології та програмування
Освітня програма

ДЕКЛАРАЦІЯ

Усвідомлюючи свою відповідальність за надання неправдивої інформації, стверджую, що подана кваліфікаційна (бакалаврська) робота на тему: «Розробка мобільного застосунку аналізу особистого раціону людини» є написаною мною особисто.

Одночасно заявляю, що ця робота:

- не передавалась іншим особам і подається до захисту вперше;
- не порушує авторських та суміжних прав, закріплених статтями 21-25 Закону України «Про авторське право та суміжні права»;
- не отримувались іншими особами, а також дані та інформація не отримувались у недозволений спосіб.

Я усвідомлюю, що у разі порушення цього порядку моя кваліфікаційна (бакалаврська) робота буде відхилена без права її захисту, або під час захисту за неї буде поставлена оцінка «незадовільно».

_____ _____

дата підпис