НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ПОГОДЖЕНО **Декан факультету**	**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ** **Завідувач кафедри комп'ютерних наук**
_____ Ігор БОЛБОТ (підпис)	**_____ Белла ГОЛУБ** (підпис)
«___» _____ 2026 р.	«___» _____ 2026 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Програмне забезпечення для персоналізованого підбору кулінарних рецептів і планування харчування »

Спеціальність «121 Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення»

Гарант освітньої програми
к.т.н., доцент _____ **Ганна ВАЙГАНГ**
(підпис)

Керівник бакалаврської кваліфікаційної роботи
к.е.н., ст.викладач _____ **Дмитро НІКОЛАЄНКО**
(підпис)

Виконала _____ **Анна ПАНАСЮК**
(підпис)

Київ-2026

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук

к.т.н., доцент _____ Белла ГОЛУБ
(підпис)

«___» _____ 2025 р.

ЗАВДАННЯ
ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
ЗДОБУВАЧУ
Панасюк Анні Костянтинівні

Спеціальність «121 Інженерія програмного забезпечення»
Освітньо-професійна програма «Інженерія програмного забезпечення»
Тема бакалаврської кваліфікаційної роботи
«Програмне забезпечення для персоналізованого підбору кулінарних рецептів і планування харчування»
затверджена наказом від «19» грудня 2025 р. № 3204 «С»
Термін подання завершеної роботи на кафедру «22» травня 2026 р.
Вихідні дані до бакалаврської кваліфікаційної роботи (проєкту): предметна область персоналізованого підбору кулінарних рецептів і планування харчування; вимоги до мобільного застосунку; дані про рецепти, харчові обмеження, калорійність страв і плани харчування користувача.
Перелік питань, що підлягають дослідженню:

1. Аналіз предметної області та вимог до системи.
2. Проєктування і реалізація програмного забезпечення.
3. Тестування та впровадження розробленої системи.

Дата видачі завдання «19» грудня 2025 р.

Керівник бакалаврської кваліфікаційної роботи _____ (підпис) **Дмитро НІКОЛАЄНКО**

Завдання взяла до виконання _____ (підпис) **Анна ПАНАСЮК**

ЗМІСТ

ВСТУП

Актуальність теми. У сучасних умовах стрімкого ритму життя все більше людей прагнуть приділяти увагу якості харчування, контролю калорійності та формуванню збалансованого раціону для власних потреб. Проте самостійний підбір рецептів і врахування рекомендацій дієтологів щодо правильного підбору меню потребують значних витрат ресурсів. Немало часу займає складання списку покупок після вибору рецептів, а хаотичне формування списку покупок провокує зайві витрати та незручності.

Додаткову складність створює необхідність валідації великої кількісті рецептів у відкритій мережі або книгах, виключення рецептів з забороненими продуктами і алергенами, підрахунок БЖУ та калорійністі рецепту, відповідно до особистих обмежень людини.

Саме тому ефективним є використання системи, що автоматизує процес персоналізованого підбору кулінарних рецептів і планування харчування. Використання такого програмного забезпечення спрощує формування меню, оптимізовує процес пошуку рецептів, автоматизовує контроль калорійності раціону, полегшує організацію списку покупок.

Мета та завдання роботи. Підвищення ефективністі підбору кулінарних рецептів і планування харчування з урахуванням дієтичних обмежень та уподобань шляхом розробки програмного забезпечення для персоналізованого підбору кулінарних рецептів і планування харчування з урахуванням індивідуальних параметрів користувача, його харчових уподобань, дієтичних обмежень та добової норми калорій

Об’єкт дослідження. Процес персоналізованого підбору кулінарних рецептів та планування щоденного харчування користувача з урахуванням індивідуальних показників.

Предмет дослідження. Методи та програмні засоби автоматизації підбору рецептів і формування меню на основі індивідуальних дієтичних обмежень, харчових уподобань та розрахункової норми калорій.

Апробація. Основні етапи роботи над дипломним проєктом доповідалися та обговорювалися на науково-практичній конференції: VII Всеукраїнської науково-практичної інтернет конференції студентів та аспірантів «Теоретичні та прикладні аспекти розробки комп'ютерних систем 2026» (27.09.2026) з доповіддю на тему «Програмне забезпечення для персоналізованого підбору кулінарних рецептів і планування харчування»

Структура роботи. Бакалаврська кваліфікаційна робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел та додатків. Перший розділ присвячений аналізу процесу персоналізованого підбору кулінарних рецептів і планування харчування — у ньому розглянуто наявні програмні рішення, сформульовано постановку задачі та виконано моделювання предметної області.

Другий розділ зосереджений на інформаційному забезпеченні системи: побудовано логічну модель даних, обґрунтовано вибір Cloud Firestore та наведено структуру основних колекцій бази даних.

Третій розділ охоплює питання організаційної структури програмного забезпечення, вибору інструментарію, реалізації інтерфейсу користувача, алгоритмізації бізнес-логіки, а також забезпечення взаємодії інтерфейсу з базою даних.

Завершальна, четверта частина роботи містить рекомендації щодо впровадження та експлуатації системи. Тут описано архітектуру, підходи до тестування, апаратні й програмні вимоги, склад інсталяційного пакету та результати дослідної експлуатації. Список використаних джерел містить 23 найменування. Додатки містять лістинг основних програмних модулів, результати модульного тестування та опубліковані тези за темою роботи

1 АНАЛІЗ ПРОЦЕСУ ПЕРСОНАЛІЗОВАНОГО ПІДБОРУ КУЛІНАРНИХ РЕЦЕПТІВ І ПЛАНУВАННЯ ХАРЧУВАННЯ ЯК ОБ'ЄКТУ ДОСЛІДЖЕННЯ

1.1 Опис процесу персоналізованого підбору кулінарних рецептів і планування харчування.

Процес індивідуального добору рецептів та планування харчування є складним завданням, яке передбачає врахування індивідальних особливостей людини, її харчових потреб, фізіологічних параметрів та способу життя. Самостійне врахування всіх цих параметрів потребує часу та спеціальних знань, тому користувачеві складно швидко сформувати збалансоване меню без допоміжних інструментів. Під час формування меню необхідно врахувати добову потребу в калоріях, співвідношення білків, жирів, вуглеволів, наявність харчових обмежень, алергенів, особистих смакових уподобань та поставлених цілей харчування[1].

До цілей харчування належать: зниження маси тіла, підтримання поточної форми, набір м'язової маси, дотримання спеціального режиму харчування. Крім цього, важливе значення мають доступність продуктів, час приготування страв, регулярність прийомів їжі та можливість адаптації меню до щоденного графіка людини.

Ключовий етап планування харчування є визначення добової потреби організму в енергії. Для цього використовуються метрики розрахунку добової норма калорій, які враховують індивідуальні показники людини: стать, вік, масу тіла, зріст, рівень фізичної активності.

Основою для визначення добової енергетичної потреби є показник базального метаболізму, який описує кількість енергії, необхідної організму для підтримання життєво важливих процесів у стані спокою.

Однією з найбільш точних метрик розрахунку базального метаболізму (BMR) є формула Mifflin–St Jeor[2], [3].

Для чоловіків формула Міффліна-Сан-Жеора :

$$BMR=(10\times\text{маса тіла [кг]})+(6.25\times\text{зріст [см]})-(5\times\text{вік [роки]})+5 \quad (1)$$

Для жінок формула Mifflin–St Jeor:

$$BMR=(10\times\text{маса тіла [кг]})+(6.25\times\text{зріст [см]})-(5\times\text{вік [роки]})-161 \quad (2)$$

де:

- BMR - базальний метаболізм
- маса тіла - маса людини у кілограмах
- зріст - зріст людини у сантиметрах
- вік - вік людини у роках

Значення значення базального метаболізму враховує лише мінімальні енергетичні потреби організму. Для визначення добової норми калорій використовується коефіцієнт фізичної активності, який напряму залежить від способу життя людини[4].

Добова норма калорій розраховується за формулою:

$$TDEE=BMR\times AF \quad (3)$$

де:

- TDEE - загальні добові енерговитрати;
- BMR - базальний метаболізм;
- AF - коефіцієнт фізичної активності.

Залежно від рівня активності використовуються коефіцієнти(AF у формулі)

- 1.2 - низька активність або сидячий спосіб життя;
- 1.375 - легка фізична активність;
- 1.55 - помірна активність;
- 1.725 - висока активність;
- 1.9 - дуже високий рівень фізичних навантажень.

Після визначення добової норми калорій здійснюється підбір персоналізованого меню та розподіл енергетичної цінності раціону між окремими прийомами їжі. Планування передбачає створення збалансованого меню на день, тиждень відповідно до режиму харчування людини та встановлених цілей. Добова норма калорійності розподіляється між сніданком, обідом, вечерею та додатковими перекусами, що підтримає стабільний режим харчування.

У середньому добова калорійність може розподілятися таким чином: приблизно 25-30 % калорій припадає на сніданок, 35-40 % на обід, 20–25 % на вечерю, а решта – на додаткові перекуси між основними прийомами їжі.

Подібний розподіл дозволяє уникнути надмірного навантаження на організм у вечірній час та підтримувати стабільний рівень енергії протягом дня.

Важливо також врахувати співвідношення білків, жирів і вуглеводів може змінюватися залежно від цілей користувача. Для формування раціону кількість жирів і вуглеводів визначається відносно маси тіла людини, тоді як кількість вуглеводів розраховується на основі залишкової добової калорійності[4].

Підчас обрахунку норми білків застосовуватися коефіцієнти споживання на 1 кг ваги тіла.

Білки:

- схуднення: 1.6 г / кг
- набір ваги: 1.5 г / кг
- підтримка ваги: 1.3 г / кг
- здорове харчування: 1.2 г / кг

Також враховується коефіцієнт активності при розрахунку норми білків від 0 до 0.2 г / кг.

Жири:

- схуднення: 0.7 г / кг
- набір ваги: 1.0 г / кг
- підтримка ваги: 0.8 г / кг
- здорове харчування: 0.8 г / кг

Вуглеводи:

Розраховуються не напряму коефіцієнтом, а з залишку калорій:

$$\text{Вуглеводи} = (\text{денні\_калорії} - \text{білки} * 4 - \text{жири} * 9) / 4 \quad (4)$$

Наведені коефіцієнти сформовані на основі загальноприйнятих діапазонів споживання макронутрієнтів, зокрема рекомендацій щодо споживання білка для фізично активних осіб у межах 1,4–2,0 г/кг маси тіла на добу та допустимих діапазонів розподілу енергії між білками, жирами й вуглеводами.

1.2 Аналіз існуючих рішень

Щоб оптимізувати процес підбору кулінарних рецептів і планування харчування, доцільно використовувати автоматизовані системи для персоналізованого підбору рецептів, контролю калорійності та формування меню. Сучасні мобільні застосунки дозволяють спростити організацію харчування, автоматизувати розрахунок добової норми калорій і забезпечити більш зручний контроль раціону користувача[5].

Однією з таких систем є Mealime[6]. Це мобільний застосунок для пошуку рецептів, перегляд покрокових інструкцій приготування та планування меню. Користувач може переглядати рецепти, знайомитися зі складом страв, послідовністю приготування та змінювати кількість порцій, формувати меню, додаючи рецепти. Система автоматично сформує продуктовий кошик для користувача на основі продуктів, що містяться в рецептах доданих до меню. Рекомендації користувач отримує згідно заповеного профілю користувача з урахуванням персональних вподобань.

На рис. 1 зображено інтерфейс користувача застосунку Mealime, де можна побачити вкладку планування меню та додані рецепти.

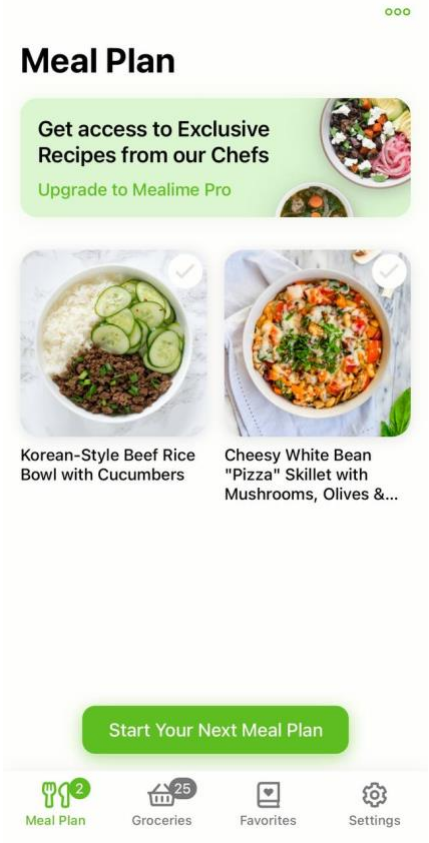

Рисунок 1 Інтерфейс планування меню в мобільному застосунку Mealime

Функціональні характеристики вище згаданої системи:

1. Формування персоналізованого меню;
2. Планування харчування на декілька днів або на тиждень;
3. Підбір рецептів відповідно до вподобань користувача;
4. Врахування алергенів та небажаних продуктів;
5. Автоматичне формування списку покупок;
6. Покрокові інструкції приготування страв;
7. Можливість налаштування кількості порцій.

Недоліки системи:

- Персоналізація більше стосується вподобань і обмежень, ніж повного індивідуального розрахунку калорійності та БЖВ;
- рецепти не відповідають локальним харчовим звичкам користувачів;
- відсутність функції детально планувати меню за днями та сформувати детальний план з розподіленням кожної страви за прийомом їжі.

Також прикладом системи із схожими функціональними можливостями є FatSecret[7]. Мобільний застосунок, призначений для підрахунку калорій, ведення харчового щоденника, контролю макронутрієнтів. Фіксуються спожиті продукти, калорійність раціону, контролюється кількість білків, жирів й вуглеводів, а також використовується база продуктів для швидкого внесення харчових даних.

На рис. 2 зображено екран налаштування харчових цілей користувача в мобільному застосунку FatSecret.

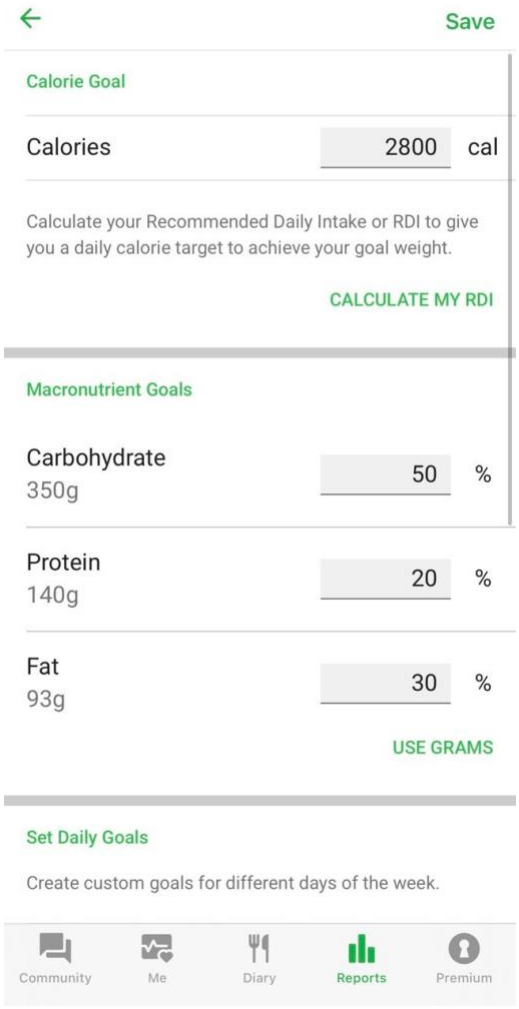

Рисунок 2 Екран налаштування добових калорій і макронутрієнтів у застосунку FatSecret

Відображена добова норма калорійності, а також розподіл макронутрієнтів вуглеводів, білків і жирів. Наведені значення автоматично згенеровані на основі персональних характеристик і цілей користувача, якщо цілі можна встановити їх самостійно. Це дає можливість користувачу

орієнтуватися на рекомендовану добову норму та контролювати співвідношення поживних речовин у раціоні.

Функціональні характеристики системи FatSecret:

1. Підрахунок добової калорійності;
2. Ведення харчового щоденника;
3. Контроль білків, жирів і вуглеводів;
4. Пошук продуктів у базі даних;
5. Сканування продуктів за штрихкодом;
6. Розпізнавання їжі за фото;
7. Ведення щоденника фізичної активності;
8. Відстеження маси тіла;
9. Формування звітів щодо калорій і макронутрієнтів.

Недоліки системи FatSecret:

- Основний акцент зроблено на фіксації спожитої їжі, а не на повноцінному автоматизованому плануванні меню;
- користувачу потрібно самостійно вносити значну частину даних про продукти та страви;
- персоналізований підбір рецептів реалізований обмежено;
- система не забезпечує повного формування раціону за конкретними прийомами їжі на кожен день.

Дивлячись на такі системи можна сказати, що вони роблять процес пошуку рецептів, контролю калорійності та планування харчування більш комфортним і структурованим. Водночас жодна з розглянутих систем не поєднує повною мірою персоналізований підбір рецептів, детальне планування меню за днями та прийомами їжі, автоматичний розрахунок добової калорійності, контроль БЖВ і формування списку покупок в межах єдиного рішення.

Таким чином, існуючі рішення частково закривають потреби користувачів у сфері контролю харчування, однак залишають простір для вдосконалення.

1.3 Постановка завдання

Метою роботи є розробка програмного забезпечення для персоналізованого підбору кулінарних рецептів і планування харчування. Розроблюване програмне забезпечення має забезпечувати автоматизацію процесу персоналізованого підбору кулінарних рецептів, планування харчування, контролю калорійності раціону та формування списку покупок.

Програмне забезпечення, що створюється, має відповідати вимогам:

1. Забезпечити створення та редагування профілю користувача.
2. Використовувати персональні параметри користувача: вік, стать, зріст, масу тіла, рівень фізичної активності та харчову ціль. Для підбору індивідуального харчування.
3. Запитувати харчові вподобання, небажані продукти, алергени та дієтичні обмеження.
4. Автоматично розраховувати добову норму калорій на основі даних профілю користувача.
5. Розраховувати співвідношення білків, жирів, вуглеводів.
6. Надати каталог кулінарних рецептів.
7. Забезпечувати пошук і фільтрацію рецептів за назвою, інгредієнтами, калорійністю, типом страви та дієтичними обмеженнями.
8. Сформувати меню з можливістю розподілити додавання рецептів між прийомами їжі.
9. Виконувати підрахунок калорійності сформованого меню.
10. Формувати список та редагувати покупок на основі інгредієнтів рецептів або власноруч доданих продуктів.

Вхідною інформацією у програмному забезпеченні є:

- Персональні параметри користувача: стать, вік, зріст, маса тіла, рівень фізичної активності;

- харчова ціль користувача: зниження маси тіла, підтримання ваги, набір маси або здорове харчування;
- харчові вподобання, алергени, дієтичні обмеження та небажані продукти;
- дані рецепта: назва, інгредієнти, калорійність, кількість порцій та інструкція приготування;

Вихідною інформацією у програмному забезпечення є:

- персоналізований перелік рекомендованих рецептів;
- розрахована добова норма калорій і рекомендоване співвідношення білків, жирів і вуглеводів;
- сформований план харчування на день або тиждень;
- загальна калорійність меню та залишок калорій до добової норми;
- список необхідних продуктів для приготування запланованих страв.

1.4 Моделювання предметної області

На етапі проєктування важливо не лише визначити вимоги до програмного забезпечення, а й описати взаємодію основних учасників із процесами предметної області. Для цього доцільно розроблюється діаграма прецедентів[9].

1.4.1 Діаграма прецедентів. Діаграма демонструє зв’язок між акторами і прецедентами системи. Основними елементами діаграми є актори і прецеденти.

Актор – це користувач або об’єкт, що взаємодіє із системою.

Прецедент – це дія, що виконується конкретним актором у системі. Прецедент може бути напряму пов’язаний з актором, з яким взаємодіє, або з іншим прецедентом, від виконання якого залежить виконання даного прецеденту.

Для побудови діаграми прецедентів розроблюваної системи спершу необхідно визначити акторів, які взаємодіють з нею. На таблиці 1 показано перелік акторів, які взаємодіють з системою[10].

Таблиця 1

Актори, які діють у системі

Актор	Опис
1	**2**
Особа, що слідкує за харчуванням та самостійно готує	Основний учасник процесу, який здійснює пошук рецептів, визначає харчові вподобання, планує меню, формує список покупок і закуповує необхідні продукти.
Дієтолог/нутриціолог	Фахівець, який надає дієтичні рекомендації, допомагає визначити норму калорій, оцінює калорійність страв і бере участь у формуванні персоналізованого раціону.

Таблиця 1(продовження)

1	2
Магазин/ринок	Зовнішній учасник предметної області, який забезпечує користувача продуктами.

Після визначення акторів, які діють у системі, небхідно описати прецеденти, які описують основний функціонал інформаційної системи (таблиця 2). На їх основі побудована діаграма прецедентів, з якою можна ознайомитись на рис. 3.

Система має основного актора – особа, що слідкує за харчуванням та самостійно готує. Також дієтолог/нутріціолог, від якого надходять дієтичні рекомендації. Магазин/ринок для забезпечення продуктами особи, що готується до приготування страви. Кожен актор має свій вплив на систему. Детальніше про кожен прецедент та роль акторів можна ознайомитися нижче.

Таблиця 2

Перелік прецедентів у системі

Актор	**Найменування прецеденту**	**Формулювання**
1	**2**	**3**
Особа, що слідкує за харчуванням та самостійно готує	Пошук рецептів	Пошук рецептів відповідно до вподобань користувача.
Особа, що слідкує за харчуванням та самостійно готує	Визначення харчових уподобань	Визначення смакових вподобань, алергенів та обмежень.
Особа, що слідкує за харчуванням та самостійно готує	Планування списку покупок	Формування переліку продуктів для обраних страв.
Особа, що слідкує за харчуванням та самостійно готує	Закупка продуктів	Придбання продуктів відповідно до списку покупок.
Особа, що слідкує за харчуванням та самостійно готує	Консультація з дієтологом	Отримання рекомендацій щодо харчування.

Таблиця 2 (завершення)

1	**2**	**3**
Дієтолог/нутриціолог	Надання дієтичних рекомендацій	Надання рекомендацій щодо раціону та обмежень.
Дієтолог/нутриціолог	Формування персоналізованого раціону	Складання меню відповідно до потреб людини.
Дієтолог/нутриціолог	Встановлення норми калорій	Визначення добової норми калорій.
Дієтолог/нутриціолог	Оцінка калорійності страв	Оцінювання енергетичної цінності страв.
Формування персоналізованого раціону	Урахування дієтичних обмежень	Врахування дієтичних та індивідуальних обмежень.
Формування персоналізованого раціону	Урахування алергенів	Виключення продуктів, що містять алергени.
Магазин/ринок	Надання продукції	Забезпечення користувача необхідними продуктами.
Магазин/ринок	Повідомлення про відсутність продукції	Інформування про відсутність окремих продуктів.

Наведені прецеденти відображають основні дії, опису предметної області. Основна частина процесу пов’язана з пошуком рецептів, визначенням харчових уподобань, формуванням раціону, урахуванням алергенів і дієтичних обмежень, а також підготовкою списку покупок. Такий перелік прецедентів дає змогу визначити межі досліджуваного процесу та якісно продемонстувати у вигляді графічної діаграми.

Для початкового наповнення колекції recipes використовувався сервіс Spoonacular Food API, який надає доступ до даних про кулінарні рецепти, інгредієнти, калорійність, харчову цінність і спосіб приготування страв [8].

Отримані дані були використані для завантаження рецептів у базу даних Cloud Firestore та подальшого використання їх у функціях пошуку, перегляду рецептів і формування плану харчування.

На рис. 3 зображено діаграму прецедентів предметної області, що демонструє взаємодію акторів із процесами персоналізованого підбору рецептів і планування харчування.

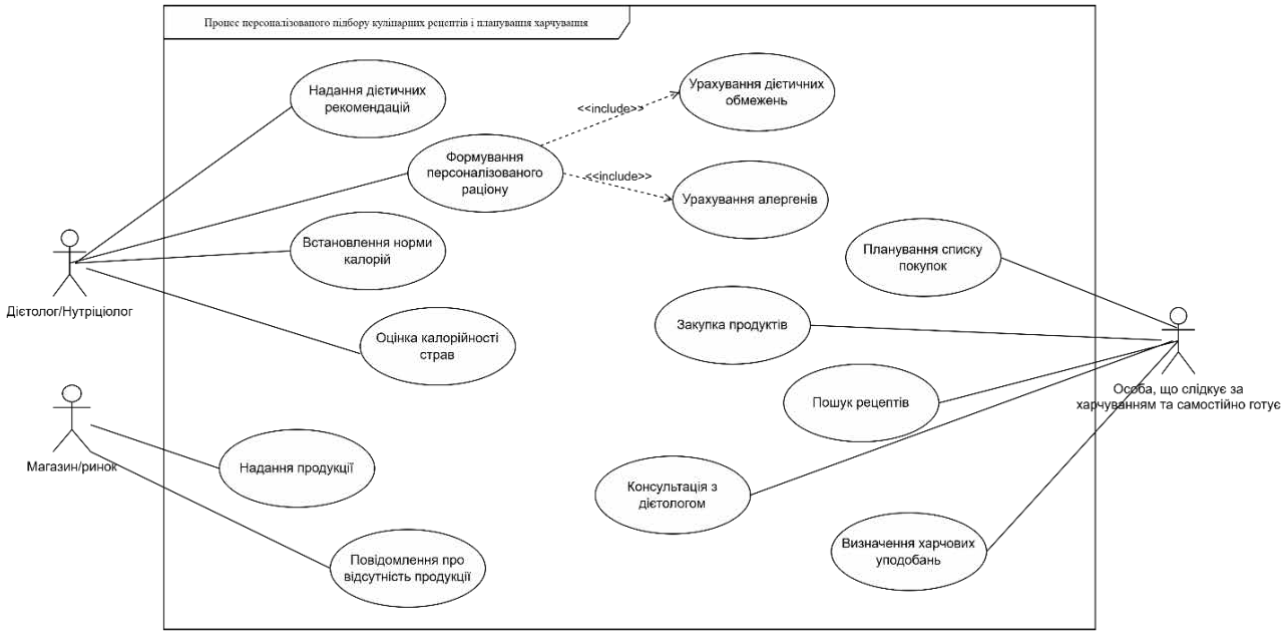

Рисунок 3 Діаграма прецедентів предметної області

1.4.2 Діаграма активності. Щоб краще розуміти за яким принципом працює система необхідно побудувати діаграму активності. Діаграма використовується для відображення послідовності виконання дій, умовних переходів та можливих альтернативних сценаріїв у межах певного процесу.

На відміну від діаграми прецедентів, яка показує загальні можливості системи та взаємодію акторів із ними, діаграма активності деталізує сам процес виконання дій. Дії відображають окремі етапи виконання процесу, переходи показують послідовність виконання дій, а розгалуження використовуються для відображення певної умови[9]. Основними елементами діаграми активності є початковий вузол, дії, переходи, вузли розгалуження та кінцевий вузол. На рис. 4 зображено діаграму активності процесу персоналізованого підбору кулінарних рецептів і планування харчування.

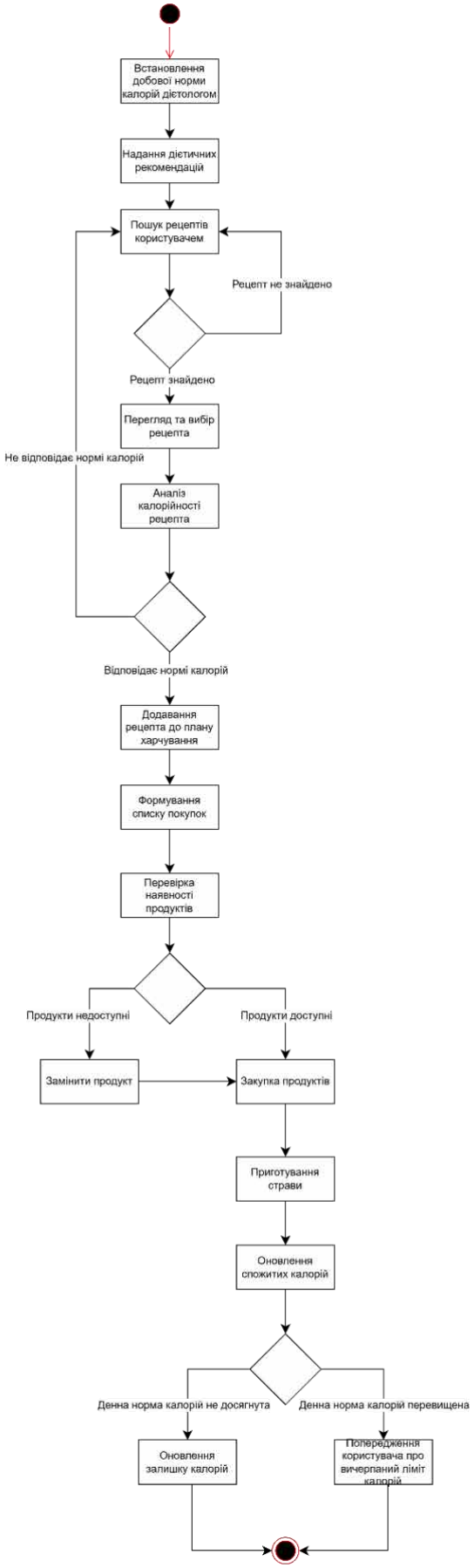

Рисунок 4 Діаграма активності процесу персоналізованого підбору рецептів і планування харчування

Діаграма активності відображає послідовність дій під час персоналізованого підбору рецепта та планування харчування. Процес починається з визначення добової норми калорій і надання дієтичних рекомендацій. Після цього користувач здійснює пошук рецепта, переглядає знайдені варіанти та обирає відповідну страву. Якщо рецепт не знайдено або він не відповідає встановленій нормі калорій, процес передбачає повернення до пошуку або вибір іншого рецепта.

Після вибору рецепта виконується аналіз його калорійності, додавання до плану харчування, формування списку покупок і перевірка наявності необхідних продуктів. Якщо продукти доступні, користувач переходить до їх закупівлі та приготування страви. Завершальним етапом є оновлення спожитих калорій і перевірка добового ліміту: якщо норму не перевищено, оновлюється залишок калорій, а в разі перевищення користувач отримує відповідне попередження.

1.4.3 Діаграма послідовності. Діаграма активності добре описує алгоритм дій актора, але необхідно розуміти з якими даними він взаємодіє з системою. Діаграма послідовності дозволяє простежити, як саме відбувається взаємодія між особою, що планує харчування, дієтологом або нутриціологом, книгою рецептів, довідником калорійності та ресурсом продуктів.

Кожний учасник діаграми має власну лінію життя, яка відображає період його участі у процесі. Вертикальна лінія життя показує, що актор або об’єкт може надсилати й отримувати дані протягом певного проміжку часу. Періоди активного виконання дій позначаються фокусом керування – вузьким вертикальним прямокутником, який демонструє момент початку та завершення певної операції[9].

Така діаграма дає змогу деталізувати процес обміну інформацією між учасниками та визначити, які дані використовуються на кожному етапі формування персоналізованого раціону.

На рис. 5 зображено діаграму послідовності процесу персоналізованого підбору кулінарних рецептів і планування харчування.

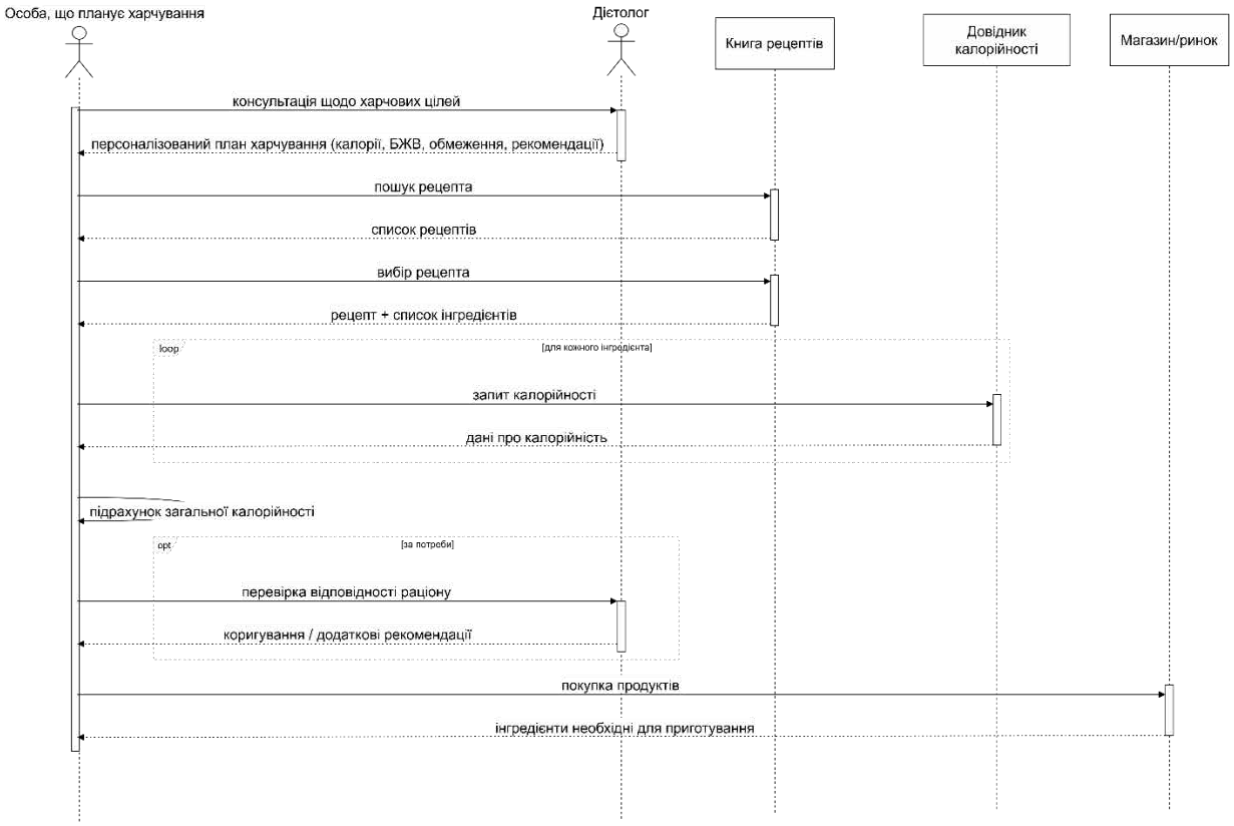

Рисунок 5 Діаграма послідовності процесу персоналізованого підбору рецептів і планування харчування

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Побудова логічної моделі

Основним атрибутом з яким працює програмне забезпечення, є дані. Для організації даних найбільш зручним методом є база даних. На етапі проєктування важливо визначити основні сутності предметної області, їх атрибути та зв’язки між ними.

Першим етапом проектування БД є створення логічної моделі. Логічна модель даних описує структуру інформації без прив’язки до конкретної системи керування базами даних. Вона дозволяє визначити, які об’єкти повинні зберігатися в системі, які властивості вони мають і як взаємодіють між собою. Основними елементами логічної моделі є сутності, атрибути та відношення[9].

У логічній моделі виділено такі основні сутності:

Таблиця 3

Сутність	Опис
1	**2**
USER	Описує користувача системи. Містить персональні дані користувача, які використовуються для формування індивідуального плану харчування та розрахунку добової норми калорій.
CALORIE_TRACKER	Відповідає за збереження інформації про добову норму калорій, кількість спожитих калорій та залишок калорій.
MEAL_PLAN	Описує план харчування користувача. Вона містить інформацію про дату планування, тип прийому їжі, розмір порції та стан виконання плану.

Таблиця 3(продовження)

RECIPE	Представляє кулінарний рецепт. Містить назву страви, тип кухні, інструкцію з приготування загальну калорійність рецепта.
SHOPPING_LIST	Описує список покупок користувача. Використовується для збереження переліку продуктів для приготування запланованих страв.
PRODUCT	Описує продукт, який може входити до складу рецепта або списку покупок.

На рис. 6 подано логічну модель даних програмного забезпечення.

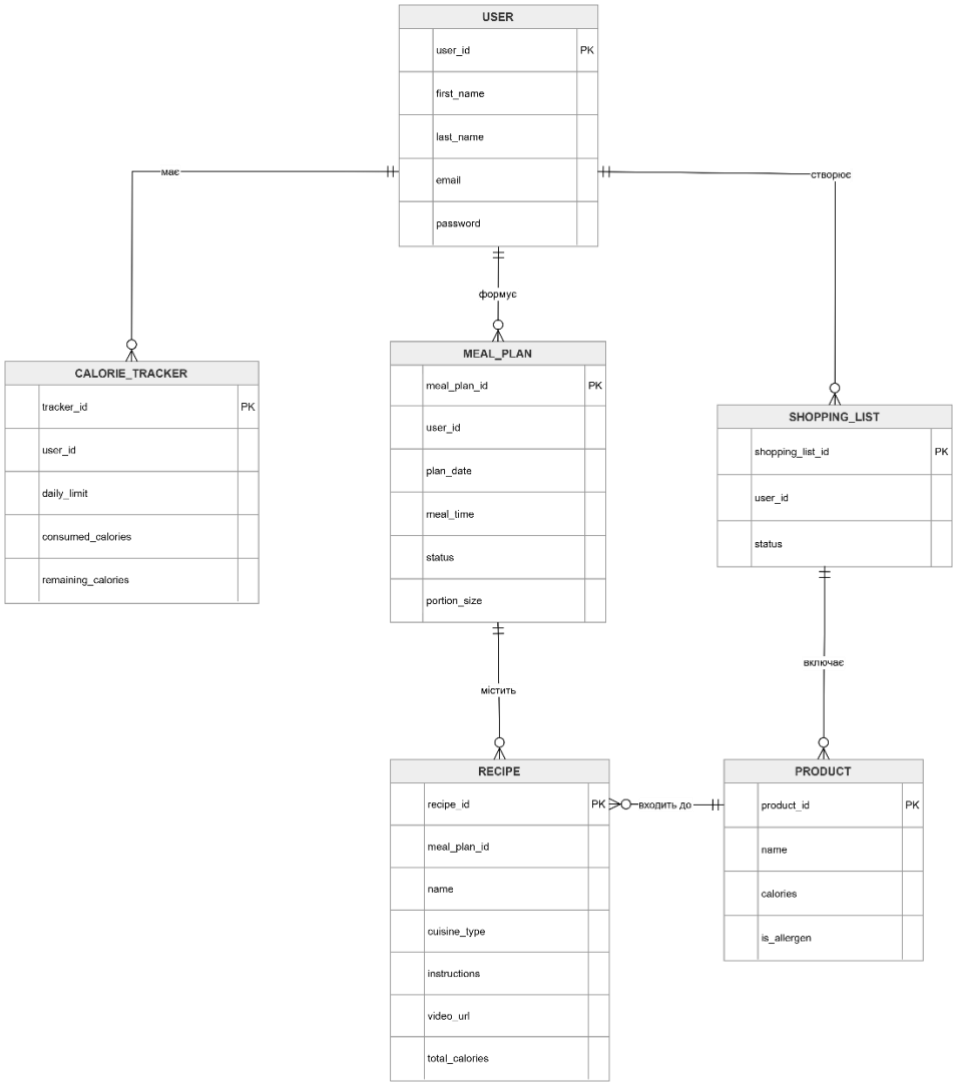

Рисунок 6 Логічна модель даних програмного забезпечення для персоналізованого підбору кулінарних рецептів і планування харчування

2.2 Вибір методів та засобів для реалізації інформаційного забезпечення системи

Для реалізації інформаційного забезпечення програмного забезпечення для персоналізованого підбору кулінарних рецептів і планування харчування доцільно використати NoSQL документо-орієнтовану базу даних[12].

Використання документо-орієнтованої NoSQL-бази даних є доцільним, оскільки структура даних у системі не є жорстко фіксованою: окремі документи можуть містити різні набори полів, гнучку структуру й можуть зберігатися у вигляді документів із вкладеними даними.

Документо-орієнтована база даних – це різновид NoSQL-бази даних, у якій інформація зберігається у вигляді документів. У Cloud Firestore документи об’єднуються в колекції, а кожен документ містить набір пар «ключ – значення». Документи містять вкладені об’єкти та підколекції, що зручно для збереження складних структур даних.

Для реалізації БД було обрано Firebase Cloud Firestore. Cloud Firestore є гнучкою масштабованою NoSQL хмарною базою даних побудованою на інфраструктурі Google Cloud, яка призначена для зберігання та синхронізації даних між клієнтськими й серверними застосунками.

Для БД була обрана Firebase Cloud Firestore завдяки перевагам[13]:

- Дані можна зберігати у вигляді документів і колекцій, що добре підходить для предсталених сутностей.
- База даних побудована на інфраструктурі Google Cloud, що дозволяє використовувати її для застосунків із потенційним зростанням кількості користувачів і даних.
- Firestore дозволяє працювати з даними навіть за нестабільного інтернет-з’єднання, що є важливим для мобільного застосунку.
- Firestore добре поєднується з іншими сервісами Firebase, зокрема Firebase Authentication.

2.3 Створення об’єктів бази даних

2.3.1 Створення бази даних у Firebase Cloud Firestore. Створення бази даних виконано у середовищі Firebase Console. Для цього в межах проєкту recipe-planner-app було створено базу даних (Рис. 7) .

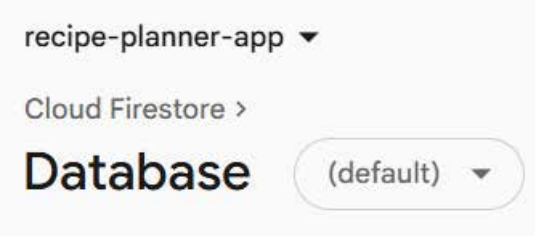

Рисунок 7 Створена БД у Cloud Firestore

2.3.2 Створення колекцій. Після створення бази даних у Cloud Firestore було сформовано основні колекції та підколекції[12], необхідні для збереження даних програмного забезпечення. У базі даних створено колекції users та recipes, а також підколекції, що належать конкретному користувачу: meal_plans, shopping_lists та favorites.

Колекція users використовується для збереження профілів користувачів. Кожен документ у цій колекції відповідає окремому користувачу та містить його персональні параметри, необхідні для планування харчування й розрахунку добової калорійності. Всередині документа користувача створюються підколекції для збереження індивідуальних даних: планів харчування, списків покупок та улюблених рецептів.

Колекція recipes призначена для збереження кулінарних рецептів. У документах цієї колекції зберігається інформація про назву рецепта, інгредієнти, калорійність, харчову цінність і спосіб приготування.

Для захисту даних користувачів у Cloud Firestore використовуються Firebase Security Rules. Вони дозволяють визначати правила читання та запису документів залежно від стану автентифікації користувача та його ідентифікатора. Це важливо для розроблюваної системи, оскільки профілі користувачів, плани харчування, списки покупок та обрані рецепти мають бути доступні лише відповідному авторизованому користувачу [14].

На рис. 8 зображено приклад документа користувача в колекції users у Cloud Firestore. Документ містить персональні параметри користувача, зокрема рівень фізичної активності, вік, алергени, добову норму калорій та показники споживання калорій і макронутрієнтів.

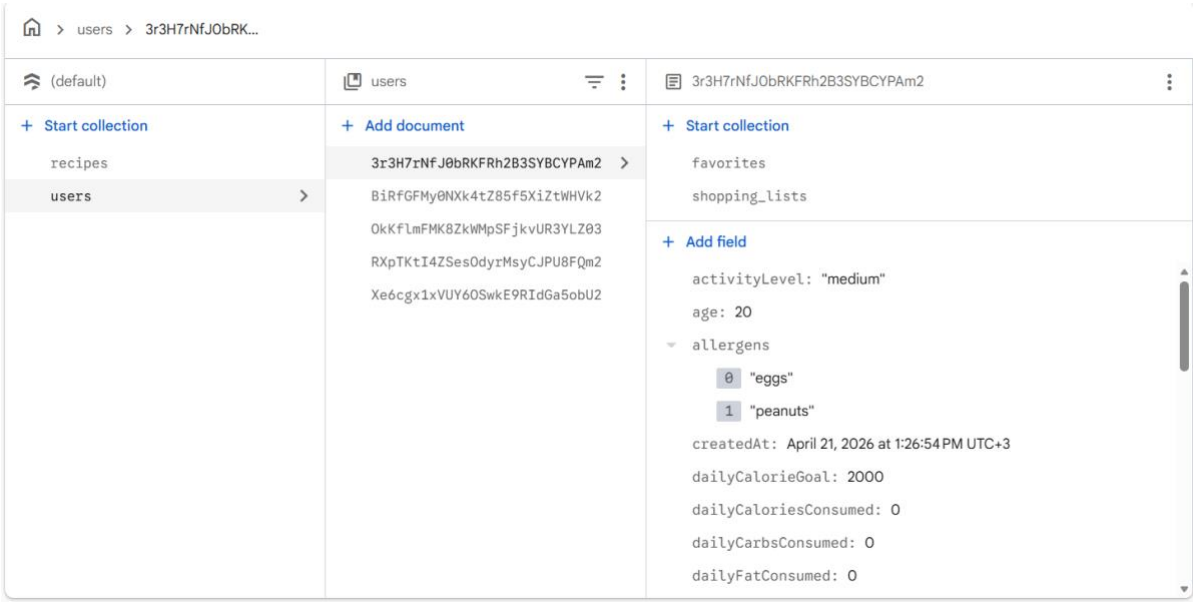

Рисунок 8 Структура документа користувача в колекції users у Cloud Firestore

На рис. 9 зображено приклад документа рецепта в колекції recipes у Cloud Firestore. У документі зберігаються основні дані про кулінарний рецепт.

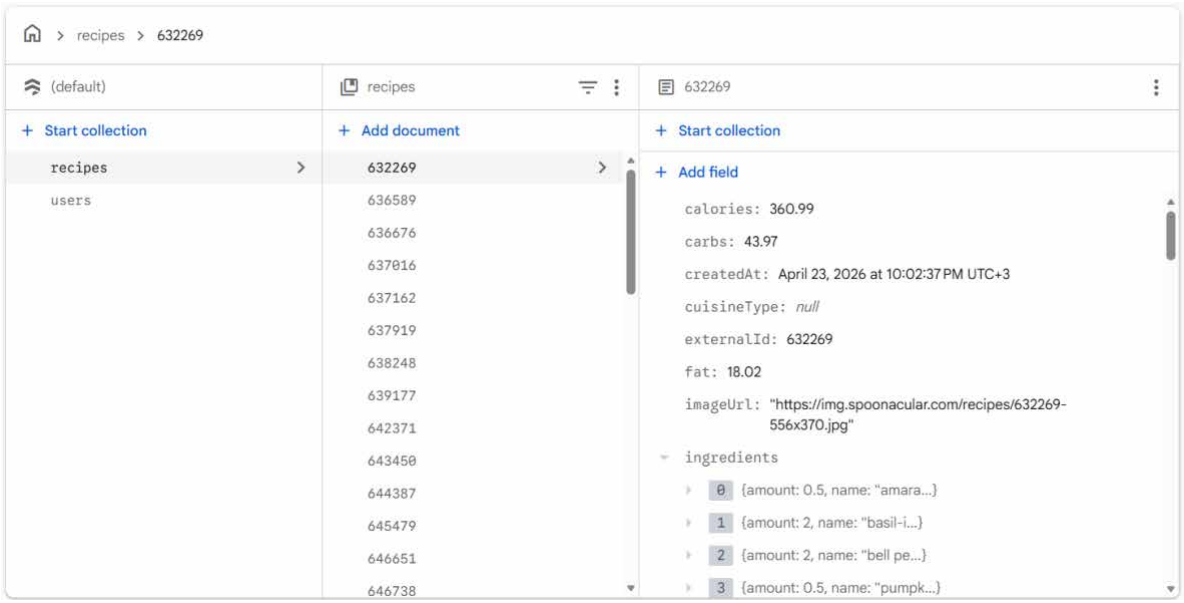

Рисунок 9 Структура документа рецепта в колекції recipes у Cloud Firestore

3 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Організаційна структура програмного забезпечення

3.1.1 Діаграма пакетів. Щоб показати структуру програмного забезпечення використовується діаграма пакетів.

Діаграма пакетів – це структурна діаграма UML, що показує структуру системи на рівні пакетів. На діаграмі пакетів, як правило, зображені такі елементи: пакет, елемент, залежність, імпорт елементів, імпорт пакетів, злиття пакетів[9].

На діаграмі виділено два основні рівні програмного забезпечення: Client та Server. Пакет Client відповідає за взаємодію користувача із системою. У ньому розміщено модуль User management, який забезпечує роботу з профілем користувача, його персональними параметрами, харчовими цілями та обмеженнями. Пакет Server містить основні функціональні модулі, які забезпечують обробку даних і реалізацію логіки системи. До нього входять модулі RecipeManagement, Meal Planning та Nutrition Analysis.

На рис. 10 зображено діаграму пакетів для системи яка розробоюється.

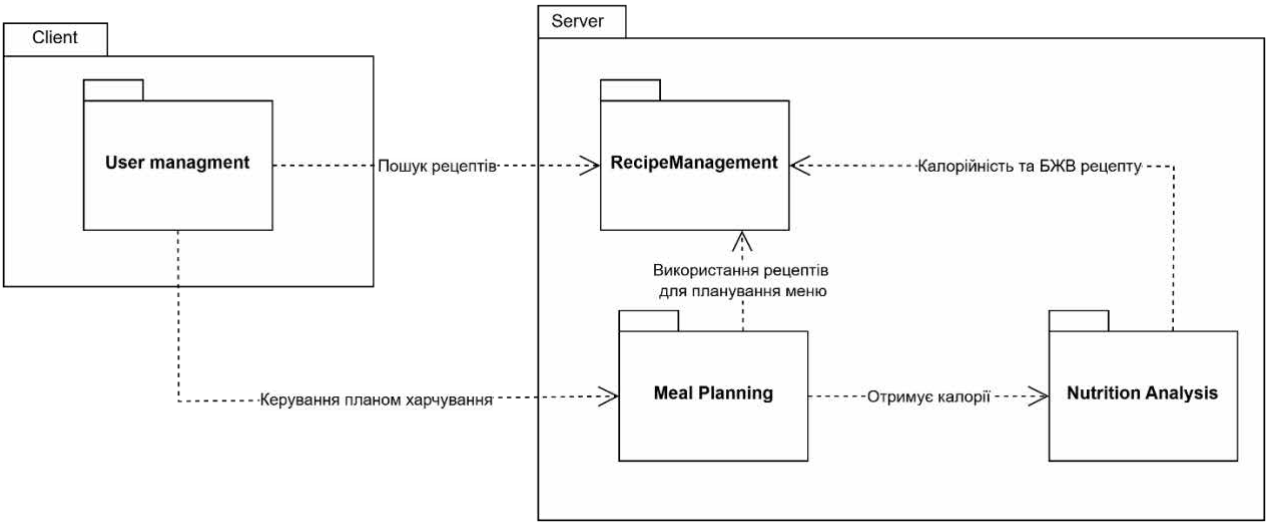

Рисунок 10 – Діаграма пакетів програмного забезпечення

3.1.2 Діаграма компонентів. Аби продемонструвати загальну структуру програмного забезпечення на рівні ключових функціональних частин будується діаграма копонентів[11]. Вона демонструє, з яких компонентів складається система, яку роль виконує кожен компонент та як взаємодіють між собою.

На діаграмі виділено систему RecipePlannerSystem, у межах якої розміщені основні компоненти програмного забезпечення: RecipePlannerFront, Catalogue, MealPlanSystem та RecipeDB.

На рис. 11 наведено діаграму компонентів, що демонструє структуру основних модулів програмного забезпечення та взаємозв'язки між ними.

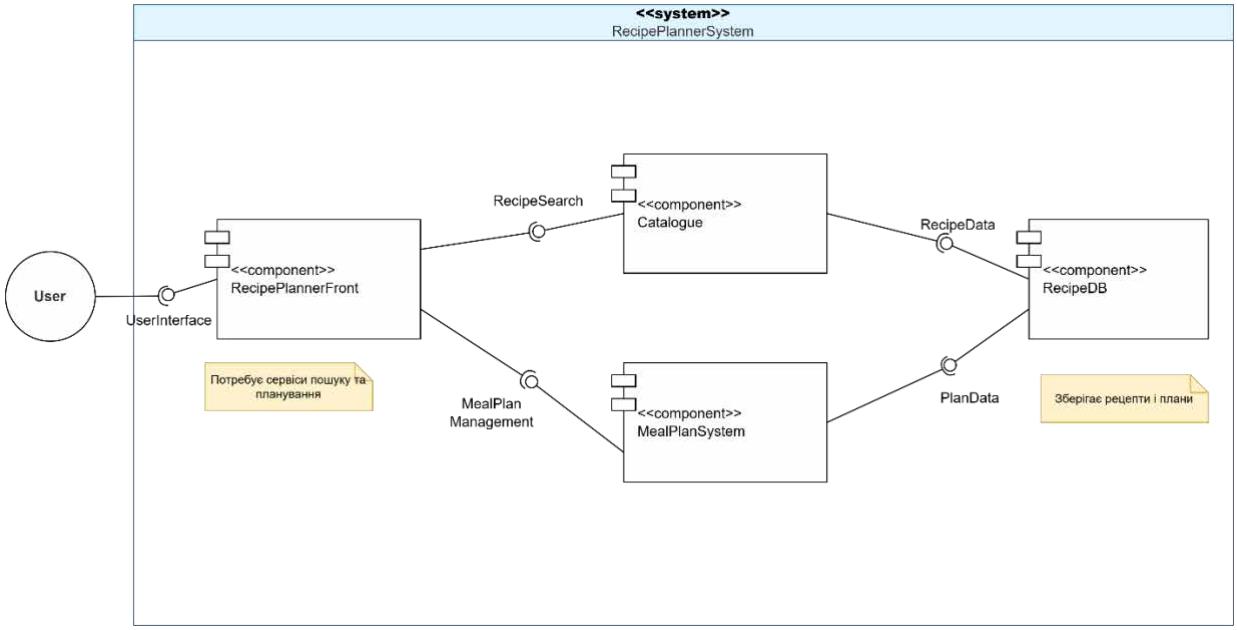

Рисунок 11 Діаграма компонентів програмного забезпечення

3.1.3 Діаграми простих кооперацій. Після побудови загальної діаграми класів доцільно розглядати окремі прості кооперації між класами. Проста кооперація описує не всю систему загалом, а окремий функціональний фрагмент[9].

Кооперація демонструє, які класи беруть участь у виконанні певної функції програмного забезпечення та які зв'язки між ними використовуються. У діаграмах кооперацій використовуються наступні типи зв'язків:

Залежність – пунктирна лінія із стрілкою, що демонструє як один клас тимчасово використовує інший клас для виконання певної дії.

Агрегація – зображається як суцільна лінія з порожнім ромбом. Такий зв’язок означає відношення типу «ціле – частина», але частина може існувати окремо від цілого.

Композиція – суцільна лінія із зафарбованим ромбом. Композиція є сильнішим типом зв’язку, ніж агрегація, тому що частина не може існувати самостійно.

Асоціація – загальний структурний зв’язок між класами, який демонструє, що об’єкти окремих класів пов’язані.

3.2.1 Кооперація «Рекомендації харчування». Відображає взаємодію класів User, DietRecommendation, Recipe та Product. Клас User містить персональні параметри користувача, на основі яких формується рекомендація. Клас DietRecommendation відповідає за створення рекомендацій відповідно до цілей харчування, а класи Recipe і Product використовуються для добору відповідних рецептів та аналізу їх складу.

Графічне представлення взаємодії зазначених класів у межах кооперації «Рекомендації харчування» наведено на рис. 12.

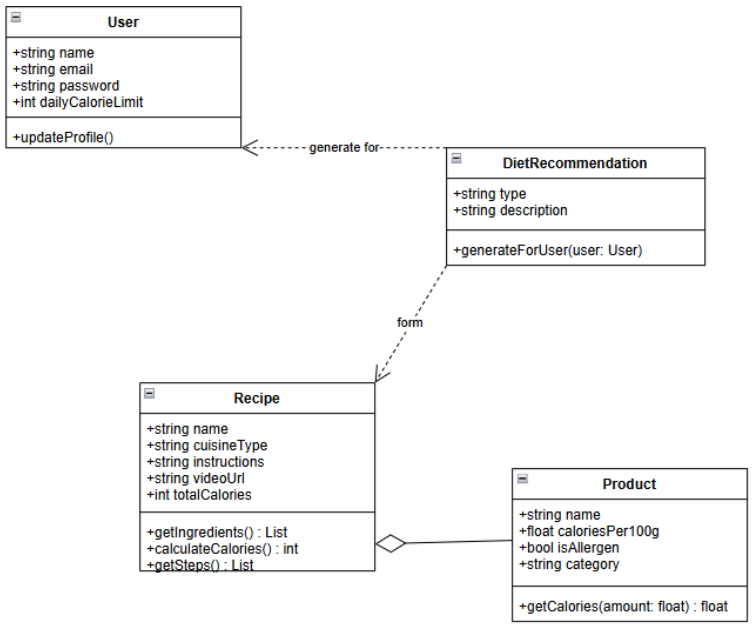

Рисунок 12 Кооперація класів для формування рекомендацій харчування

3.2.2 Кооперація «Формування списку покупок». Показує взаємодію класів User, ShoppingList, Recipe та Product. Користувач формує список покупок на основі обраних рецептів. Клас ShoppingList містить методи додавання, видалення та позначення продуктів як придбаних. Клас Product описує окремі продукти, які входять до списку.

Графічне представлення кооперації зображено на рис. 13

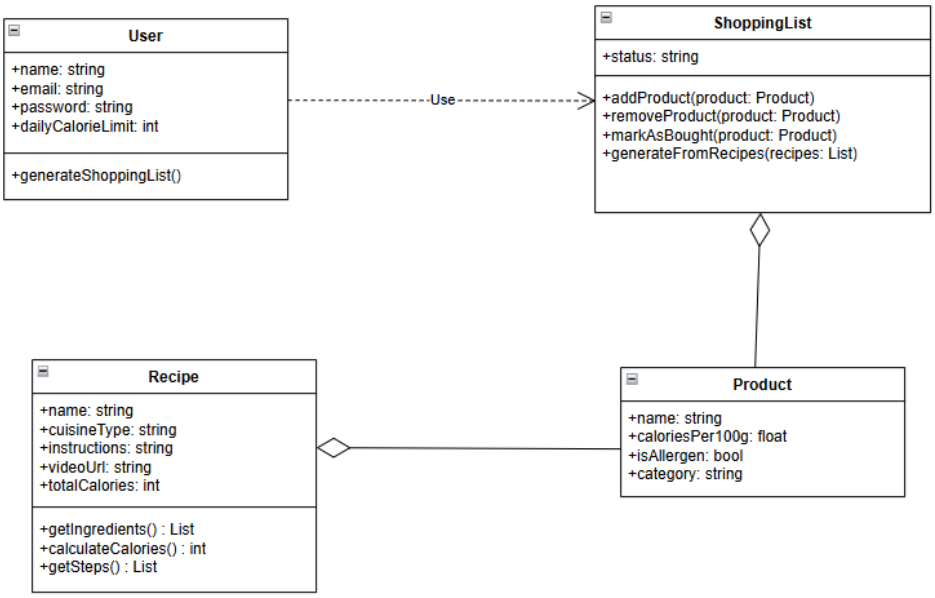

Рисунок 13 Кооперація класів для формування списку покупок

3.2.3 Кооперація «Відстеження калорій». Демонструє взаємодію класів CalorieTracker, Recipe та Product. CalorieTracker відповідає за збереження добової норми, спожитих та залишку калорій. Recipe використовується для визначення калорійності обраної страви, а Product – для врахування калорійності одиниці продукту (Рис. 14).

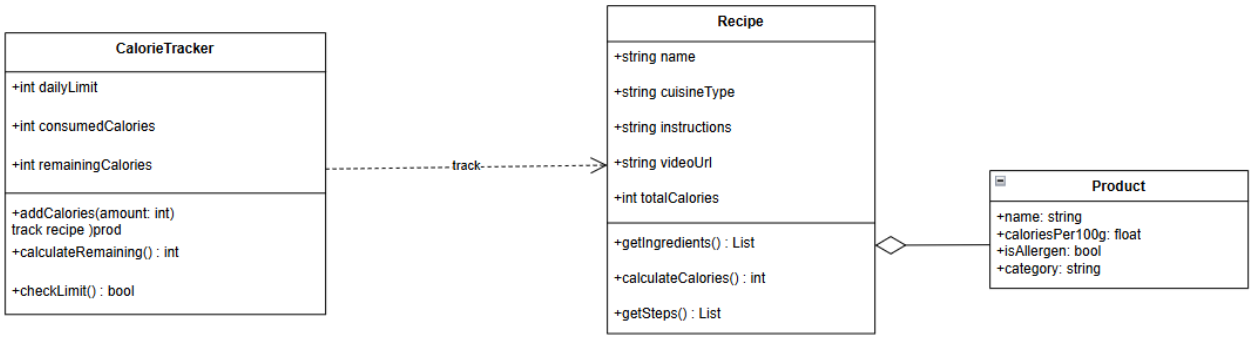

Рисунок 14 Кооперація класів для відстеження спожитих калорій

3.2 Вибір інструментарію для реалізації програмного забезпечення

Проектування структури програмного забезпечення важливий етап перед вибором інструментарію для подальшої реалізації системи. Для створення мобільного застосунку важливо обрати сучасний стек технологій який має відповідати функціональним вимогам системи та буде зручний для маштабування системи.

Для реалізації мобільного застосунку доцільно використати мову програмування Dart та фреймворк Flutter.

Dart – об’єктно-орієнтована мова програмування, використовується для створення клієнтських застосунків, підходить для розробки інтерфейсів користувача. Вона підтримує статичну типізацію, асинхронне програмування, роботу з класами, що є необхідним для реалізації моделей. Вона використовується для створення кросплатформених застосунків і чудово поєднується із фреймворком Flutter[15], [16].

Фреймворк Flutter – програмне середовище з відкритим вихідним кодом, яке дозволяє розробляти кросплатформені застосунки із єдиною кодовою базою. Основними перевагами Flutter є висока продуктивність, швидкий “гарячий перезапуск”, багата система віджетів для побудови інтерфейсів користувача, а також гнучка система стилізації та навігації. Завдяки інтеграції зі стандартними бібліотеками Dart, а також великій екосистемі пакетів для роботи з мережевими API, збереженням даних, управлінням станом тощо, Flutter став оптимальним вибором для проєктування застосунку, орієнтованого на подальший розвиток та масштабування[17].

У межах цієї роботи платформою розробки й демонстрації програмного забезпечення було обрано Android. Вибір Android як основної платформи зумовлений її широким використанням на мобільних пристроях, зручністю тестування застосунку та доступністю інструментів розробки. Крім того, Flutter дає можливість адаптувати щастосунок до інгих платформ.

3.3 Реалізація інтерфейсу користувача

Проєктування інтерфейсу користувача є одним із ключових етапів розробки мобільного програмного забезпечення. Саме від якості UI/UX значною мірою залежить зручність використання системи, ефективність виконання основних сценаріїв та загальне враження від застосунку.

Для побудови інтерфейсу застосовуються компоненти, відомі як віджети, які є невід’ємною частиною архітектури Flutter. Створення кожної сторінки або екрана застосунку базується на комбінуванні різних віджетів для забезпечення адаптивності інтерфейсу під різні роздільні здатності екранів та сценарії використання. Підхід орієнтований на максимальне забезпечення інтуїтивної послідовності дій користувача, зручність навігації та швидкий доступ до основного функціоналу.

На етапі проєктування інтерфейсу враховується цільова аудиторія, типова поведінка користувача, а також досвід найкращих практик у сфері мобільної розробки. Кожен з екранів виконує окрему роль у загальній взаємодії із застосунком, забезпечуючи користувачу змогу виконати авторизацію, ознайомитися із головною інформацією, переглянути персоналізоване меню, здійснювати пошук та вибір рецептів, формувати списки продуктів тощо. Розроблений інтерфейс показаний у розділі 4.5

3.4 Алгоритмізація та програмування програмних модулів

Ефективна розробка сучасних програмних систем базується на принципах алгоритмізації та модульного програмування. На даному етапі прокатування системи для персоналізованого планування меню та підбору рецептів, особливе значення має формалізація процесів у вигляді алгоритмів та їх структуризація у програмні модулі.

3.4.1 Моделювання бізнес логіки. Моделювання бізнес-процесів системи дозволяє краще розуміти алгоритми дій та передбачати різні сценарії. Для цього використовуються блок-схеми. На рис. 15 наведено блок-схему алгоритму розрахунку персоналізованої добової норми калорій. Алгоритм описує повну послідовність кроків - від моменту, коли користувач вводить свої персональні показники, до отримання індивідуальної рекомендації щодо споживання калорій.

Алгоритм запускається у момент, коли користувач звертається до функції розрахунку добової норми калорій. Система отримує дані які користувач додав під час реєстрації. Отриманий набір даних проходить валідацію за бізнес-правилами, які перевіряють обов’язковість усіх параметрів і допустимість числових значень.

Обчислення складається з трьох послідовних підпроцесів. У межах першого з них на основі статі користувача обирається відповідна формула Міффліна-Сан-Жеора, за якою обчислюється базальний рівень метаболізму BMR, різні для жінок і чоловіків.

На наступному етапі отримане значення множиться на коефіцієнт фізичної активності - 1,2 для низького рівня, 1,55 для середнього та 1,725 для високого, що дозволяє визначити загальні добові енерговитрати TDEE.

Третій підпроцес здійснює корекцію значення TDEE відповідно до обраної цілі: мінус 300 ккал у разі схуднення, плюс 300 ккал для набору ваги, без змін - для підтримки ваги та режиму здорового харчування.

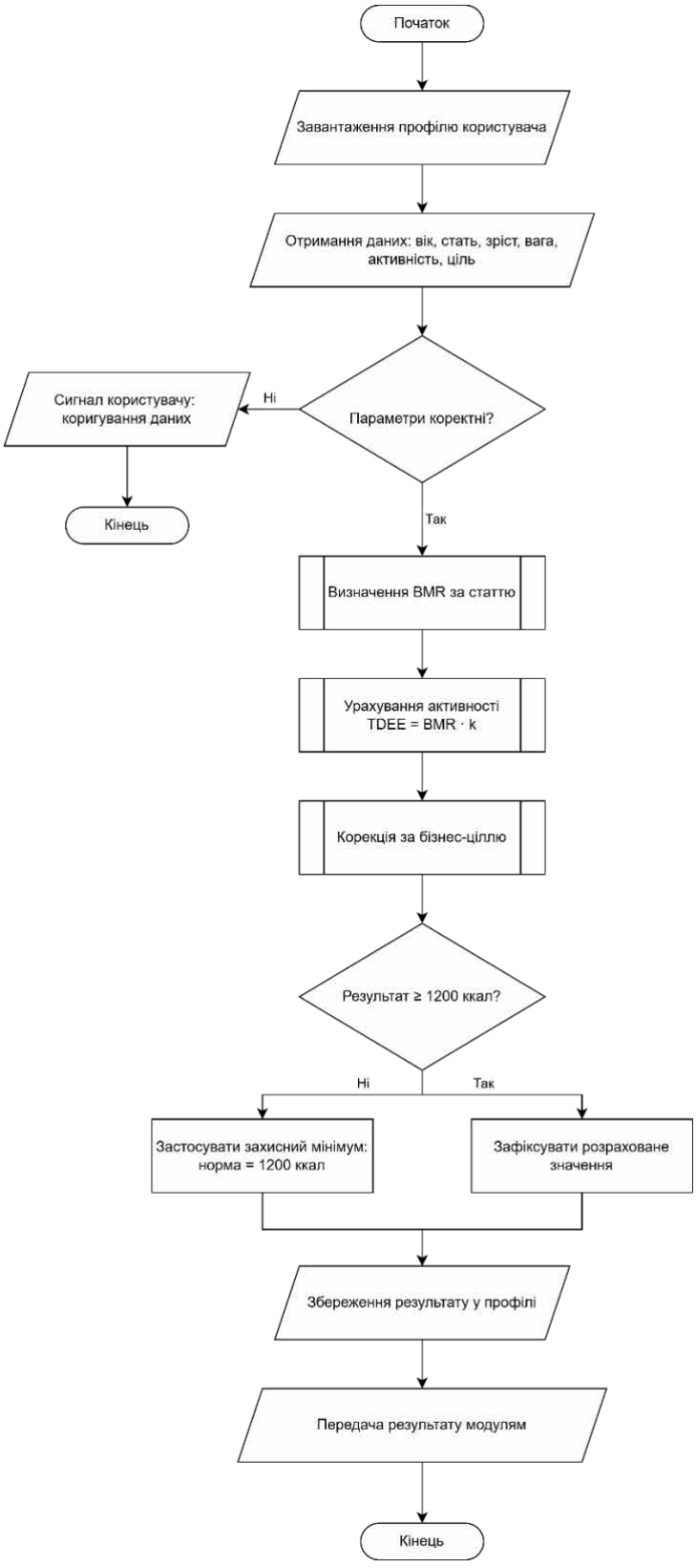

Рисунок 15 Блок-схема алгоритму розрахунку персоналізованої добової норми калорій

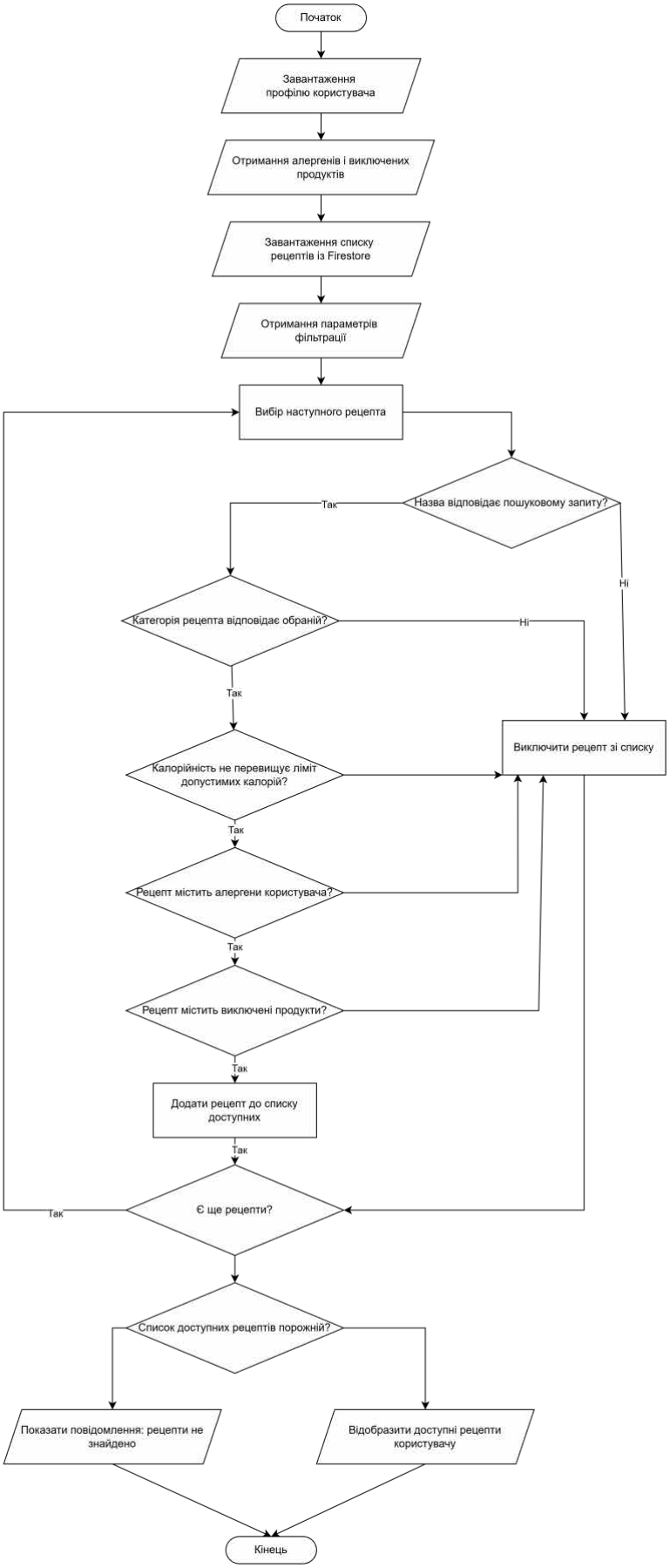

Рисунок 16 Блок-схема алгоритму підбору рецептів зшідно вподобань користувача

На рис. 16 наведено блок-схему алгоритму персоналізованого підбору рецептів. Процес починається з завантаження профілю користувача, отримання інформації про алергени та виключені продукти, після чого система отримує список рецептів і параметри фільтрації.

Далі кожен рецепт обробляється за наступними умовами: відповідність назви пошуковому запиту, належність до вибраної категорії, відсутність алергенів і продуктів, які користувач виключив зі свого раціону.

Якщо рецепт не відповідає хоча б одній із заданих умов, він виключається зі списку доступних. Якщо рецепт проходить усі перевірки, він додається до списку рекомендованих рецептів.

Після обробки всіх рецептів система перевіряє, чи сформовано список варіантів: якщо список порожній, користувач отирмає повідомлення про відсутність відповідних рецептів, а якщо рецепти знайдено – відображаються для подальшого перегляду.

На рис. 17 наведено блок-схему алгоритму формування плану харчування користувача. Алгоритм розпочинається отриманням вхідних даних: поточної дати, харчової цілі та персональних обмежень. Далі система визначає поточний тиждень і перевіряє, чи вже існує план харчування на цей період. У випадку існування плану, він завантажується з бази даних; якщо ні – створюється новий документ тижня та формується структура із семи днів і відповідних прийомів їжі. Після цього система обирає рецепт. Якщо доступних рецептів не знайдено, користувачу виводиться відповідне повідомлення, і процес завершується.

Рецепти знайдено, відповідно виконується підпроцес добору рецепта після чого формується запис MealEntry і додається до дня та прийому їжі. Після заповнення всіх необхідних прийомів їжі план зберігається, оновлюється локальна структура даних, а результат передається модулям інтерфейсу, калорійності та списку покупок.

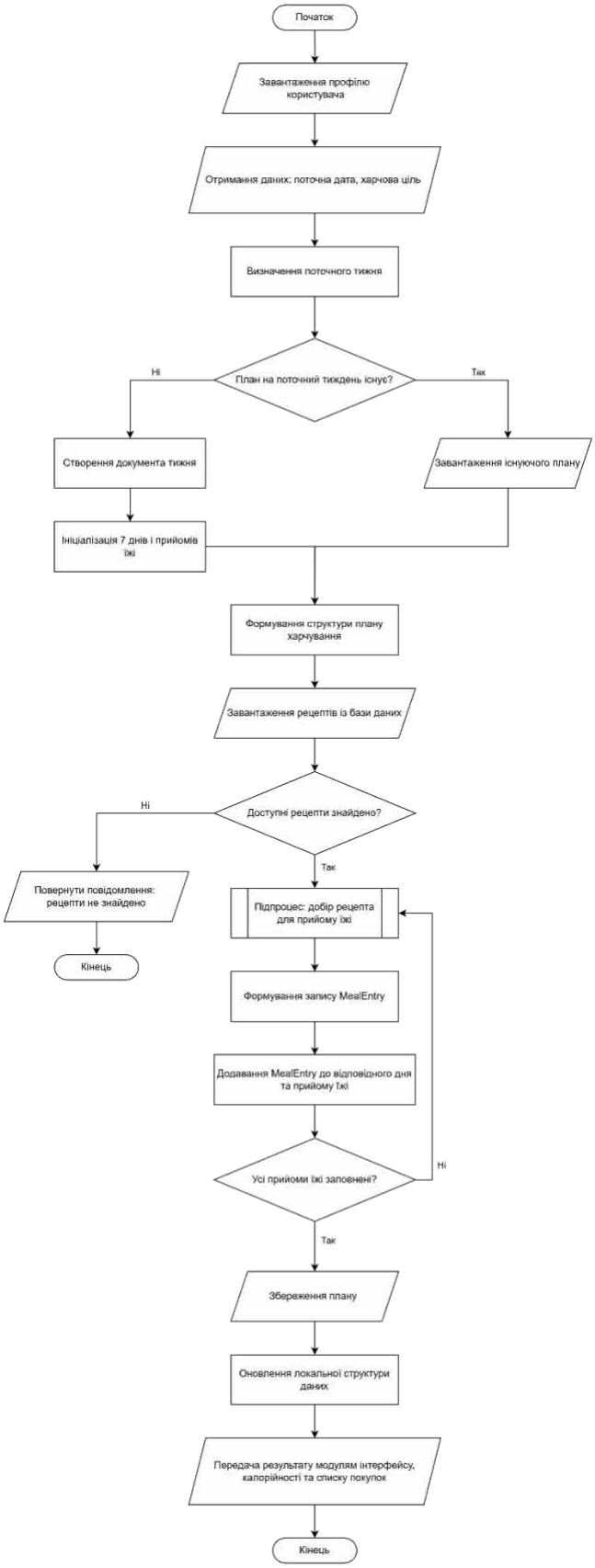

Рисунок 17 Блок-схема алгоритму автоматичного підбору персонального меню

3.5.1 Забезпечення інтерфейсу з БД. Для коректної роботи інтерфейсу користувача мобільний застосунок повинен мати доступ до даних, які зберігаються у базі даних. У розробленій системі як хмарне сховище використовується Cloud Firestore, а для ідентифікації користувача застосовується Firebase Authentication. Такий підхід дозволяє зберігати персональні дані користувача, профіль харчування, список покупок, обрані рецепти, план харчування та інші дані, необхідні для роботи інтерфейсу.

Підключення до Firebase виконується під час запуску застосунку у файлі main.dart. Перед відображенням головного інтерфейсу система ініціалізує Firebase. Лістинг програмного коду зображений на рис. 18

Рисунок 18 Ініціалізація Firebase під час запуску мобільного застосунку

Наведений фрагмент коду демонструє початкове підключення застосунку до Firebase. Метод WidgetsFlutterBinding.ensureInitialized() забезпечує підготовку середовища Flutter перед виконанням асинхронних операцій. Після цього викликається Firebase.initializeApp(), який ініціалізує сервіси Firebase відповідно до параметрів поточної платформи[18].

Після ініціалізації Firebase інтерфейс отримує можливість звертатися до сервісів автентифікації та бази даних. Для відокремлення логіки роботи з БД від візуальної частини застосунку використовуються сервісні класи. Вони виконують запити до Cloud Firestore, отримують необхідні дані.

Відповідно інтерфейс не працює з базою даних напряму, а отримує вже підготовлені дані через окремий програмний шар.

4 РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ

4.1 Архітектура системи

Для розробленого мобільного застосунку було використано клієнт-серверну архітектуру, у межах якої функціонал клієнтської частини відокремлений від серверної інфраструктури.

Така організація системи відповідає принципам сучасної мобільної розробки, забезпечує розмежування обов'язків між компонентами, гнучкість у масштабуванні й незалежну еволюцію клієнтського і серверного шарів.

Взаємодія між клієнтом і сервером здійснюється через захищені HTTPS-з'єднання за допомогою офіційних SDK Firebase для Flutter, що забезпечує шифрування переданих даних і відповідність вимогам безпеки. Структуру системи зображено на діаграмі розгортання (рис. 19).

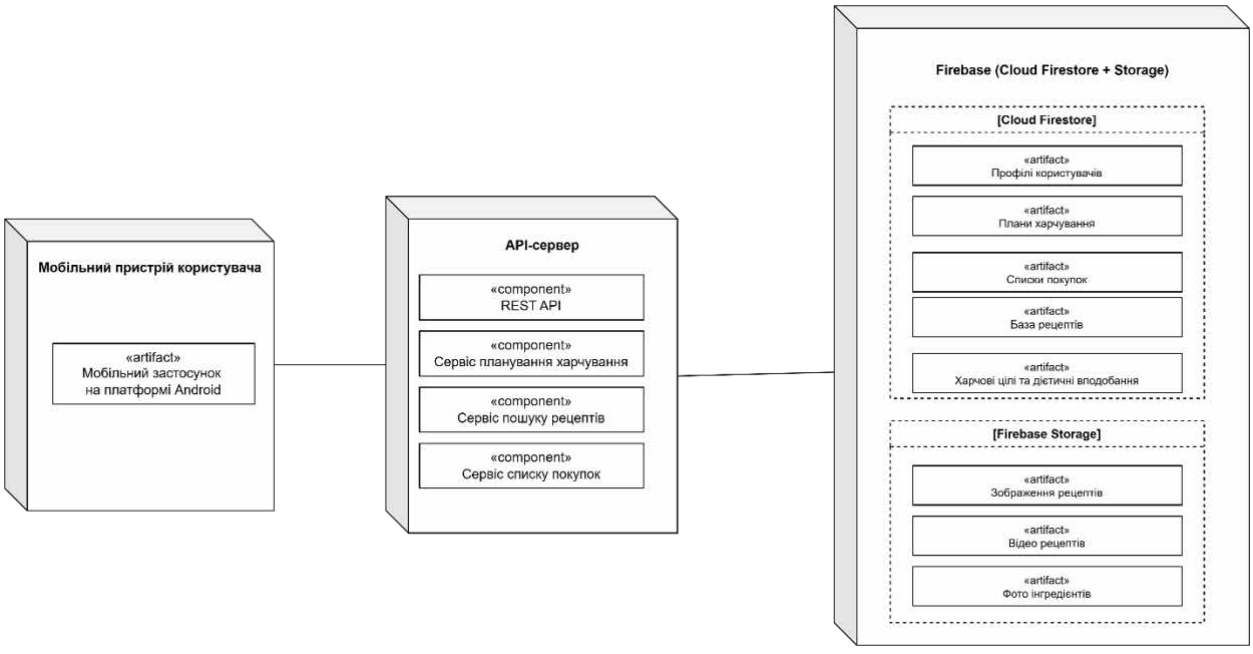

Рисунок 19 Діаграма розгортання

4.2 Тестування системи

Тестування програмного забезпечення є важливою складовою життєвого циклу розробки і виконує роль контролю якості, який використаний, щоб виявити дефекти, перевірити відповідність функціональним вимогам та оцінити готовність системи до експлуатації[20].

4.2.1 Застосовані види тестування. Щоб ефективно протестувати програмне забезпечення було використано декілька рівнів і видів тестування, кожен з яких спрямований на перевірку окремого аспекту функціонування системи:

- Модульне тестування – перевірялася коректність роботи окремих функцій і класів у ізоляції від інших компонентів системи. Особлива увага приділялася модулю NutritionCalculator, який містить ключову обчислювальну логіку (розрахунок BMR, TDEE, макронутрієнтів).
- Інтеграційне тестування – перевірялася взаємодія клієнтського застосунку з Firebase, аутентифікації, читання та запису у Cloud Firestore.
- Функціональне тестування – визначалися відповідності поведінки системи функціональним вимогам розроблюваного програмного забезпечення.
- Тестування користувацького інтерфейсу – встановлення коректності відображення елементів інтерфейсу та оцінка зручності навігації.
- Ручне приймальне тестування – перевірка готовності продукту до релізу шляхом проходження типових сценаріїв користувача.

4.2.2 Процес проведення модульного тестування. Модульне тестування у середовищі Flutter здійснюється засобами вбудованого пакету `flutter_test`[21], що дає можливость для написання тестових випадків, групування їх у тестові набори та автоматичного виконання з виведенням результатів у консоль.

У межах модульного тестування `NutritionCalculator` було розроблено набір тестових випадків, основні з яких наведено у таблиці 4.

Таблиця 4

Назва тесту	Вхідні дані	Очікуваний результат
1	**2**	**3**
Розрахунок для чоловіка з ціллю схуднення	age=30, gender=male, height=180, weight=85, activity=medium, goal=weight_loss	≈ 2236 ккал
Розрахунок для жінки з ціллю підтримки ваги	age=25, gender=female, height=165, weight=60, activity=low	≈ 1610 ккал
Перевірка нижньої межі	age=80, gender=female, height=150, weight=40, activity=low, goal=weight_loss	1200 ккал (захисний мінімум)
Некоректні дані - від’ємний вік	age=-5, gender=male, height=180, weight=80	NULL
Відсутність обов’язкового параметра	gender=null, інші параметри коректні	NULL
Невідоме значення статі	gender="unknown"	NULL
Розрахунок для категорії «інше»	gender=other, age=30, height=170, weight=70	усереднене значення

Приклади реалізованих тестів наведені у ДОДАТОК Б. Результати проведеного тестування показали, що програмне забезпечення коректно виконує усі функціональні сценарії.

4.3 Вимоги до апаратного та програмного забезпечення

Описана у розділі 4.1 архітектура передбачає виконання основної обчислювальної частини на стороні клієнтського пристрою, тоді як серверні ресурси, що споживає система, забезпечуються хмарною платформою Firebase. Завдяки такому підху суттєво знижуються загальні витрати на впровадження та супровід програмного продукту.

4.3.1 Апаратні вимоги. Для коректного функціонування програмного забезпечення на стороні користувача необхідні такі мінімальні апаратні вимоги:

Таблиця 5

Компонент	**Мінімальне значення**	**Рекомендоване значення**
1	**2**	**3**
Тип пристрою	Смартфон на платформі Android	Смартфон або планшет на платформі Android
Процесор	ARMv7, 1,4 ГГц, 4 ядра	ARM64, 2,0 ГГц, 8 ядер
Оперативна пам’ять (RAM)	2 ГБ	3 ГБ і більше
Внутрішня пам’ять	150 МБ вільного простору	500 МБ і більше
Дисплей	4,7″, 720×1280 пікселів	5,5″ і більше, 1080×1920 пікселів
Мережевий модуль	Wi-Fi або 3G	Wi-Fi або 4G/5G

З боку серверної частини апаратні вимоги забезпечуються інфраструктурою компанії Google і користувачу або розробнику не потрібують окремого розгортання обладнання.

4.3.3 Програмні вимоги. Програмні вимоги до клієнтського пристрою користувача, які вимагає застосунок продемонстровані у табл. 6.

Таблиця 6

Компонент	Характеристика
1	**2**
Операційна система	Android версії 6.0 (Marshmallow, API level 23) і вище[22]
Сервіси Google Play	Актуальна версія для забезпечення коректної роботи Firebase SDK
Підключення до мережі Інтернет	Активне з’єднання для аутентифікації, синхронізації даних із Cloud Firestore [19] та завантаження зображень із Firebase Storage
Дозволи застосунку	• Доступ до мережі , отримання стану мережі ; • інші дозволи запитуються контекстно при використанні відповідних функцій

Наведені програмні вимоги є достатніми для коректної роботи основних функцій застосунку, зокрема авторизації користувача, перегляду рецептів, формування плану харчування та синхронізації даних із хмарною базою даних. Оскільки застосунок орієнтований на мобільне використання, стабільне підключення до мережі Інтернет є важливою умовою для своєчасного оновлення даних і завантаження медіафайлів.

Серверна частина не потребує встановлення додаткового програмного забезпечення з боку користувача чи розробника, оскільки сервіси Firebase надаються у формі готової хмарної платформи. Інструменти, які використовуються розробником для супроводу та оновлення системи, наведено у табл. 7.

Вимоги повністю відповідають типовій конфігурації сучасних Android-пристроїв і не створять обмежень для потенційних користувачів системи.

Таблиця 7

Інструмент	**Призначення**
1	**2**
Firebase Console	Веб-інтерфейс для управління проектом, базою даних, користувачами та правилами безпеки
Flutter SDK (версія 3.x і вище)	Фреймворк для розробки, оновлення та компіляції клієнтського застосунку
Dart SDK (версія 3.x і вище)	Мова програмування для реалізації бізнес-логіки та інтерфейсу користувача
Android SDK (мінімум API level 23)	Збирання інсталяційних пакетів формату APK/AAB

4.4 Склад інсталяційного пакету

Після завершення процесу розробки та тестування програмного забезпечення необхідно сформувати готовий до розповсюдження інсталяційний пакет.

Інсталяційний пакет являє собою набір файлів, що містять усі необхідні компоненти для встановлення та подальшого функціонування програмного продукту на цільовому пристрої кінцевого користувача.

4.4.1 Формат інсталяційного пакету. Для розповсюдження мобільних застосунків на платформі Android прийнято стандартний формат APK (Android Package Kit) [23]- архівний файл, що містить скомпільований програмний код, ресурси, метадані та цифровий підпис застосунку. Структурно APK-файл є архівом ZIP-формату.

4.4.2 Процес формування інсталяційного пакету. Формування APK-файлу здійснювалося засобами Flutter SDK у режимі релізної збірки, що передбачає компіляцію Dart-коду в нативний машинний код та виконання оптимізації обсягу й продуктивності.

Послідовність операцій включала такі етапи:

- Підготовка проєкту - налаштування файлу pubspec.yaml з переліком залежностей, версією застосунку та ідентифікатором пакету;
- Конфігурація релізного підпису - створення ключа цифрового підпису та його реєстрація у файлі конфігурації android/app/build.gradle;
- Виконання команди збірки flutter build apk --release[23], у результаті якої Flutter SDK здійснює компіляцію вихідного коду, об’єднання ресурсів та генерацію підписаного APK-файлу;
- Розміщення сформованого файлу за стандартним шляхом build/app/outputs/flutter-apk/app-release.apk;
- Верифікація цілісності пакету та перевірка відповідності цифрового підпису перед розповсюдженням.

4.4.3 Склад готового інсталяційного пакету. Кінцевий інсталяційний пакет, що передається користувачам системи, включає такі складові:

- app-release.apk - підписаний інсталяційний файл застосунку у форматі APK;
- Файл README - стисла інструкція щодо встановлення та запуску застосунку;
- Контрольна сума (SHA-256) - хеш-код для перевірки цілісності APK-файлу.

Розмір сформованого APK-файлу становить 45–55 МБ

4.4.4 Встановлення інсталяційного пакету на пристрій користувача. Сформований APK-файл придатний для встановлення на будь-який Android-пристрій, що відповідає вимогам, наведеним у розділі 4.3.

Процедура встановлення:

- перенесення файлу app-release.apk на пристрій користувача;
- дозвіл встановлення застосунків з невідомих джерел у налаштуваннях безпеки Android;
- запуск файлу APK через файловий менеджер пристрою;
- підтвердження запитуваних дозволів та автоматичне встановлення застосунку.

Після завершення встановлення застосунок готовий до використання за умови наявності активного підключення до мережі Інтернет для аутентифікації та синхронізації даних з хмарною базою Firebase.

4.5 Дослідна експлуатація

Дослідна експлуатація розробленого мобільного застосунку Meal Planner проводилася на реальному Android-пристрої з метою перевірки коректності виконання основних функціональних сценаріїв і визначення зручності взаємодії користувача із системою.

Початок взаємодії із програмним продуктом починається з стартового вікна мобільного застосунку (рис. 20).

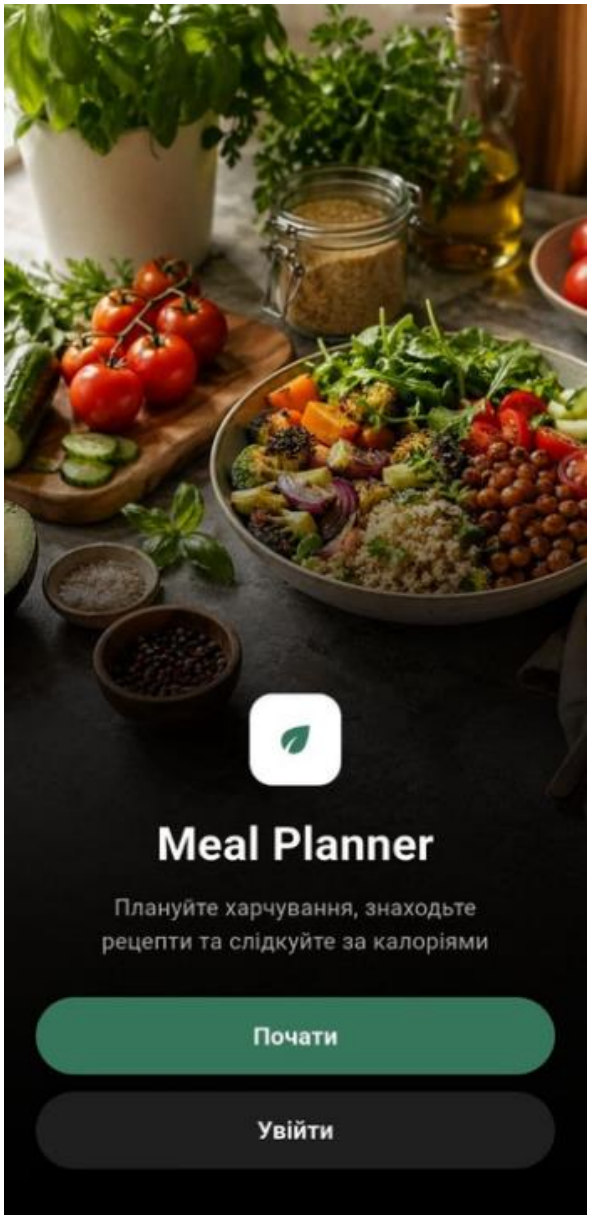

Рисунок 20 – Стартове вікно мобільного застосунку Meal Planner

Далі користувач може обрати опцію входу в систему чи реєстрації. Для входу в систему потрібно використати особисту пошту та користувача. При

реєстрації вказується: особиста пошта користувача, прізвище та заповнюється особистий профіль харчування.

Наступним важливим кроком є формування особистого профілю харчування, що допоможе підібрати рецепти та сформування меню згідно цілей користувача.

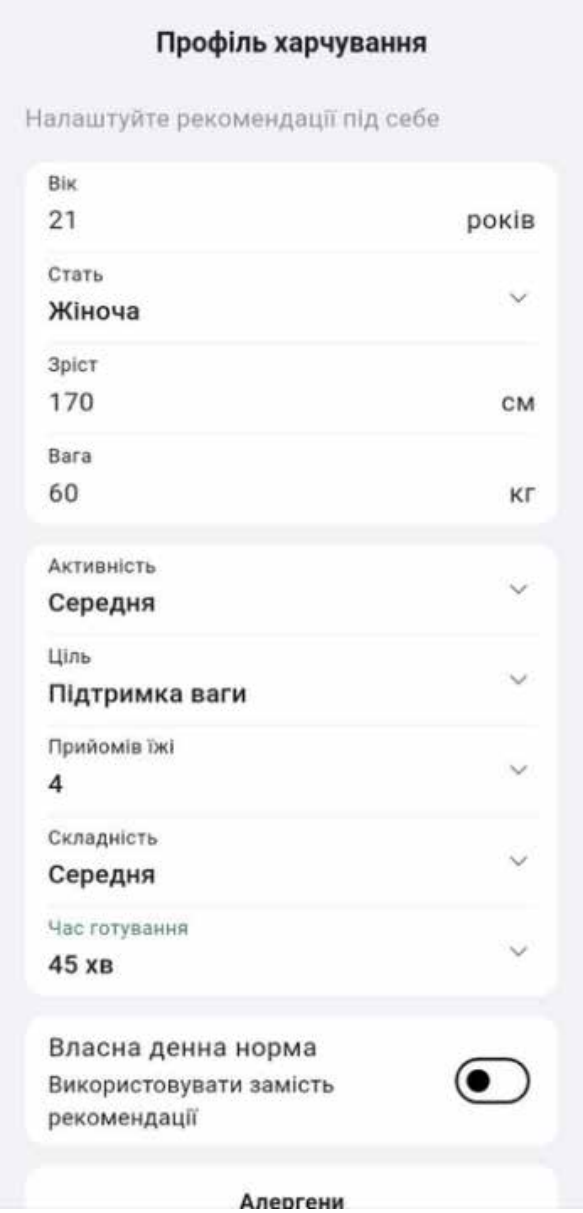

Рисунок 21 Вікно для заповнення профілю харчування(1 частина)

На рис. 21 зображений процес заповнення профілю харчування де користувач вказує персональні параметри для формування індивідувальних рекомендацій, харчові цілі. Також є можливість встановити власноруч норму калорій та БЖУ, за вподобаннями особи.

Користувач вказує алергени та продукти, що бажає виключити з рекомендацій та особистого меню(рис. 22).

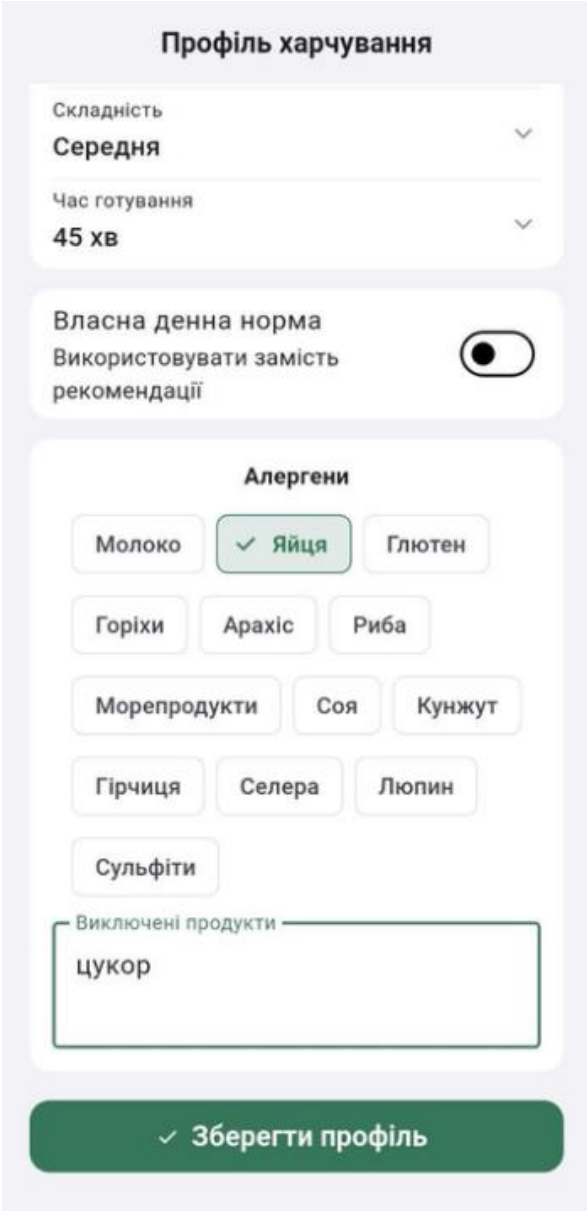

Рисунок 22 Вікно для заповнення профілю харчування(2 частина)

Після реєстрації або входу в систему користувач потрапляє на головну сторінку застосунку де може переглянути рекомендації рецептів, згідно заповненого профіля харчування та рекомендовану норму калорій та БЖВ. Також реалізована навігаційна панель у нижній частині екрану забезпечує швидкий перехід між основними розділами застосунку(рис. 23).

Зазначені обмеження застосовуються при підборі рецептів – система автоматично виключає страви, до складу яких входять небажані інгредієнти або харчові алергени.

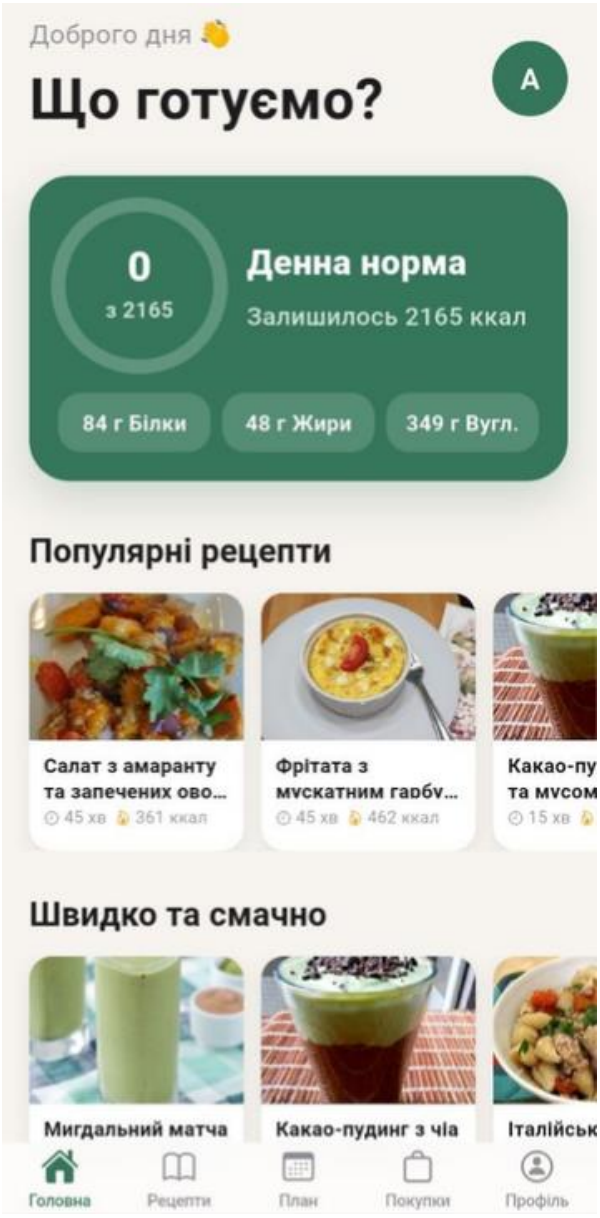

Рисунок 23 Головне вікно застосунку

Також на рис. 23 можна побачити наглядно роботу модуля NutritionCalculator - обчислену добову норму калорій з урахуванням введених персональних показників і обраної цілі.

На головному вікні можна обрати реуеомендавані рецепти згідно вподобань, але зручнішим для пошуку є вікно для пошуку рецептів (рис. 24).

Розділ каталогу рецептів відображає перелік рекомендованих страв із короткою інформацією про калорійність, час приготування та категорію. Користувач має змогу здійснювати пошук за назвою та фільтрувати рецепти за енергетичною цінністю, типом прийому їжі або харчовими обмеженнями.

Додати рецепт до улюблених, щоб не втратить улюблений рецепт.

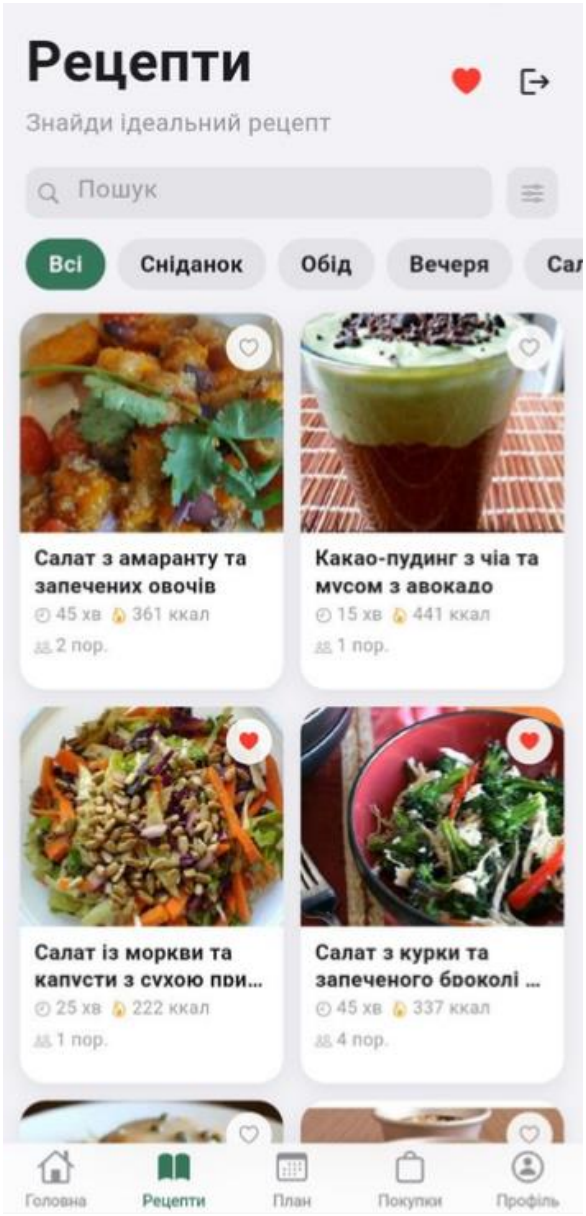

Рисунок 24 Вікно для пошуку рецептів та перегляду рекомендованих

При натисканні на рецепт у каталозі або на головній сторінці (рис. 24) користувач переходить на детальний екран, який містить зображення страви, перелік інгредієнтів із зазначенням маси, поетапну інструкцію приготування та повну поживну цінність. (рис. 25)

Рисунок 25 Вікно для перегляду рецепту

На цьому ж екрані користувач може обрати продукти, та додати їх до продуктового кошика або персоналізованого меню.

Екран планування меню відображає сформований раціон користувача на обраний день, розподілений за прийомами їжі (сніданок, обід, вечеря, перекуси). Поряд з кожним прийомом їжі відображається сучарна калорійність і відсоток виконання денної норми. Користувач має змогу редагувати склад меню, додаванням або видаленням страви(рис. 26).

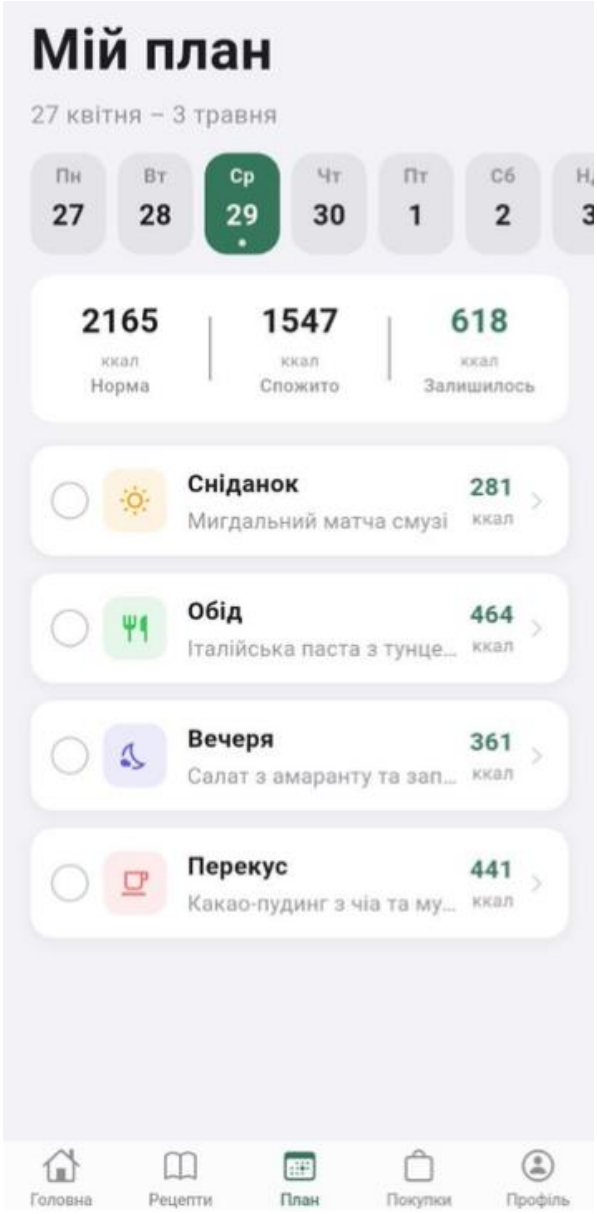

Рисунок 26 Вікно для формування персоналізованого меню

Список покупок користувач формує особисто або система формує його, є можливість додати продукт з стрінки рецепту чи власноруч додати товар та його кількість. Він доступний з будь якого пристрою після авторизації(рис. 27).

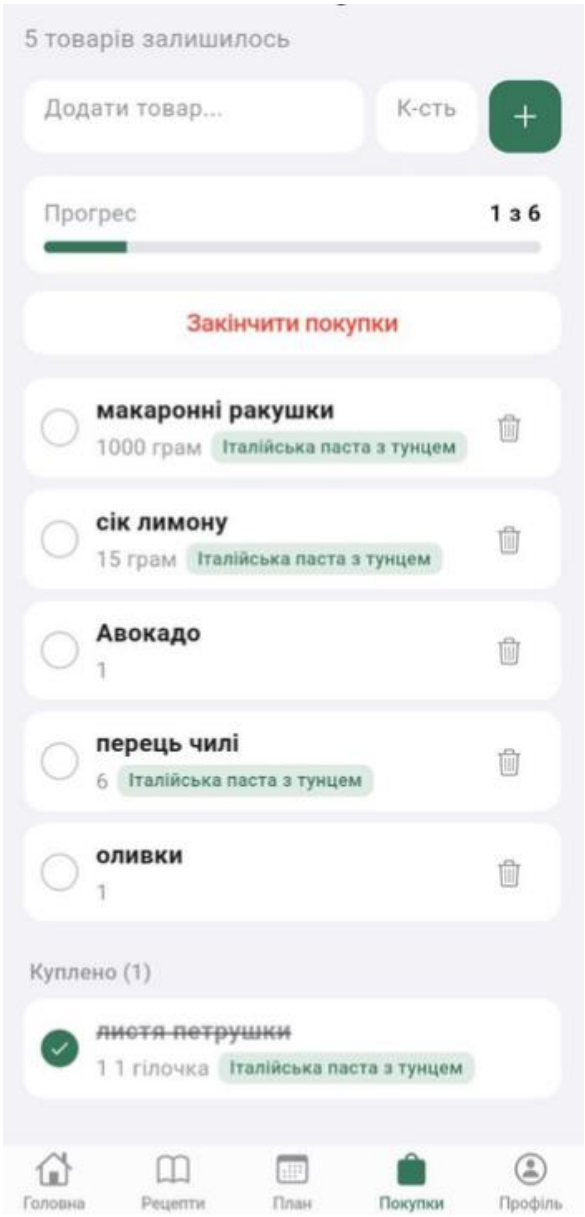

Рисунок 27 Вікно з доданими продуктами до проудктового кошика

Користувач також може змінити харчові цілі та рекомендації будуть оновлені відповідно до оновлень в особистому пофілі. Зміни синхронізуються з хмарною базою даних в реальному часі.

За результатами дослідної експлуатації доведено, що розроблений мобільний застосунок коректно виконує функціональні сценарії, описані у вимогах: реєстрація користувача, формування профілю харчування, розрахунок добової норми калорій, підбір та перегляд рецептів, планування меню та формування списку покупок.

ВИСНОВКИ

У процесі виконання дипломного проєкту «Програмне забезпечення для персоналізованого підбору кулінарних рецептів і планування харчування» було:

- проаналізовано об’єкт дослідження;
- розглянуто існуючі програмні рішення у сфері планування меню та підбору рецептів, визначено їхні переваги й недоліки;
- визначено поствлене завдання та змодельовано систему у вигляді діаграм прецедентів, активності, послідовності;
- описано інформаційне забезпечення системи та побудовано логічну модель бази даних, що відображає основні сутності;
- обґрунтовано вибір нереляційної бази даних Cloud Firestore у складі хмарної платформи Firebase, а також мови програмування Dart і фреймворка Flutter для реалізації клієнтської частини застосунку;
- спроєктовано бізнес-логіку системи у вигляді блок-схем, серед яких ключовим є алгоритм розрахунку персоналізованої добової норми калорій;
- реалізовано інтерфейс користувача засобами фреймворка Flutter, описано програмні модулі застосунку, зв’язок з базою даних Cloud Firestore;
- визначено вимоги до апаратного та програмного забезпечення з боку клієнтського пристрою користувача, описано вимоги до серверної частини, реалізованої на базі сервісів Firebase;
- представлено склад інсталяційного пакету у форматі APK та описано процедуру встановлення застосунку на Android-пристрій;
- продемонстровано процес роботи програмного забезпечення з використанням знімків екрану, що ілюструють основні сценарії взаємодії користувача із застосунком.

Розробленене «Програмне забезпечення для персоналізованого підбору кулінарних рецептів і планування харчування» у межах дипломного проєкту надає користувачу можливість в автоматичному режимі отримувати персоналізовані рекомендації щодо щоденного раціону, добирати кулінарні рецепти з огляду на індивідуальні параметри й харчові обмеження, складати план харчування на обраний період і формувати перелік необхідних продуктів для здійснення покупок.

Таким чином, поставлена на початку роботи мета була досягнута: завдяки впровадженню створеного програмного забезпечення спрощується щоденний підбір страв, забезпечується контроль калорійності й балансу нутрієнтів та враховуються особисті дієтичні потреби кожного користувача.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. World Health Organization. Healthy diet [Електронний ресурс]. – Режим доступу: https://www.who.int/news-room/fact-sheets/detail/healthy-diet – Дата звернення: 16.05.2026.
2. Mifflin M. D., St Jeor S. T., Hill L. A., Scott B. J., Daugherty S. A., Koh Y. O. A new predictive equation for resting energy expenditure in healthy individuals [Електронний ресурс] // The American Journal of Clinical Nutrition.– Режим доступу: https://pubmed.ncbi.nlm.nih.gov/2305711/ – Дата звернення: 16.05.2026.
3. Frankenfield D., Roth-Yousey L., Compher C. Comparison of predictive equations for resting metabolic rate in healthy nonobese and obese adults [Електронний ресурс] // Journal of the American Dietetic Association. – Режим доступу: https://pubmed.ncbi.nlm.nih.gov/15883556/ – Дата звернення: 16.05.2026.
4. Trumbo P., Schlicker S., Yates A. A., Poos M. Dietary reference intakes for energy, carbohydrate, fiber, fat, fatty acids, cholesterol, protein, and amino acids [Електронний ресурс] // Journal of the American Dietetic Association.– Режим доступу: https://pubmed.ncbi.nlm.nih.gov/12449285/ – Дата звернення: 16.05.2026.
5. Abeltino A., Riente A., Bianchetti G. et al. Digital applications for diet monitoring, planning, and precision nutrition for citizens and professionals: a state of the art [Електронний ресурс] // Nutrition Reviews. – 2025. – Режим доступу: https://pubmed.ncbi.nlm.nih.gov/38722240/ – Дата звернення: 16.05.2026.
6. Mealime. Meal Planning App for Healthy Eating [Електронний ресурс]. – Режим доступу: https://www.mealime.com/ – Дата звернення: 16.05.2026.

7. FatSecret. Reach Your Weight Loss Goals with fatsecret [Електронний ресурс]. – Режим доступу: https://www.fatsecret.com/ – Дата звернення: 16.05.2026.
8. Spoonacular. Spoonacular Food API [Електронний ресурс]. – Режим доступу: https://spoonacular.com/food-api – Дата звернення: 16.05.2026.
9. Object Management Group. Unified Modeling Language Specification, Version 2.5.1 [Електронний ресурс]. – Режим доступу: https://www.omg.org/spec/UML/2.5.1/ – Дата звернення: 16.05.2026.
10. Visual Paradigm. What is Use Case Diagram? [Електронний ресурс]. – Режим доступу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/ – Дата звернення: 16.05.2026.
11. Visual Paradigm. What is Component Diagram? [Електронний ресурс]. – Режим доступу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-component-diagram/ – Дата звернення: 16.05.2026.
12. Cloud Firestore Data Model [Електронний ресурс]. – Режим доступу: https://firebase.google.com/docs/firestore/data-model – Дата звернення: 16.05.2026.
13. Firebase Authentication for Flutter [Електронний ресурс]. – Режим доступу: https://firebase.google.com/docs/auth/flutter/start – Дата звернення: 16.05.2026.
14. Firebase Security Rules [Електронний ресурс]. – Режим доступу: https://firebase.google.com/docs/rules – Дата звернення: 16.05.2026.
15. Dart Documentation [Електронний ресурс]. – Режим доступу: https://dart.dev/docs – Дата звернення: 16.05.2026.
16. Dart. Introduction to Dart [Електронний ресурс]. – Режим доступу: https://dart.dev/language – Дата звернення: 16.05.2026.

17. Flutter Documentation [Електронний ресурс]. – Режим доступу: https://docs.flutter.dev/ – Дата звернення: 16.05.2026.
18. Firebase for Flutter [Електронний ресурс]. – Режим доступу: https://firebase.google.com/docs/flutter – Дата звернення: 16.05.2026.
19. Cloud Storage for Firebase on Flutter [Електронний ресурс]. – Режим доступу: https://firebase.google.com/docs/storage/flutter/start – Дата звернення: 16.05.2026.
20. ISO/IEC/IEEE 29119-1:2013. Software and systems engineering — Software testing — Part 1: Concepts and definitions [Електронний ресурс]. – Режим доступу: https://www.iso.org/standard/45142.html – Дата звернення: 16.05.2026.
21. Flutter. Testing Flutter apps [Електронний ресурс]. – Режим доступу: https://docs.flutter.dev/testing – Дата звернення: 16.05.2026.
22. Android Developers. SDK Platform release notes [Електронний ресурс]. – Режим доступу: https://developer.android.com/tools/releases/platforms – Дата звернення: 16.05.2026.
23. Flutter. Build and release an Android app [Електронний ресурс]. – Режим доступу: https://docs.flutter.dev/deployment/android – Дата звернення: 16.05.2026.

ДОДАТОК А

ЛІСТИНГ ОСНОВНИХ ПРОГРАМНИХ МОДУЛІВ

Ініціалізація Firebase

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

import 'auth/auth_gate.dart';
import 'firebase_options.dart';
import 'theme_controller.dart';

Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized();

  await Firebase.initializeApp(options: DefaultFirebaseOptions.currentPlatform);

  runApp(const MyApp());
}
```

Сервіс авторизації користувача

```
// Wraps FirebaseAuth + Firestore user-profile operations.
class AuthService {
  AuthService({FirebaseAuth? auth, FirebaseFirestore? firestore})
    : _auth = auth ?? FirebaseAuth.instance,
      _firestore = firestore ?? FirebaseFirestore.instance;

  final FirebaseAuth _auth;
  final FirebaseFirestore _firestore;

  /// Stream of auth state changes — used by [AuthGate].
  Stream<User?> authStateChanges() => _auth.authStateChanges();

  User? get currentUser => _auth.currentUser;

  /// Sign in with email & password.
  Future<UserCredential> signIn({
    required String email,
    required String password,
  }) {
    return _auth.signInWithEmailAndPassword(
      email: email.trim(),
      password: password,
    );
  }

  /// Create account, set displayName, and create Firestore profile doc.
  Future<UserCredential> signUp({
  required String email,
```

```
    required String password,
    required String firstName,
    required String lastName,
  }) async {
    final cred = await _auth.createUserWithEmailAndPassword(
      email: email.trim(),
      password: password,
    );

    final displayName = '${firstName.trim()} ${lastName.trim()}'.trim();
    await cred.user?.updateDisplayName(displayName);

    await UserService(
      auth: _auth,
      firestore: _firestore,
    ).createUserProfileAfterRegistration(displayName: displayName);

    return cred;
  }

  /// Send password-reset email.
  Future<void> sendPasswordReset(String email) {
    return _auth.sendPasswordResetEmail(email: email.trim());
  }

  /// Sign out the current user.
  Future<void> signOut() => _auth.signOut();
```

Сервіс роботи з Cloud Firestore

```
class UserService {
```

```
  UserService({FirebaseAuth? auth, FirebaseFirestore? firestore})
    : _auth = auth ?? FirebaseAuth.instance,
      _firestore = firestore ?? FirebaseFirestore.instance;

  final FirebaseAuth _auth;
  final FirebaseFirestore _firestore;
  final NutritionCalculator _calculator = const NutritionCalculator();

  User? get _currentUser => _auth.currentUser;

  DocumentReference<Map<String, dynamic>> _userRef(String uid) {
    return _firestore.collection('users').doc(uid);
  }

  Future<UserModel?> getCurrentUserProfile() async {
    final user = _currentUser;
    if (user == null) return null;

    final doc = await _userRef(user.uid).get();
    if (!doc.exists) return null;

    return UserModel.fromFirestore(doc);
  }

  Stream<UserModel?> watchCurrentUserProfile() {
    final user = _currentUser;
  if (user == null) return Stream<UserModel?>.value(null);

    return _userRef(user.uid).snapshots().map((doc) {
```

```
    if (!doc.exists) return null;
    return UserModel.fromFirestore(doc);
   });
  }

  Future<void> updateUserProfile(UserModel user) async {
   final currentUser = _currentUser;
   if (currentUser == null) {
    throw StateError('User must be authenticated to update a profile.');
   }

   final data = user.copyWith(uid: currentUser.uid).toFirestore()
    ..['email'] = currentUser.email ?? user.email
    ..['updatedAt'] = FieldValue.serverTimestamp();

   await _userRef(currentUser.uid).set(data, SetOptions(merge: true));
   await recalculateAndSaveNutritionTargets();
  }
}
```

Модуль NutritionCalculator

```
class NutritionCalculator {
  const NutritionCalculator();

  int? calculateProposedDailyCalories({
    required int? age,
    required String? gender,
    required double? heightCm,
    required double? weightKg,
    required String? activityLevel,
    required String? nutritionGoal,
  }) {
    if (age == null ||
        heightCm == null ||
        weightKg == null ||
        gender == null ||
        activityLevel == null ||
        nutritionGoal == null) {
      return null;
    }

    final bmr = switch (gender) {
      'male' => 10 * weightKg + 6.25 * heightCm - 5 * age + 5,
      'female' => 10 * weightKg + 6.25 * heightCm - 5 * age - 161,
      _ => null,
    };

    if (bmr == null) return null;
    final multiplier = switch (activityLevel) {
```

```
      'low' => 1.2,
      'medium' => 1.55,
      'high' => 1.725,
      _ => null,
    };

    if (multiplier == null) return null;

    final tdee = bmr * multiplier;

    final adjusted = switch (nutritionGoal) {
      'weight_loss' => tdee - 300,
      'weight_maintenance' => tdee,
      'weight_gain' => tdee + 300,
      'healthy_eating' => tdee,
      _ => null,
    };

    if (adjusted == null) return null;
    return max(1200, adjusted.round());
  }
}
```

Логіка підбору рецептів

```
List<QueryDocumentSnapshot<Map<String, dynamic>>> _filter(
  List<QueryDocumentSnapshot<Map<String, dynamic>>> docs,
  UserModel? profile,
) {
  final userAllergens =
      profile?.allergens.map((a) => a.toLowerCase()).toSet() ??
      const <String>{};

  final excludedProducts =
      profile?.excludedProducts.map((p) => p.toLowerCase()).toList() ??
      const <String>[];

  return docs.where((doc) {
    final data = doc.data();

    final name = (data['name'] as String? ?? '').toLowerCase();
    final matchesSearch =
        _searchQuery.isEmpty || name.contains(_searchQuery.toLowerCase());

    final mealType = List<String>.from(data['mealType'] as List? ?? [])
        .map((e) => e.toLowerCase())
        .toList();

    final matchesCategory =
        _selectedCategory == 'Всі' || mealType.contains(_selectedCategory);

    final calories = (data['calories'] as num?)?.toDouble();
final matchesCalories =
```

```
        !_showCalorieFilter || calories == null || calories <= _maxCalories;

    return matchesSearch &&
        matchesCategory &&
        matchesCalories &&
        !_hasBlockedAllergen(data, userAllergens) &&
        !_hasExcludedProduct(data, excludedProducts);
  }).toList();
}
```

Логіка формування плану харчування

```
class MealPlanService {
  MealPlanService({FirebaseAuth? auth, FirebaseFirestore? firestore})
      : _auth = auth ?? FirebaseAuth.instance,
        _firestore = firestore ?? FirebaseFirestore.instance;

  final FirebaseAuth _auth;
  final FirebaseFirestore _firestore;

  static const List<String> dayIds = [
    'monday',
    'tuesday',
    'wednesday',
    'thursday',
    'friday',
    'saturday',
    'sunday',
  ];

  static const List<String> mealTypes = [
    'breakfast',
    'lunch',
    'dinner',
    'snack',
  ];

  Future<void> initializeWeek(String weekId) async {
    final monday = mondayOfWeek(weekId);
final sunday = monday.add(const Duration(days: 6));
```

```
    final weekRef = _weekRef(weekId);
    final weekSnap = await weekRef.get();

    if (!weekSnap.exists) {
      await weekRef.set({
        'weekId': weekId,
        'startDate': Timestamp.fromDate(monday),
        'endDate': Timestamp.fromDate(sunday),
        'createdAt': FieldValue.serverTimestamp(),
        'updatedAt': FieldValue.serverTimestamp(),
      });
    }

    final batch = _firestore.batch();

    for (int i = 0; i < dayIds.length; i++) {
      final dayId = dayIds[i];
      final dayDate = monday.add(Duration(days: i));

      batch.set(_dayRef(weekId, dayId), {
        'dayId': dayId,
        'date': Timestamp.fromDate(dayDate),
      });

      for (final mealType in mealTypes) {
  batch.set(_mealRef(weekId, dayId, mealType), {
          'mealType': mealType,
        });
```

```
      }
    }

    await batch.commit();
  }

  Future<void> setMeal({
    required String weekId,
    required String dayId,
    required MealEntry meal,
  }) async {
    await _mealRef(weekId, dayId, meal.mealType).set(
      meal.toFirestore(),
      SetOptions(merge: true),
    );
  }
}
```

Логіка формування списку покупок

```
class ShoppingListService {
  ShoppingListService({FirebaseAuth? auth, FirebaseFirestore? firestore})
    : _auth = auth ?? FirebaseAuth.instance,
      _firestore = firestore ?? FirebaseFirestore.instance;

  final FirebaseAuth _auth;
  final FirebaseFirestore _firestore;

  CollectionReference<Map<String, dynamic>> _itemsRef([
    String listId = kDefaultShoppingListId,
  ]) {
    return _firestore
        .collection('users')
        .doc(_auth.currentUser!.uid)
        .collection('shopping_lists')
        .doc(listId)
        .collection('items');
  }

  Stream<List<ShoppingListItem>> watchItems([
    String listId = kDefaultShoppingListId,
  ]) {
    return _itemsRef(listId)
        .orderBy('addedAt', descending: false)
        .snapshots()
        .map((s) => s.docs.map(ShoppingListItem.fromDoc).toList());
  }
```

```
Future<void> addIngredients({
    required List<Map<String, dynamic>> ingredients,
    String? recipeName,
    String listId = kDefaultShoppingListId,
  }) async {
    for (final ing in ingredients) {
      final name = (ing['name'] as String?)?.trim() ?? '';
      if (name.isEmpty) continue;

      final qty = (ing['amount'] as num?)?.toDouble() ?? 1;
      final unit = (ing['unit'] as String?) ?? '';

      await addItem(
        name: name,
        quantity: qty,
        unit: unit,
        recipeName: recipeName,
        listId: listId,
      );
    }
  }
}
```

ДОДАТОК Б

РЕЗУЛЬТАТИ МОДУЛЬНОГО ТЕСТУВАННЯ

Програмний код модульних тестів класу NutritionCalculator

```
import 'package:flutter_test/flutter_test.dart';
import 'package:recipe_planner/services/nutrition_calculator.dart';
void main() {
  late NutritionCalculator calculator;
  setUp(() {
    calculator = const NutritionCalculator();
  });
  group('NutritionCalculator.calculateProposedDailyCalories', () {
    test('calculates calories for a male user with weight loss goal', () {
      final result = calculator.calculateProposedDailyCalories(
        age: 30,
        gender: 'male',
        heightCm: 180,
        weightKg: 85,
        activityLevel: 'medium',
        nutritionGoal: 'weight_loss',
      );
      expect(result, 2537);
    });
    test('calculates calories for a female user with maintenance goal', () {
      final result = calculator.calculateProposedDailyCalories(
        age: 25,
        gender: 'female',
        heightCm: 165,
        weightKg: 60,
        activityLevel: 'low',
        nutritionGoal: 'weight_maintenance', );
      expect(result, 1614);
```

```
  });
  test('applies minimum safety limit for daily calories', () {
    final result = calculator.calculateProposedDailyCalories(
      age: 80,
      gender: 'female',
      heightCm: 150,
      weightKg: 40,
      activityLevel: 'low',
      nutritionGoal: 'weight_loss',
    );
    expect(result, 1200);
  });
  test('returns null for negative age', () {
    final result = calculator.calculateProposedDailyCalories(
      age: -5,
      gender: 'male'
 heightCm: 180,
      weightKg: 80,
      activityLevel: 'medium',
      nutritionGoal: 'weight_maintenance',
    );
    expect(result, isNull);
  });
  test('returns null when required gender parameter is missing', () {
    final result = calculator.calculateProposedDailyCalories(
      age: 30,
gender: null,
      heightCm: 180,
      weightKg: 80,
```

```
      activityLevel: 'medium',
      nutritionGoal: 'weight_maintenance',
    );
    expect(result, isNull);
  });
  test('returns null for unknown gender value', () {
    final result = calculator.calculateProposedDailyCalories(
      age: 30,
      gender: 'unknown',
      heightCm: 180,
 weightKg: 80,
      activityLevel: 'medium',
      nutritionGoal: 'weight_maintenance',
    );
    expect(result, isNull);
  });
  test('calculates averaged value for other gender category', () {
    final result = calculator.calculateProposedDailyCalories(
      age: 30,
      gender: 'other',
      heightCm: 170,
      weightKg: 70,
      activityLevel: 'medium',
      nutritionGoal: 'weight_maintenance');

    expect(result, 2378);
  });
});
group('NutritionCalculator.calculateRecommendedMacros', () {
```

```
    test('calculates recommended macronutrients from daily calories', () {
      final result = calculator.calculateRecommendedMacros(
        dailyCalories: 2000,
        nutritionGoal: 'healthy_eating',
        weightKg: 70,
        activityLevel: 'medium',
      );
      expect(result, {'protein': 91, 'fat': 56, 'carbs': 283});
    });
  });
}
```

Результат виконання модульних тестів

00:00 +0: loading test/nutrition_calculator_test.dart

00:00 +1: calculates calories for a male user with weight loss goal

00:00 +2: calculates calories for a female user with maintenance goal

00:00 +3: applies minimum safety limit for daily calories

00:00 +4: returns null for negative age

00:00 +5: returns null when required gender parameter is missing

00:00 +6: returns null for unknown gender value

00:00 +7: calculates averaged value for other gender category

00:00 +8: calculates recommended macronutrients from daily calories

00:00 +8: All tests passed!

ДОДАТОК В

ОПУБЛІКОВАНІ ТЕЗИ НА
VII ВСЕУКРАЇНСЬКОЇ НАУКОВО-ПРАКТИЧНОЇ ІНТЕРНЕТ КОНФЕРЕНЦІЇ СТУДЕНТІВ ТА АСПІРАНТІВ «ТЕОРЕТИЧНІ ТА ПРИКЛАДНІ АСПЕКТИ РОЗРОБКИ КОМП'ЮТЕРНИХ СИСТЕМ 2026» (27.09.2026)

УДК 004.42

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ПЕРСОНАЛІЗОВАНОГО ПІДБОРУ КУЛІНАРНИХ РЕЦЕПТІВ І ПЛАНУВАННЯ ХАРЧУВАННЯ

Панасюк А. К., науковий керівник Ніколаєнко Д. В.

Мета дослідження полягає у проєктуванні програмного забезпечення для персоналізованого підбору кулінарних рецептів і планування харчування, яке враховує індивідуальні параметри користувача, його харчові цілі, дієтичні обмеження, алергени та добову норму калорій.

Об'єктом дослідження є процес персоналізованого планування харчування користувача, що охоплює вибір рецептів, формування меню, контроль калорійності раціону та підготовку списку необхідних продуктів.

Предметом дослідження є програмне забезпечення для автоматизації підбору кулінарних рецептів і планування харчування на основі персональних даних користувача, його харчових уподобань і встановлених обмежень.

Актуальність дослідження.

У сучасних умовах значна кількість користувачів прагне організувати харчування відповідно до власних потреб, однак самостійний пошук рецептів і складання збалансованого меню потребують часу та додаткових знань. Особа має враховувати калорійність страв, склад продуктів, наявність алергенів, харчову ціль, кількість прийомів їжі та доступність інгредієнтів. Через це виникає потреба у програмному засобі, який спрощує процес вибору рецептів і допомагає формувати персоналізований план харчування.

Розроблюване програмне забезпечення орієнтоване на користувачів, які самостійно планують раціон і готують їжу. Система передбачає можливість створення профілю користувача, зазначення персональних параметрів, вибору харчових обмежень і пошуку рецептів за заданими критеріями. На основі цих даних застосунок може пропонувати релевантні рецепти та допомагати формувати меню на день або тиждень.

Окрему роль у системі відіграє контроль калорійності. Після визначення добової норми калорій користувач може додавати рецепти до плану харчування, а система перевіряє сумарну калорійність запланованих страв. Це дає змогу уникнути перевищення добово ї норми та зробити планування харчування більш зручним і контрольованим.

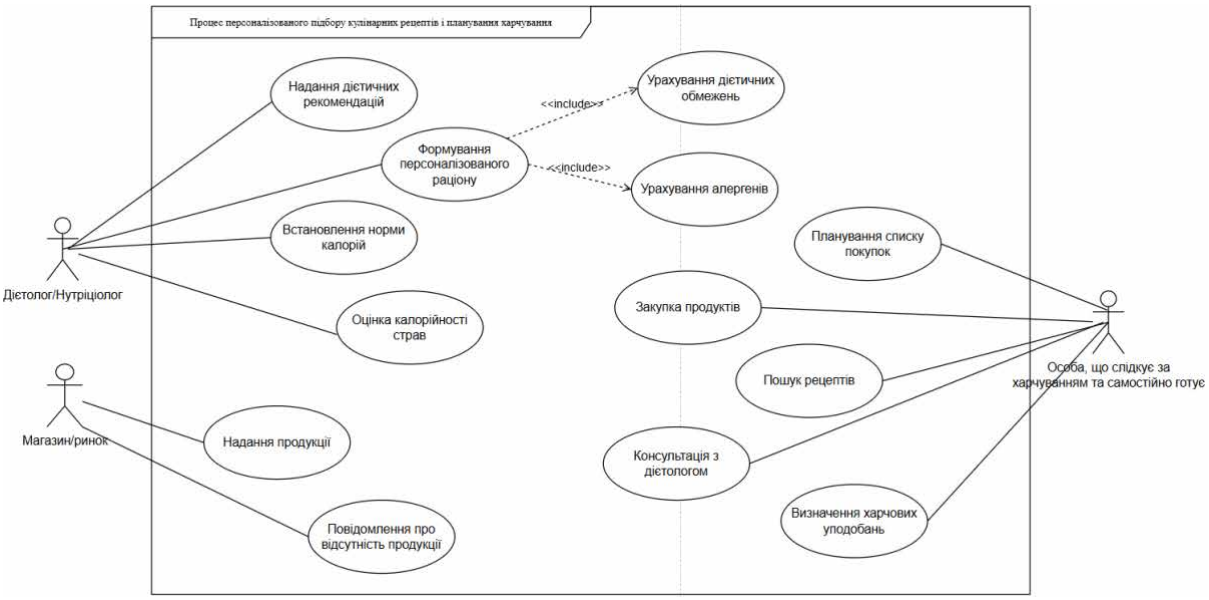

Рисунок 28 - Діаграма прецедентів системи персоналізованого підбору кулінарних рецептів і планування харчування

На рисунку 1 зображено діаграму прецедентів, яка відображає основних учасників процесу персоналізованого підбору рецептів і планування харчування з точки зору предметної області. Центральним актором є особа, що слідкує за харчуванням та самостійно готує: вона визначає харчові уподобання, здійснює пошук рецептів, консультується з дієтологом, планує список покупок і виконує закупівлю продуктів. Дієтолог або нутриціолог забезпечує формування персоналізованого раціону, встановлення норми калорій, оцінку калорійності страв і врахування дієтичних обмежень та алергенів. Магазин або ринок виступає зовнішнім джерелом продуктів і повідомляє про їх наявність або відсутність.

Технічна реалізація системи передбачає створення мобільного застосунку для операційної системи Android. Для розробки клієнтської частини обрано фреймворк Flutter і мову програмування Dart. Як серверну та хмарну частину системи доцільно використати платформу Firebase, зокрема сервіси автентифікації користувачів, збереження даних і синхронізації інформації між застосунком та базою даних. Для збереження інформації про користувачів, рецепти, плани харчування, калорійність і списки покупок використовується Cloud Firestore у складі Firebase.

Важливою складовою системи є організація даних, на основі яких виконується персоналізований підбір рецептів. У базі даних зберігаються відомості про рецепти, інгредієнти, калорійність страв, категорії харчування, алергени та обмеження користувача. Під час формування меню система порівнює параметри профілю користувача з характеристиками рецептів і відбирає ті страви, які відповідають заданим критеріям. Такий підхід дозволяє не лише автоматизувати пошук рецептів, а й підвищити якість рекомендацій, оскільки результат формується з урахуванням індивідуальних потреб користувача, його харчових цілей і доступності необхідних продуктів.

Отже, розроблення програмного забезпечення для персоналізованого підбору кулінарних рецептів і планування харчування є актуальним, оскільки спрямоване на автоматизацію рутинних дій, пов’язаних із вибором страв, складанням меню та контролем раціону. Практична цінність системи полягає у підвищенні зручності планування харчування та адаптації рекомендацій до індивідуальних потреб користувача.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Abeltino A., Grigorakis N., Tse G., et al. Digital applications for diet monitoring, planning, and precision nutrition: a critical review [Електронний ресурс] // *Nutrition Reviews*. – 2025. – Режим доступу: https://academic.oup.com/nutritionreviews/article/83/2/e574/7667651 (дата звернення: 26.04.2026).

2. Trumbo P., Schlicker S., Yates A. A., Poos M. Dietary reference intakes for energy, carbohydrate, fiber, fat, fatty acids, cholesterol, protein and amino acids [Електронний ресурс] // *Journal of the American Dietetic Association*. – 2002. – Vol. 102, № 11. – P. 1621–1630. – Режим доступу: https://pubmed.ncbi.nlm.nih.gov/12449285/ (дата звернення: 26.04.2026).

3. Firebase Documentation [Електронний ресурс]. – Режим доступу: https://firebase.google.com/docs (дата звернення: 26.04.2026).

4. Spoonacular Food API and Recipe API [Електронний ресурс]. – Режим доступу: https://spoonacular.com/food-api (дата звернення: 26.04.2026).

ДОДАТОК Г

ДЕКЛАРАЦІЯ ПРО АКАДЕМІЧНУ ДОБРОЧЕСНІСТЬ ТА ВИКОРИСТАННЯ ШІ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

Декларація про академічну доброчесність та використання ШІ

ДЕКЛАРАЦІЯ ЗДОБУВАЧА

Я, Панасюк Анна Костянтинівна , здобувачка НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

1. Моя бакалаврська кваліфікаційна робота (проєкт) на тему ««Програмне забезпечення для персоналізованого підбору кулінарних рецептів і планування харчування» є результатом моєї особистої праці.
2. Усі запозичення з текстів інших авторів мають відповідні посилання згідно з чинними стандартами.
3. Щодо використання інструментів ШІ зазначаю, що (обрати необхідне):
    - [✓] інструменти генеративного ШІ не використовувалися.
    - [✓] інструменти ШІ ChatGPT, Gemini, Claude, DeepL були використані для:
        - [✓] перекладу іншомовних джерел;
        - [✓] стилістичного редагування та перевірки граматики;
        - [✓] допомоги у написанні/відлагодженні програмного коду;
        - [ ] інше: _____.

Я розумію, що надання недостовірної інформації може призвести до скасування результатів захисту та відрахування з числа здобувачів НУБіП України.

«___» _____ 2026р. _____ **Анна ПАНАСЮК**

(підпис)