Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська дипломна робота

на тему: «Розробка програмного застосунку для визначення та аналізу раціону харчування людини»

Виконав: студент 4 курсу
групи 1ПІ-20б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Доценко В.С.
(прізвище та ініціали)
Керівник: к.т.н., доц. каф. ПЗ Хошаба О.М.
(прізвище та ініціали)
Рецензент: к.тн., доц. каф. ЗІ Лукічов В.В.
(прізвище та ініціали)

Допущено до захисту
Зав. кафедри _____
«10» червня 2024 р.

ВНТУ – 2024

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

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О.Н.
«12» березня 2024 р.

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

Доценко Вікторії Сергіївні

1. Тема роботи – «Розробка програмного застосунку для визначення та аналізу раціону харчування людини»

Керівник роботи: Хошаба Олександр Мирославович, к.т.н., доц. кафедри ПЗ, затверджені наказом вищого навчального закладу від 11 березня 2024 р. № 80.

2. Строк подання студентом роботи 3 червня 2024 р.

3. Вихідні дані до роботи: середовища розробки Visual Studio Code та IntelliJ IDEA, мови розробки JavaScript та Java, фреймворки React та Spring.

4. Зміст розрахунково-пояснювальної записки: вступ; аналіз стану питання розробки; порівняльний аналіз аналогів; аналіз методів розв'язання задачі; постановка задачі; аналіз даних, розробка моделі застосунку, розробка методу отримання інформації про нутріативні цінності продуктів харчування, розробка діаграм станів та загального алгоритму роботи застосунку; варіантний аналіз та обґрунтування вибору мови програмування для клієнтської частини застосунку, варіантний аналіз та обґрунтування вибору мови програмування для клієнтської частини застосунку, розробка структурних схем інтерфейсу; аналіз методів тестування, тестування розробленого програмного застосунку; висновки; список

використаних джерел; додатки.

5. Перелік графічного матеріалу: титульний слайд; актуальність теми; мета, об'єкт та предмет дослідження; задачі дослідження, наукова новизна, практична цінність одержаних результатів; порівняльний аналіз аналогів; використані технології, тестування, висновки, апробація результатів роботи і публікації.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Хошаба О.М., к.т.н., доцент кафедри ПЗ	13.03.24	03.06.24

7. Дата видачі завдання 13 березня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз стану розвитку програм для визначення та аналізу раціону харчування людини	14.03.24 – 24.03.24	виконано
2	Розробка структури та алгоритмів програмного застосунку	24.03.24 – 06.04.24	виконано
3	Варіантний аналіз та вибір засобів розробки	06.04.24 – 11.04.24	виконано
4	Розробка програмного застосунку	12.04.24 – 29.04.24	виконано
5	Тестування програмного застосунку	29.04.24 – 11.05.24	виконано
6	Оформлення матеріалів до захисту БДР	12.05.24– 30.05.24	виконано

Студент _____ (підпис) Доценко В.С. (прізвище та ініціали)

Керівник бакалаврської дипломної роботи _____ (підпис) Хошаба О.М. (прізвище та ініціали)

АНОТАЦІЯ

УДК 004.42

Доценко В.С. Розробка програмного застосунку для визначення та аналізу раціону харчування людини: бакалаврська дипломна робота зі спеціальності 121 Інженерія програмного забезпечення, освітня програма – Інженерія програмного забезпечення. Вінниця: ВНТУ, 2024. 63 с. На укр. мові. Бібліогр. : 20 назв ; рис. : 42; табл. 2.

У бакалаврській дипломній роботі проведено аналіз стану питання розробки застосунку для визначення та аналізу раціону харчування людини. Було проведено аналіз аналогів, визначено їх переваги і недоліки та доведено доцільність розробки власного додатка.

Виконане обґрунтування вибору мови та середовища програмування, а також технологій, які були використані під час розробки додатка.

Розроблено програмний продукт, який являє собою застосунок для визначення та аналізу раціону харчування людини.

Додаток реалізований за допомогою мов програмування Java та JavaScript. В якості середовищ для розробки програмного забезпечення було обрано Visual Studio Code та IntelliJ IDEA.

Отримані в бакалаврській дипломній роботі результати можна використати для отримання інформації про поживні цінності обраних продуктів, збору даних про фізичні параметри тіла людини, вести історію зміни ваги.

Ключові слова: веб-додаток, аналіз, харчування людини.

ANNOTATION

UDC 004.42

Dotsenko V.S. Development of a software application for determining and analyzing a person's diet: bachelor's thesis on the specialty 121 Software Engineering, educational program - Software Engineering. Vinnytsia: VNTU, 2024. 63p. In Ukrainian speech Bibliogr. : 20 titles; Fig. : 42; table 3.

In the bachelor's thesis, an analysis of the state of development of an application for determining and analyzing a person's diet was carried out. An analysis of analogues was carried out, their advantages and disadvantages were determined, and the expediency of developing one's own application was proven.

The justification of the choice of language and programming environment, as well as the technologies that were used during the development of the application, was carried out.

A software product has been developed, which is an application for determining and analyzing a person's diet.

The application is implemented using Java and JavaScript programming languages. Visual Studio Code and IntelliJ IDEA were chosen as software development environments.

The results obtained in the bachelor thesis can be used to obtain information about the nutritional value of selected products, collect data about the physical parameters of the human body, and keep a history of weight changes.

Keywords: web application, analysis, human nutrition.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Контроль раціону харчування є важливим аспектом ведення здорового способу життя. Збалансоване і правильно розроблене харчування допомагає забезпечити організм необхідними поживними речовинами, підтримувати оптимальну вагу та знижувати ризик виникнення ряду захворювань [1].

Основними принципами контролю раціону харчування є:

– вживання достатньої кількості фруктів, овочів, цільних зернових, білків, здорових жирів та обмеження споживання цукру, солі та насичених жирів.

– регулярний прийом їжі невеликими порціями, щоб уникнути перевантаження організму.

– приділення уваги відстеженню складу страв, калорійності та порцій.

– питний режим - вживання достатньої кількості рідини, переважно води.

Здоровий раціон харчування має низку важливих переваг для організму людини. Одна з основних переваг - підтримка оптимального здоров'я. Збалансоване харчування, багате на поживні речовини, вітаміни та мінерали, допомагає зміцнювати імунну систему, знижувати ризик серцево-судинних захворювань, діабету, деяких видів раку та інших хронічних патологій. Це дозволяє підтримувати організм у належному стані та попереджувати розвиток багатьох небезпечних недуг.

Не менш важливою перевагою є контроль ваги тіла. Дотримання принципів здорового харчування, в поєднанні з фізичною активністю, сприяє підтриманню здорової ваги або досягненню бажаного результату за умови надлишкової ваги.

Ще одна суттєва перевага - покращення когнітивних функцій. Повноцінне надходження мікро- та макронутрієнтів підвищує працездатність мозку, покращує пам'ять, увагу та здатність до навчання.

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

Метою роботи є покращення можливостей визначення раціону харчування людини шляхом розробки та використання спеціалізованого програмного застосунку який дозволяє автоматизувати процес аналізу раціону конкретної людини та сформувати рекомендації щодо раціону .

Відповідно до поставленої мети необхідно вирішити такі завдання:

- проаналізувати предметну область;
- провести порівняльний аналіз аналогів;
- провести аналіз методів розв’язання задачі;
- розробити модель застосунку;
- розробити метод пошуку продуктів харчування;
- розробити метод отримання інформації про поживні цінності продуктів харчування;
- розробити метод аналізу раціону харчування;
- розробити діаграми станів та загальний алгоритм роботи застосунку;
- провести варіантний аналіз засобів реалізації застосунку;
- розробити функціонал застосунку;
- розробити інтерфейс користувача;
- провести аналіз методів тестування;
- провести тестування розробленого застосунку.

Об’єктом дослідження є процеси розробки програмного застосунку для визначення та аналізу раціону харчування людини.

Предметом дослідження є методи і засоби реалізації програмного застосунку для визначення та аналізу раціону харчування людини.

Головною задачею є розробка програмних засобів системи аналізу та визначення раціону харчування людини.

Методи дослідження. У процесі досліджень використовувались такі методи:

- метод отримання інформації про нутріативні цінності продуктів харчування;
- методи тестування програмного застосунка;
- методи розробки бази даних та серверної частини додатка;

- методи розробки інтерфейсу користувача.

Новизна отриманих результатів.

1. Подальшого розвитку отримав метод пошуку продуктів харчування, який, на відміну від існуючих, пропонує користувачеві сортування продуктів від найбільш до найменш релевантних залежно від його потреб, що визначаються на основі останнього аналізу його раціону.

2. Подальшого розвитку набув метод отримання інформації про поживні цінності продуктів харчування, який включає збереження цих даних в локальну базу даних для пришвидшення їх повторного отримання, що відрізняє розроблений додаток від аналогів, у яких наявність такої функції перевірити неможливо через закритість коду, або ж така функція у них відсутня.

3. Подальшого розвитку отримав метод аналізу раціону харчування, який, на відміну від існуючих, використовує штучний інтелект, що дозволяє створювати персоналізовані рекомендації щодо внесення змін в раціон харчування залежно від даних користувача у випадку, якщо введений ним раціон не відповідає вимогам

Практична цінність отриманих результатів. Практична цінність одержаних результатів полягає у можливості використання розробленого додатка для аналізу раціону харчування, перегляду інформації про нутріативні цінності продуктів харчування, вести історію зміну параметрів тіла та формуванні здорового раціону харчування ,що відповідає потребам користувача.

Особистий внесок здобувача. Усі результати, викладені у бакалаврській дипломній роботі, отримані автором особисто.

Апробація результатів роботи. Описані у бакалаврській дипломній роботі положення доповідались на конференції «Стан, досягнення та перспективи інформаційних систем і технологій» та опубліковані в тезах доповіді.

Публікації. Результати роботи були опубліковані в матеріалах конференції - Стан, досягнення та перспективи інформаційних систем і технологій [2].

Аналіз. У пояснювальній записці до бакалаврської дипломної роботи було розглянуто 4 розділи та було використано 20 літературних джерел.

1 АНАЛІЗ СТАНУ РОЗВИТКУ ПРОГРАМ ДЛЯ ВИЗНАЧЕННЯ ТА АНАЛІЗУ РАЦІОНУ ХАРЧУВАННЯ ЛЮДИНИ

1.1 Аналіз предметної області

Додатки для визначення та аналізу раціону харчування людини – це важливий напрямок у сфері управління здоровим харчуванням. Вони надають користувачам широкий спектр можливостей для відстеження та оптимізації їхнього харчування.

Одна з ключових функцій таких додатків – відстеження споживання калорій, макро- та мікронутрієнтів. Користувачі можуть вводити інформацію про спожиті продукти, страви та напої, а додаток автоматично підраховує кількість спожитих калорій, білків, жирів, вуглеводів, вітамінів, мінералів тощо. Ця інформація накопичується протягом дня, тижня чи місяця, даючи змогу аналізувати харчові звички користувача.

Важливим аспектом є можливість порівняти фактичне споживання з рекомендованими нормами. Додатки надають інформацію про оптимальні норми споживання для різних вікових, гендерних та фізіологічних груп. Це дає користувачам змогу оцінити, наскільки їхній раціон харчування відповідає рекомендованим значенням.

Крім того, такі додатки пропонують функції планування та оптимізації раціону харчування. Вони можуть надавати рецепти страв із розрахунком їх поживної цінності, дозволяють складати меню на день, тиждень чи місяць з урахуванням цілей користувача та надавати рекомендації щодо покращення раціону.

Важливою перевагою є можливість підтримувати мотивацію та відстежувати прогрес користувача. Додатки можуть інтегруватися з фітнес-трекерами та іншими пристроями для моніторингу активності, дозволяють ставити цілі щодо споживання калорій, макро- та мікронутрієнтів, а також відображають динаміку змін у споживанні та вагових показниках.

Популярність таких додатків серед людей, які прагнуть вести здоровіший спосіб життя, контролювати вагу або оптимізувати своє харчування для досягнення

певних цілей, зумовлена їхньою здатністю підвищувати інформованість користувачів про склад їхнього раціону та допомагати у прийнятті обґрунтованих рішень для його покращення.

Таким чином, є актуальною розробка програмних засобів системи аналізу та визначення раціону харчування людини.

1.2 Порівняльний аналіз аналогів

Розглянемо існуючі застосунки для визначення та аналізу раціону харчування людини.

MyFitnessPal [2] - це популярний додаток для відстеження калорій та ведення здорового способу життя. Він був запущений ще у 2005 році і із роками став одним з найвідоміших додатків такого типу. Ключова перевага MyFitnessPal - це його всеосяжна база даних продуктів, яка дозволяє детально аналізувати споживані калорії, поживні речовини та інші показники. Користувачі можуть вводити інформацію про свої прийоми їжі та отримувати повну статистику.

Не менш важливою функцією додатку є можливість інтеграції з фітнес-трекерами. Це дозволяє відслідковувати витрати енергії під час тренувань та щоденної активності. Додаток також пропонує персоналізовані цілі та рекомендації, щоб допомогти користувачам досягти бажаних результатів зі схудненням або набору м'язової маси. Крім того, MyFitnessPal має соціальні функції, що дозволяють обмінюватися успіхами та мотивувати один одного.

Додаток MyFitnessPal продемонстровано на рисунку 1.1.

Lose It! [3] - популярний додаток для контролю калорій та ведення здорового способу життя. Він був запущений у 2008 році і з тих пір залишається одним з лідерів у своєму сегменті. Подібно до MyFitnessPal, основна функція Lose It! - це допомога користувачам вести облік споживаних калорій та харчування.

Додаток має велику базу даних продуктів, завдяки якій користувачі можуть легко вводити та аналізувати інформацію про свій раціон. Lose It! також дозволяє синхронізуватися з фітнес-трекерами, що надає повну картину про спожиті та витрачені калорії. Крім того, додаток пропонує персоналізовані цілі та планування

прийомів їжі для досягнення цілей схуднення або підтримки ваги. Соціальні функції Lose It! дають змогу користувачам ділитися успіхами та підтримувати один одного.

Додаток Lose It! продемонстровано на рисунку 1.2.

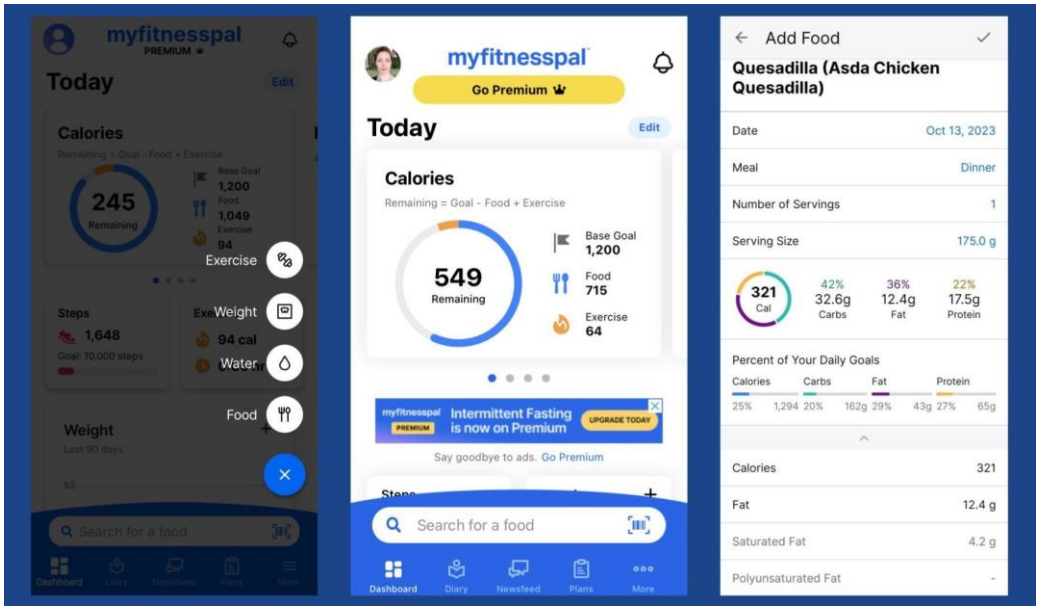


Рисунок 1.1 – MyFintessPal

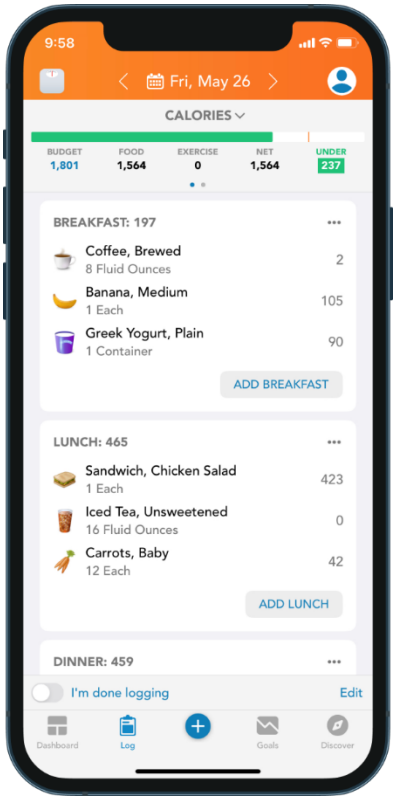


Рисунок 1.2 – Додаток Lose It!

Fooducate [4] - це унікальний додаток, що фокусується не просто на підрахунку калорій, а на аналізі харчової цінності продуктів. Основна мета Fooducate - допомогти користувачам робити більш свідомий вибір під час походів до продуктового магазину та приготування їжі вдома.

Ключова особливість додатку - це його детальна база даних продуктів харчування. Сканувавши штрих-код будь-якого продукту, Fooducate надає розширену інформацію про його склад, поживну цінність та ступінь обробки. Додаток оцінює продукти за шкалою від A до D і надає корисні поради щодо більш здорових альтернатив. Крім того, Fooducate пропонує рецепти, меню та інші інструменти для планування збалансованого харчування. Соціальні функції додатку дозволяють користувачам ділитися своїми відкриттями та досвідом.

Додаток Fooducate продемонстровано на рисунку 1.3.

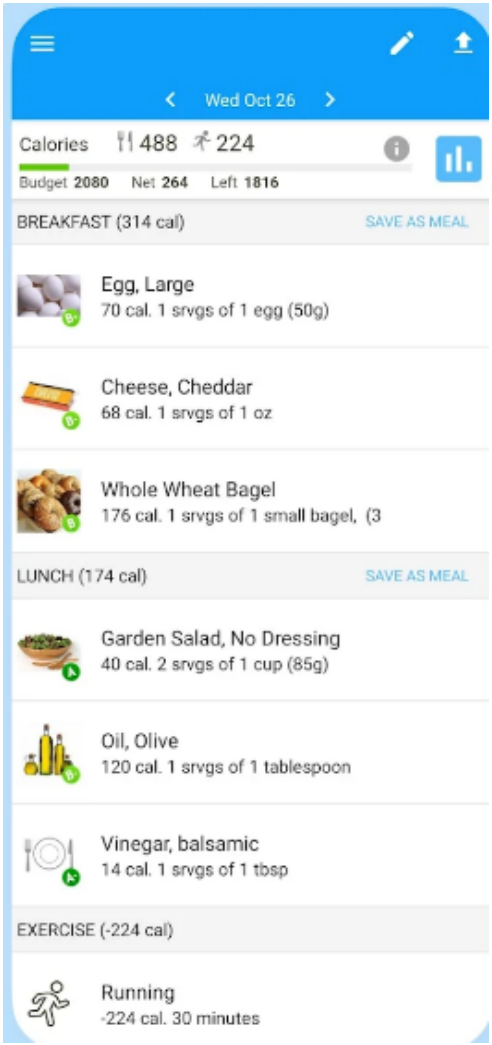


Рисунок 1.3 – Fooducate

Для демонстрації відмінностей між аналогами та власною розробкою, було сформовано порівняльну таблицю характеристик (таблиця 1.1).

Таблиця 1.1 – Порівняння характеристик застосунків

Характеристика	MyFitnessPal	Lose it!	Fooducate	Власна розробка
Аналіз нутрієнтного складу продуктів	1	1	1	1
Розрахунок параметрів тіла користувача	1	1	0	1
Ведення історії раціону користувача	1	1	1	1
Ведення історії змін ваги користувача	1	0	1	1
Персональні рекомендації	0	0	0	1
Додавання своїх продуктів користувачем	0	0	1	1
Сумарний бал	4	3	4	6

Таким чином, власна розробка за функціональними можливостями перевершує Fooducate, MyFitnessPal на 33.4% (100% - 4/6*100% = 33.4%), Lose it! на 50% (100% - 3/6*100% = 50%).

1.3 Аналіз методів розв’язання задачі

Для розв’язання задачі розробки програмного застосунку для визначення та аналізу раціону харчування людини необхідно розробити серверну та клієнтську частину додатка.

Для розробки серверної частини програмного застосунку для визначення та аналізу раціону харчування людини можна використати ряд способів та інструментів. Насамперед, необхідно обрати мову програмування, яка підходить

для побудови RESTful API, обробки HTTP-запитів та взаємодії з базою даних. Популярними варіантами є Django, Node.js, Spring.

Node.js, Django та Spring є трьома популярними технологіями, які можна використовувати для розробки серверної частини застосунку для визначення та аналізу раціону харчування людини. Кожна з них має свої особливості та переваги, тому вибір залежить від специфіки проекту, досвіду команди розробників, вимог до продуктивності, масштабованості та інтеграції з іншими системами.

Node.js [5] - це середовище виконання JavaScript, що дозволяє будувати швидкі та масштабовані мережеві додатки. Асинхронна природа Node.js робить його ефективним для обробки великої кількості паралельних запитів, що важливо для додатку з багатьма користувачами. Крім того, наявність великої екосистеми модулів та фреймворків, таких як Express.js, спрощує та прискорює розробку веб-орієнтованих серверних застосунків. Node.js також підходить для розробки мікросервісної архітектури, що може бути корисно для масштабування додатку.

Django [6] - це Python-фреймворк, який пропонує потужні інструменти для швидкого та ефективного створення веб-застосунків. Він використовує архітектуру Модель-Представлення-Шаблон (MVT), що дозволяє чітко розділити логіку програми на окремі компоненти.

Модель Django відповідає за взаємодію з базою даних, використовуючи вбудовану об'єктно-реляційну модель (ORM). Це дає можливість працювати з базою даних через Python-код, не вдаючись до безпосереднього написання SQL-запитів. Така абстракція спрощує розробку та підвищує переносимість коду.

Spring [7] - це один з найбільш популярних Java-фреймворків для розробки Enterprise-рівневих веб-додатків. Spring Boot пропонує комплексне рішення для швидкої розробки веб-застосунків з вбудованою підтримкою безпеки, моніторингу та розгортання. Java, як статично типізована мова, забезпечує кращу масштабованість, продуктивність та придатність для великих, довгострокових проектів.

Насамперед, Spring відрізняється своєю потужною екосистемою, яка включає широкий спектр модулів та інструментів для вирішення практично будь-яких

завдань, що виникають у процесі розробки складних веб-додатків. Це включає в себе модулі для роботи з базами даних, веб-сервісами, безпекою, моніторингом та іншими важливими аспектами. Така всеосяжність і комплексність Spring дозволяє розробникам швидко та ефективно вирішувати завдання, не витрачаючи додаткові зусилля на інтеграцію різноманітних бібліотек та інструментів.

Зважаючи на ці фактори, Spring є найбільш доцільним вибором для розробки серверної частини додатку для визначення та аналізу раціону харчування. Його масштабованість, комплексність та надійність роблять його ідеальним рішенням для реалізації складних аналітичних завдань, забезпечення безпеки та ефективної роботи застосунку в довгостроковій перспективі.

Окрім цього, важливо відмітити те, що використання Spring полегшує процес розробки необхідних програмних компонентів, дозволяє автоматизувати налаштування. Spring має також багато інших функцій, що покращують процес розробки та забезпечують створення якісних програмних застосунків.

Наступним кроком є створення RESTful API за допомогою відповідних фреймворків. Це передбачає визначення ендпоінтів для виконання основних операцій CRUD (створення, читання, оновлення, видалення) над даними користувачів та продуктів харчування. Важливим аспектом є інтеграція з базою даних, де слід обрати найбільш підходящу систему управління базами даних, як реляційні (MySQL, PostgreSQL) так і NoSQL (MongoDB, Cassandra) рішення.

Розглянемо такі системи управління базами даних, як MySQL, PostgreSQL та MongoDB.

PostgreSQL [8] - потужна об'єктно-реляційна база даних з широким спектром можливостей. Вона є надійною, масштабованою та має багату екосистему розширень. PostgreSQL добре підходить для додатків, які вимагають складної обробки транзакцій, підтримки ACID-властивостей, роботи з JSON-даними та аналітичних запитів. Завдяки вбудованим типам даних, PostgreSQL спрощує роботу з різноманітною інформацією, зокрема, пов'язаною зі здоров'ям та харчуванням.

Приклад роботи з PostgreSQL наведено на рисунку 1.4.

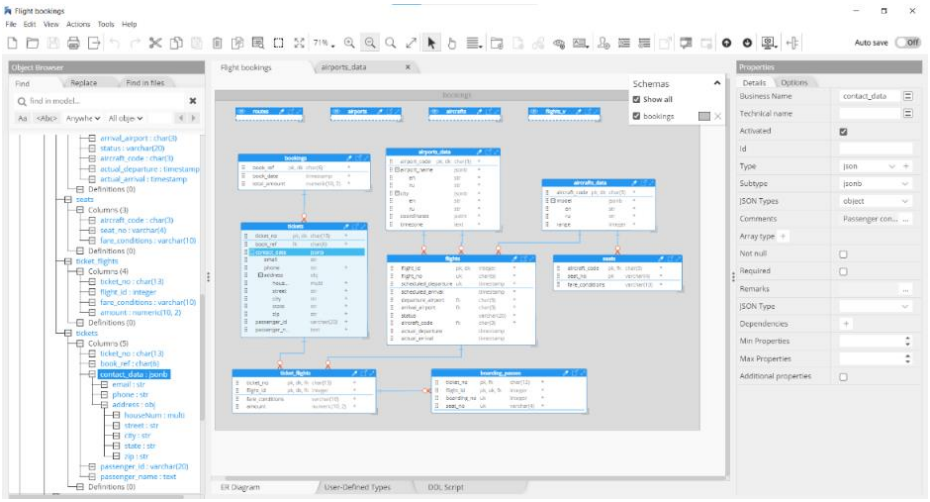


Рисунок 1.4 – Приклад використання PostgreSQL

MySQL [9] - одна з найпопулярніших реляційних баз даних з відкритим кодом. Вона відома своєю швидкістю, простотою використання та широким спектром інструментів. MySQL добре підходить для веб-додатків, електронної комерції та інших проектів, що вимагають високої продуктивності для типових операцій CRUD. Однак, MySQL має дещо обмежені можливості в плані підтримки складних запитів та транзакцій.

Приклад роботи з MySQL наведено на рисунку 1.5.

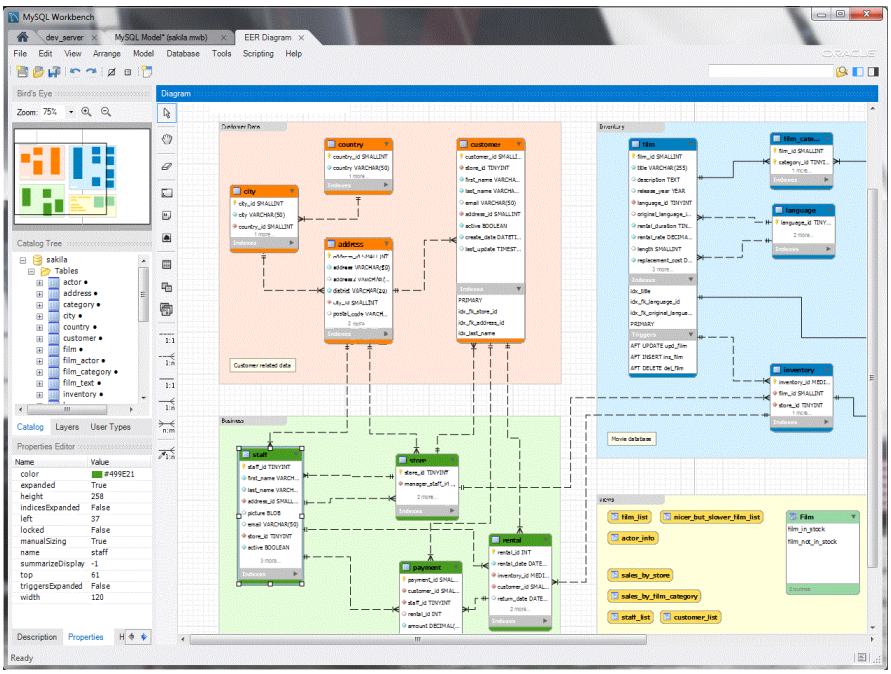


Рисунок 1.5 – MySQL

MongoDB [10] - одна з провідних NoSQL баз даних, орієнтована на документи. Вона відрізняється гнучкістю, масштабованістю та високою швидкістю. MongoDB добре підходить для додатків, які працюють з неструктурованими даними, такими як веб-контент, логи, дані користувачів. Ця база даних може бути корисною для зберігання та швидкого пошуку інформації про продукти харчування, їх склад та властивості. Однак, MongoDB не має такої потужної підтримки транзакцій, як PostgreSQL чи MySQL.

Приклад роботи з MongoDB наведено на рисунку 1.6.

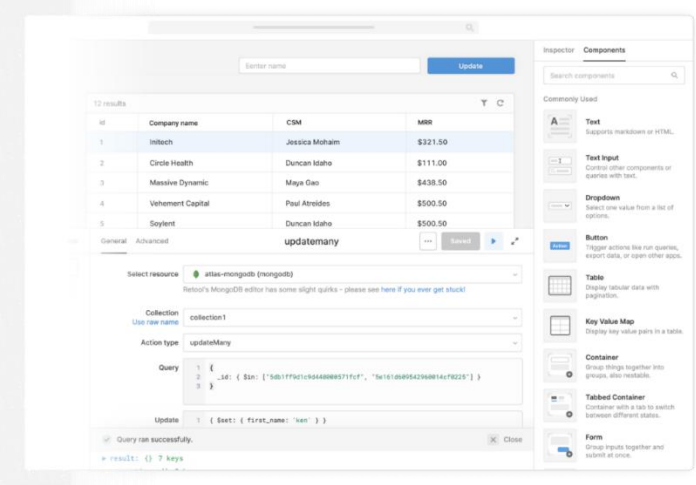


Рисунок 1.6 – MongoDB

Для розробки серверної частини додатку для визначення та аналізу раціону харчування людини, PostgreSQL може бути найбільш доречним вибором. Його можливості для обробки складних запитів, роботи з JSON-даними, забезпечення цілісності даних та аналітики роблять його привабливим рішенням для такого типу додатків.

Окрім вищезазначеного, PostgreSQL відмінну масштабованість та надійність, що важливо для додатку, який може мати багато активних користувачів. Багата екосистема інструментів та розширень, що спрощує розробку, моніторинг та обслуговування.

Реалізація логіки додатку включає обробку запитів користувачів на сервері, проведення розрахунків показників здоров'я, норми калорій та аналіз раціону

харчування. Окрім цього, необхідно забезпечити зберігання історії змін ваги та раціону користувача. Для оптимізації взаємодії з базою даних, доцільно використовувати механізми кешування.

Забезпечення безпеки є важливим аспектом, який включає реалізацію аутентифікації та авторизації користувачів, захист API та шифрування даних, що передаються між клієнтом і сервером.

1.4 Постановка задач дослідження

Провівши аналіз розв’язання поставленої задачі, аналіз стану застосунків для визначення та аналізу раціону харчування людини та провівши порівняння аналогів, було визначено такий функціонал для власної розробки:

- виведення інформації про поживні цінності обраних користувачем продуктів;
- збереження цих даних в локальну базу даних для пришвидшення їх повторного отримання;
- збір даних про фізичні параметри користувача та розрахунок певних показників, таких як індекс маси тіла, норма калорій і т.п.;
- ведення історії зміни ваги користувача;
- розрахунок та ведення історії споживання користувачем різних речовин, калорій, білків і т.п.

Технічне завдання на розробку наведено в додатку А.

1.5 Висновки

У цьому розділі було проведено аналіз предметної області застосунків для визначення та аналізу раціону харчування людини, доведено актуальність такої розробки. Було проведено порівняльний аналіз аналогів, складено таблицю порівняння характеристик, в результаті чого було доведено, що власна розробка має кращі характеристики порівняно з аналогами. Було проведено аналіз методів розв’язання задачі. В результаті, проведено постановку задач дослідження та сформовано функціонал власного додатка.

2 РОЗРОБКА СТРУКТУРИ ТА АЛГОРИТМІВ ПРОГРАМНОГО ДОДАТКУ

2.1 Аналіз даних

Визначення та аналіз раціону харчування людини залежить від конкретної людини, її характеристик, та того, що для виконання цієї задачі необхідно введення даних до програми самих користувачем, після чого відбувається їх аналіз та видача результату. Загалом, у великій мірі сама суть такої програми покладає в аналізі та обробці даних.

У вікні реєстрації користувача вводяться такі дані, як: ім’я, пошта, пароль, дата народження, стать. Ці дані є обов’язковими для встановлення аутентичності створюваного аккаунту та для того, щоб система точніше обробляла отримані дані та розробляла кращий для певного користувача раціон харчування і рекомендації. Крім цього, користувач може ввести також необов’язкові дані, які хоч і також дадуть системі краще розуміння про фізичний стан користувача: вага, зріст, бажана вага, інші антропометричні параметри.

У вкладці «Харчування» користувач може ввести назву спожитих продуктів харчування та їхню кількість. На основі цього, додаток розраховує характеристики цих продуктів – калорійність, енергетичну цінність, жири, білки, вуглеводи, інші показники. Після додання певного переліку продуктів, схожим чином розраховується загальні показники по раціону харчування.

Окрім цього, ці дані можна порівнювати із бажаними показниками, які раніше ввів користувач, для контролю раціону харчування.

2.2 Розробка моделі застосунку

Для кращого розуміння архітектури розроблюваного застосунка було побудовано модель системи.

Ця модель описує структуру мікросервісів застосунку. Мікросервіси обмінюються між собою даними з використанням Kafka.

Kafka [11] ключовим компонентом в архітектурі мікросервісів, забезпечуючи

надійний і масштабований механізм обміну повідомленнями між різними сервісами. Одна з основних переваг Kafka - це її здатність розв'язувати зв'язки між мікросервісами. Замість того, щоб мікросервіси напряму викликали один одного (що може призвести до тісної зв'язаності), вони публікують дані в Kafka-теми, а інші сервіси можуть підписуватися на ці теми та обробляти дані незалежно.

Завдяки цим можливостям, Kafka стає невід'ємною частиною сучасних мікросервісних архітектур. Вона дозволяє створювати гнучкі, масштабовані та стійкі до відмов системи, розділяючи дані та логіку обробки між окремими сервісами. Це робить архітектуру мікросервісів більш ефективною, керованою та придатною для розгортання в хмарних середовищах.

Спершу, при завантаженні застосунка, запускається мікросервіс User Interface, який являє собою інтерфейс користувача, з яким взаємодіє користувач з використанням інтернету.

Залежно від дій користувача, User Interface передає запит з використанням REST API мікросервісу Controller, який є керуючим в системі та обробляє ці запити. Він перевіряє введені користувачем логін та пароль та перевіряє їх. В разі правильного введення даних, він запускає мікросервіс User, що відповідає за користувача, і користувач взаємодіє уже з цим мікросервісом. User має свою окрему базу даних User DB.

Зображення моделі застосунку наведено на рисунку 2.1.

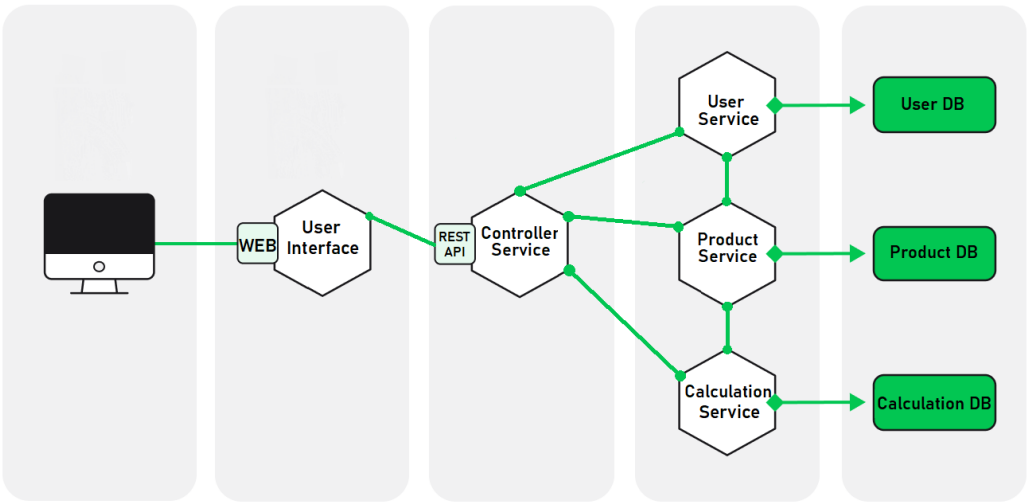


Рисунок 2.1 – Модель застосунку

Мікросервіс Product відповідає за надання інформації про продукти, раціон харчування, його нутріативної цінності. Після успішної авторизації, мікросервіс Controller отримує від User токен аккаунту, та з ним звертається уже до мікросервісу Product для виконання необхідних користувачу операцій.

Мікросервіс Calculation використовується для розрахунку нутріативної цінності раціону харчування, складання раціону харчування.

2.3 Розробка методу пошуку продуктів харчування

Метод пошуку продуктів харчування призначений для оптимізації раціону харчування користувача шляхом сортування продуктів за вмістом основних поживних речовин: білків, жирів, вуглеводів і цукрів. Система аналізує дані про поточний раціон користувача і виявляє відхилення вмісту певних речовин від рекомендованих норм. Якщо відхилення перевищують 5%, система пропонує відповідні сортування продуктів для корекції раціону, відображаючи рекомендації у вигляді позначок.

Цей метод включає складається з таких кроків:

1.Користувач відкриває меню пошуку продуктів харчування для додавання до раціону.

2.Користувач відкриває список критеріїв сортування.

3.Система відображає йому можливі сортування продуктів за вмістом поживних речовин у грамах. А саме білків, жирів, вуглеводів та цукрів.

4.Система отримує із бази даних результати останнього аналізу раціону харчування користувача.

5. Якщо система виявляє відхилення вмісту деяких речовин від норми більше, ніж на 5%, вона формує рекомендацію про необхідність застосування сортування

6. Дані щодо рекомендацій відображаються у вигляді відповідних позначок поруч з назвою виду сортування

7.Користувач обирає один із запропонованих сортувань.

Блок-схема алгоритму роботи цього методу наведена на рисунку 2.2.

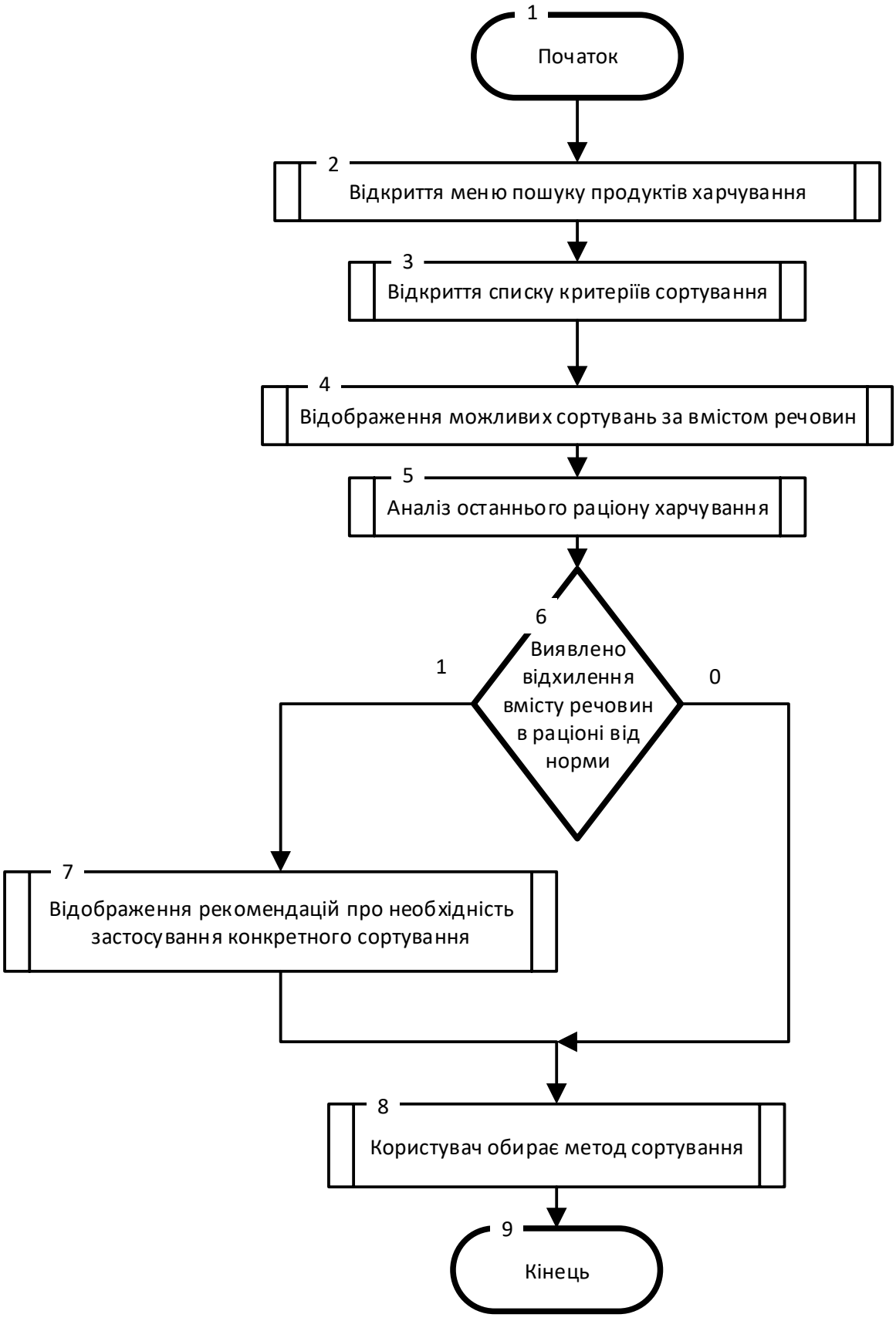


Рисунок 2.2 – Блок-схема алгоритму роботи методу пошуку продуктів

Застосування рекомендованих фільтрів дозволяє користувачу в першу чергу

додати до раціону харчування ті продукти, які сприятимуть поверненню показників до норми.

2.4 Розробка методу отримання інформації про нутріативні цінності продуктів харчування

Метод отримання інформації про нутріативні цінності продуктів харчування необхідний для того, щоб допомогти користувачу краще контролювати свій раціон харчування, бути більш обізнаним в цінності та можливій шкоді від продуктів харчування, які він споживає, та для аналізу додатком раціону користувача. Також, цей метод дає поради користувачу щодо загальновизнаної денної норми споживання обраного продукту та аналізує, скільки саме йому потрібно споживати від денної норми цього продукту залежно від його віку, зросту та його потребах – схуднути, набрати вагу чи набрати м’язову масу.

Метод отримання інформації про нутріативні цінності продуктів харчування включає функціонал:

1. Користувач входить в додаток та обирає меню «Пошук».
2. Користувач вводить назву необхідного йому продукту.
3. Система розпочинає пошук необхідної інформації в базі даних серед продуктів створених користувачем.
4. Якщо інформація не була знайдена, система шукає дані в базі даних серед продуктів усіх продуктів.
5. Якщо інформація не була знайдена, тоді надсилається запит до віддаленого серверу з ресурсами. Після цього, отримуються необхідні дані, які зберігаються в базу даних системи.
6. Інформація підраховується на основі маси, введеної користувачем та виводиться на екран.
7. Якщо користувач вибирає дату на прийом їжі та натискає «Додати», продукт додається в раціон.

Збереження інформації про властивості продуктів харчування в локальну базу даних є характерною особливістю цього методу порівняно з розробками у аналогів,

наявність у яких такої функції перевірити неможливо через закритість коду додатків, або ж, така функція у них відсутня.

Блок-схема алгоритму роботи методу наведена на рисунку 2.3.

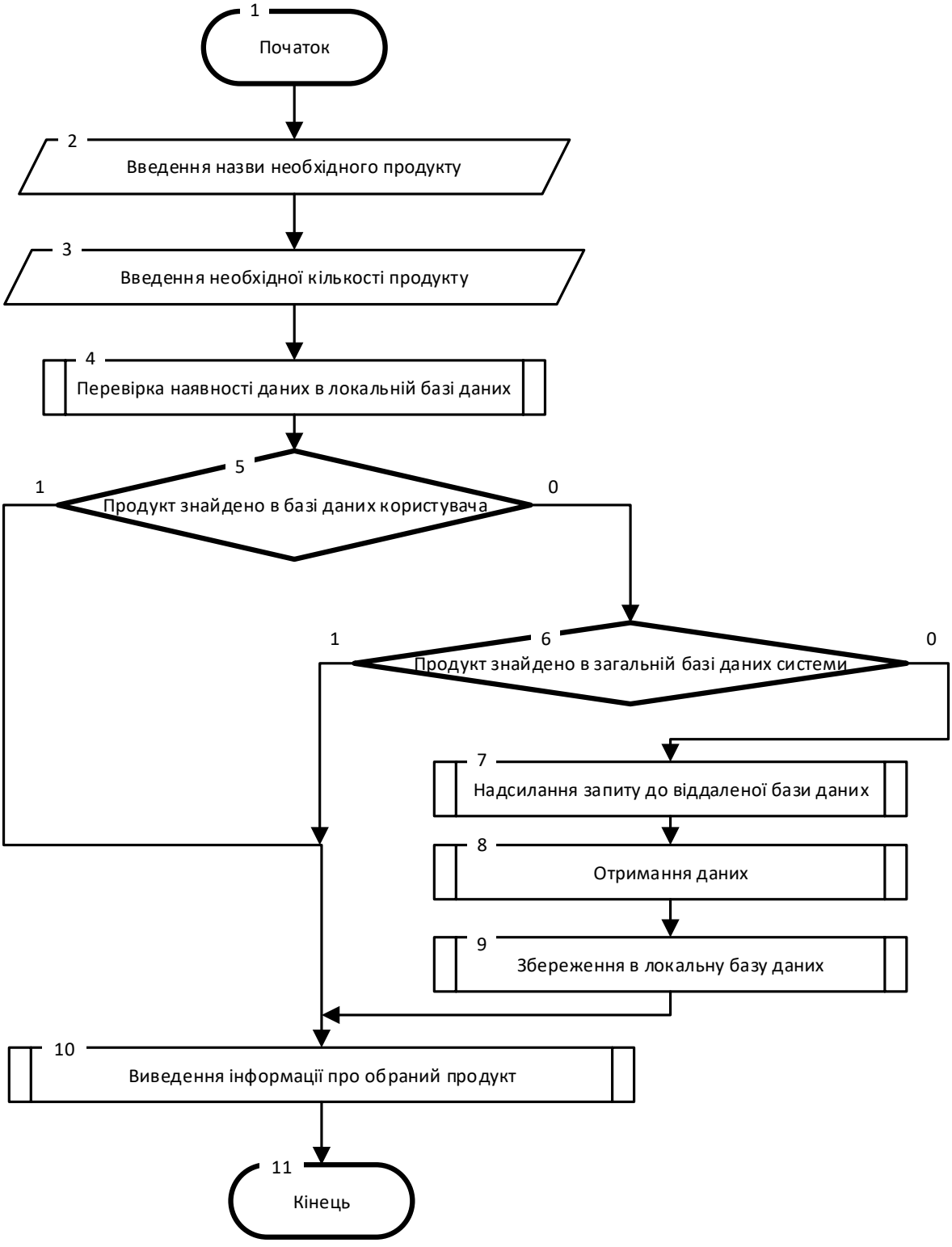


Рисунок 2.3 – Блок-схема методу отримання інформації про нутріативні цінності продуктів харчування

Збереження даних в локальну базу даних є важливим через те, що це пришвидшує пошук необхідних даних у разі їх присутності в локальній базі даних та позбавляє необхідності кожен раз звертатися до віддаленої бази даних для отримання необхідних даних, що може зайнятий більше часу, ніж відправка запиту до бази даних та отримання результату від неї. Окрім того, збереженння даних в локальній базі даних дозволяє отримати необхідну інформацію без інтернет-з'єдання, що може бути корисним для користувачів в ситуаціях, коли у них відсутній доступ до Інтернет або ж рівень зв'язку є незадовільним для швидкого отримання інформації.

2.5 Розробка методу аналізу раціону харчування

Метод аналізу раціону харчування призначений для персоналізованої рекомендації продуктів харчування на основі аналізу поточного раціону користувача. Користувач переходить у свій профіль та запускає аналіз раціону, після чого система надсилає запит чат-боту зі штучним інтелектом, включаючи дані з профілю користувача. Чат-бот аналізує отриману інформацію і пропонує три продукти, які варто додати до раціону, щоб поліпшити його баланс і відповідати індивідуальним потребам користувача.

Цей метод складається з таких кроків.

1. Користувач переходить у вкладку профілю.

2. Натиснуто на кнопку "Аналіз".

3. Система проводить аналіз останнього раціону харчування та виводить результат користувачеві.

4. Надсилається запит чат-боту зі штучним інтелектом з питанням "Які 3 продукти раціону харчування варто додати в мій раціон". Для більш точної відповіді, разом із цим питанням надсилається також інформація з профілю користувача.

5. Чат-бот дає відповідь, яка відображається користувачеві в його профілі.

Блок-схема алгоритму цього методу наведено на рисунку 2.4.

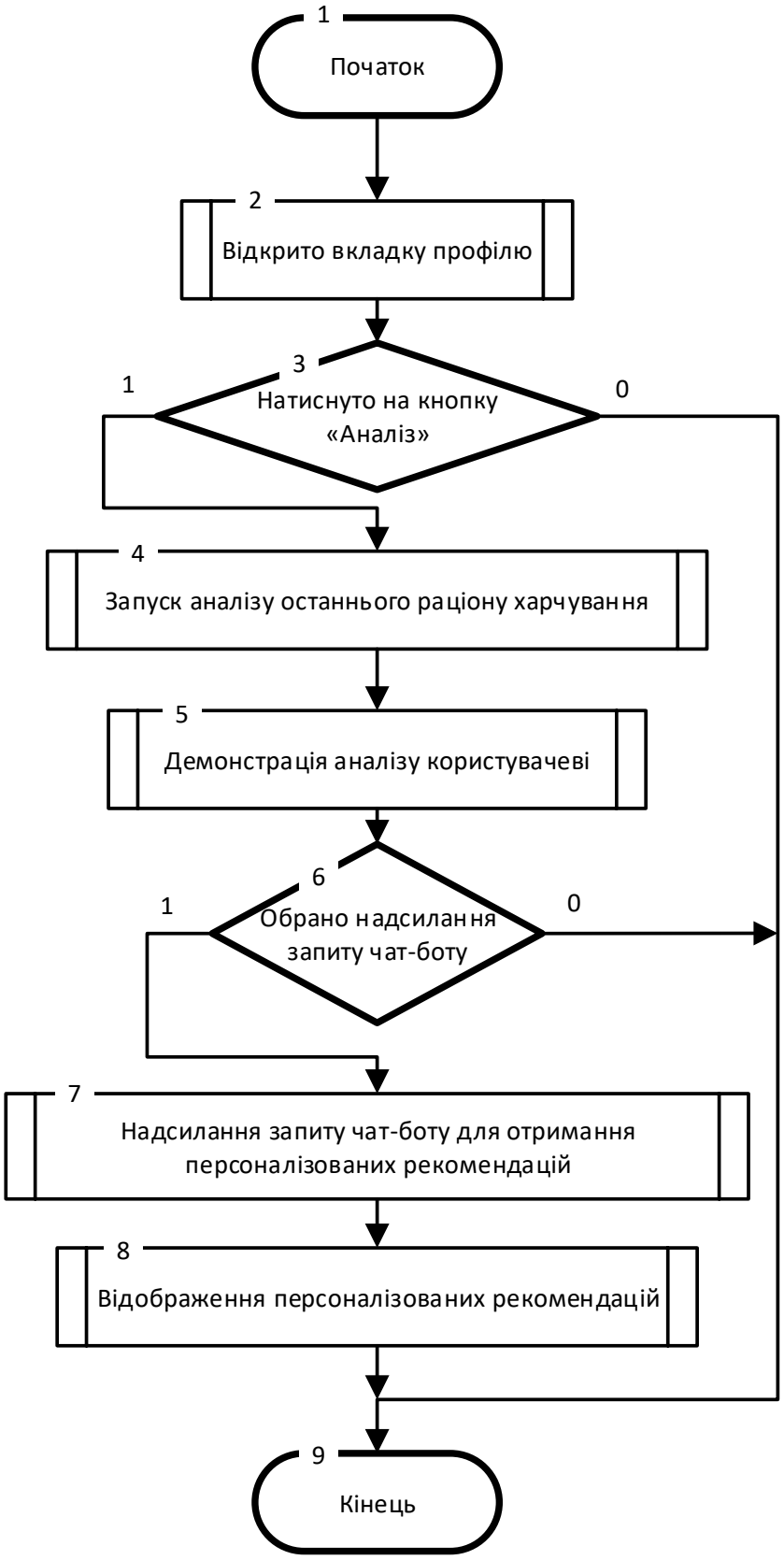


Рисунок 2.4 – Блок-схема алгоритму роботи методу аналізу раціону харчування

Цей підхід забезпечує точні, персоналізовані рекомендації, що сприяють здоровому харчуванню.

2.6 Розробка діаграм станів та алгоритмів роботи застосунку

Розробка алгоритмів роботи застосунку є необхідною складовою для розробки програмного забезпечення. Вони слугують інструкцією щодо розробки та демонструють очікувану логіку роботи застосунка, відхилення від якої вважатиметься помилковою роботою застосунка.

Було побудовано діаграми станів для розроблюваного додатка. Діаграма станів під час реєстрації користувача продемонстрована на рисунку 2.5.

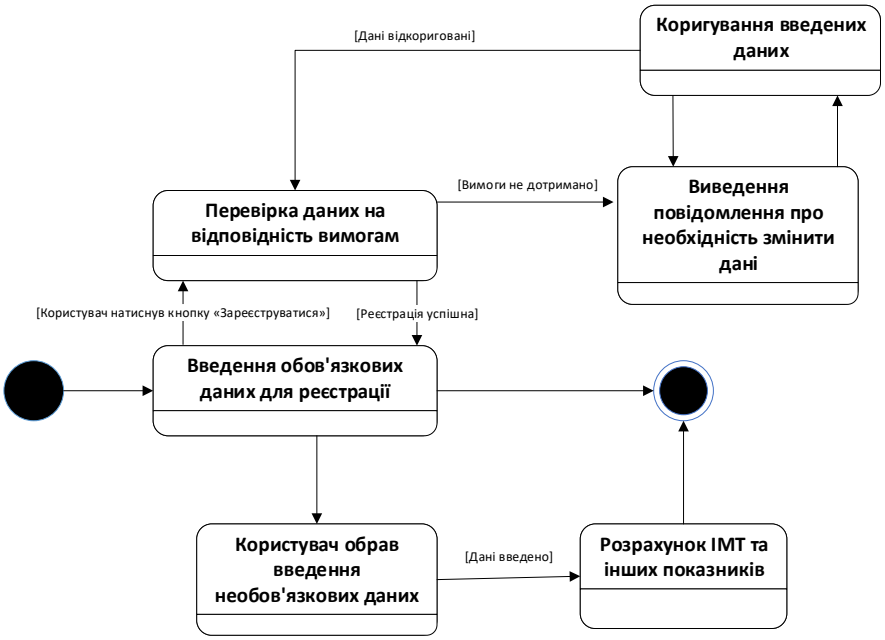


Рисунок 2.5 – Діаграма станів під час реєстрації користувача

Діаграма станів при роботі у вкладці «Мої результати» продемонстрована на рисунку 2.6.

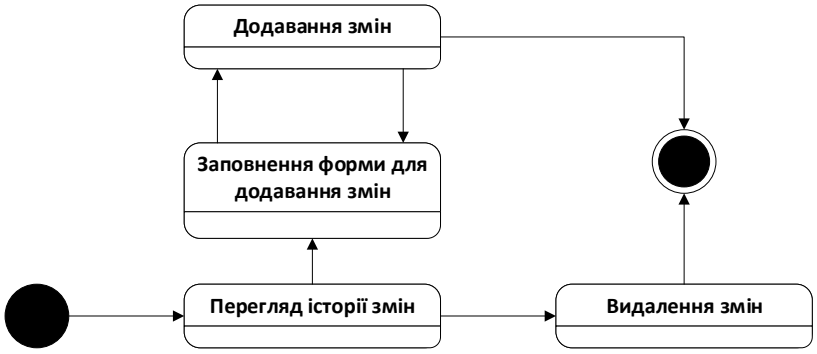


Рисунок 2.6 – Діаграма станів при роботі у вкладці «Мої результати»

Блок-схема роботи модуля реєстрації користувача наведена на рисунку 2.7.

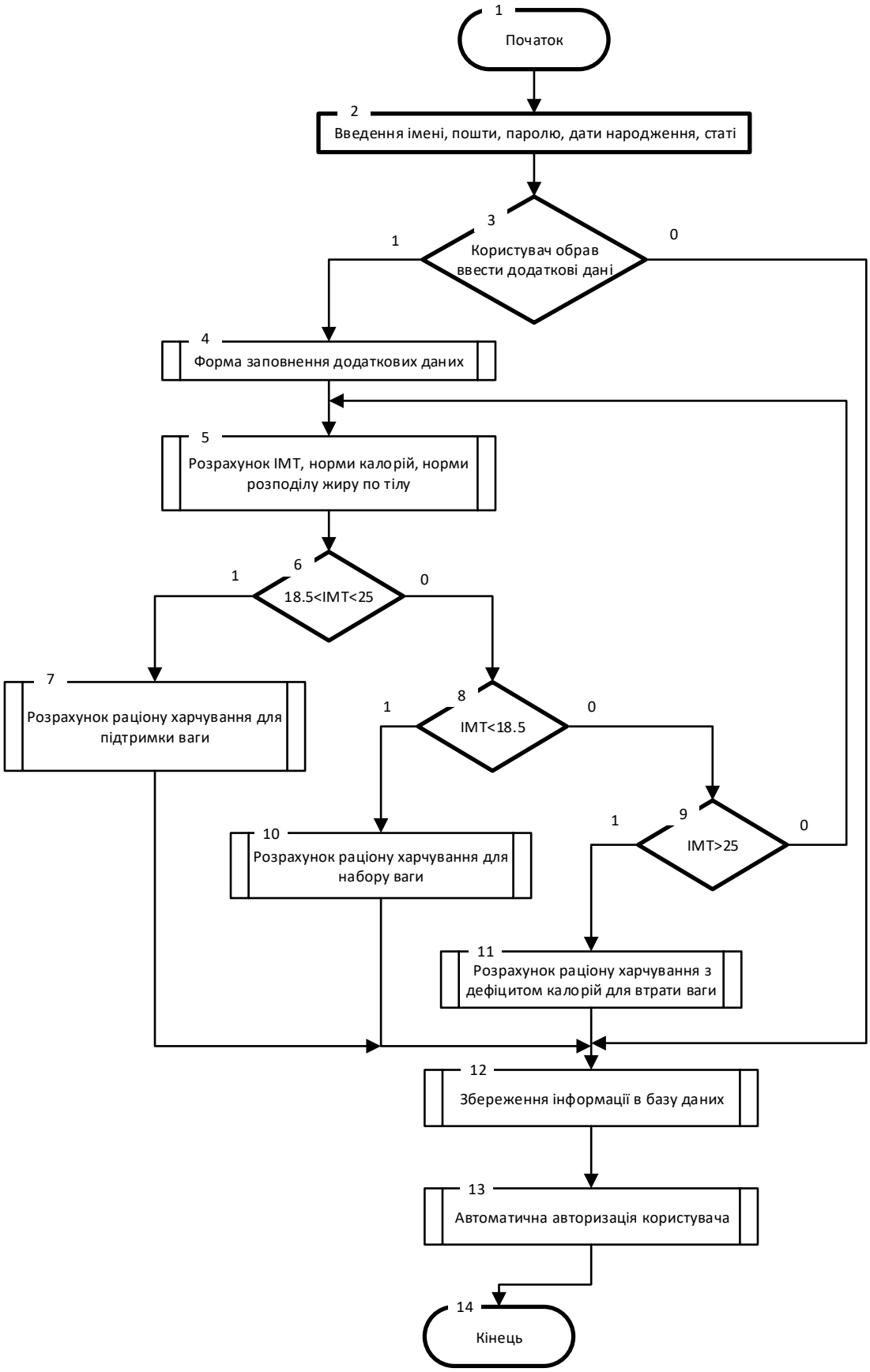


Рисунок 2.7 – Блок-схема роботи модуля реєстрації користувача

Цей метод збирає персональні дані користувача, такі як ім’я, електронна пошта, дата народження і стать. Окрім цього, він може ввести додаткові дані, на основі яких буде проведено розрахунок показників його тіла, залежно від яких буде автоматично запропоновано раціон харчування, що відповідає потребам користувача.

Загальний алгоритм додатку є основним для додатка. Він демонструє озв’язок між усіма компонентами додатка та його загальну поведінку. Загальний алгоритм роботи додатка продемонстровано на рисунку 2.8.

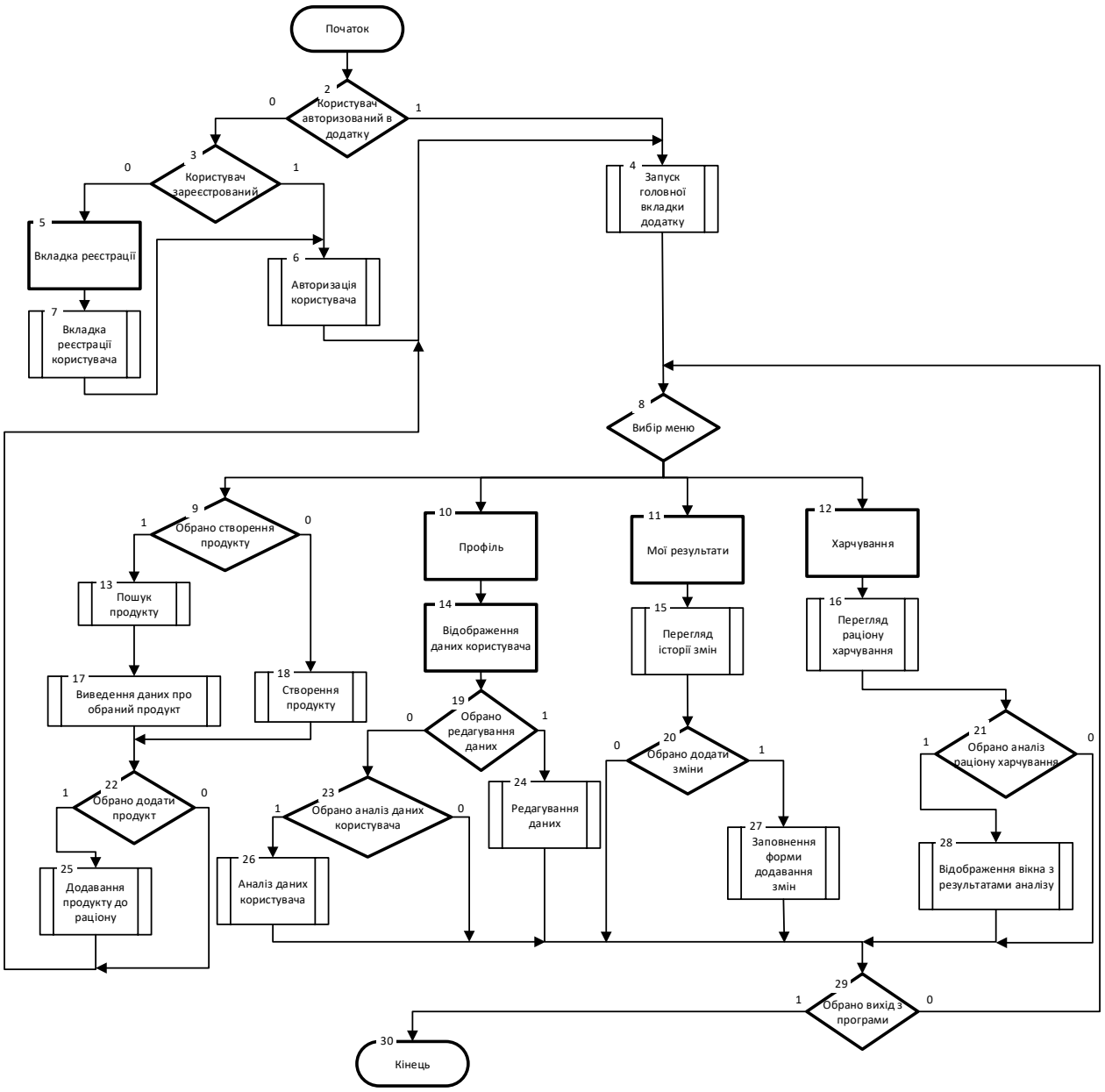


Рисунок 2.8 – Загальний алгоритм роботи додатка

2.7 Висновки

В цьому розділі було проведено аналіз даних програмного застосунку для визначення та аналізу раціону харчування людини. Визначено, які дані вводяться, обробляються та використовуються при використанні додатка. Розроблено модель компонентів додатка, продемонстровано взаємозв’язок його компонентів та використання самих компонентів. Розроблено метод отримання інформації про нутріативні цінності продуктів харчування. Побудовано діаграми станів роботи додатка та загальний алгоритм роботи додатка.

3 РОЗРОБКА ДОДАТКУ ДЛЯ ПІДБОРУ РАЦІОНУ ХАРЧУВАННЯ

3.1 Варіантний аналіз та обґрунтування вибору мови програмування для серверної частини застосунку

Для розробки додатку необхідно розробити серверну та клієнтську частину. Для серверної частини використовується Spring, що працює на мові Java. Відповідно, необхідно обрати середовище розробки для цієї мови програмування. Розглянемо найпопулярніші серед таких.

IntelliJ IDEA [12] - це популярне середовище розробки (IDE) для Java, яке використовується багатьма розробниками програмного забезпечення.

IntelliJ IDEA поєднує в собі широкий спектр інструментів, необхідних для розробки Java-додатків, таких як редактор коду, відладчик, компілятор, система контролю версій та інші.

IDE пропонує розумні підказки щодо коду, які допомагають підвищити продуктивність розробки. Автозаповнення коду та швидке виправлення помилок економлять ваш час.

IntelliJ IDEA надає потужні інструменти рефакторингу, такі як перейменування, зміна сигнатури методу, видалення неіспользовуваного коду тощо. Вбудований аналізатор коду допомагає виявляти та виправляти проблеми в коді.

IDE добре інтегрується з популярними Java-фреймворками, такими як Spring, Hibernate, Maven та Gradle, що спрощує розробку складних додатків.

IntelliJ IDEA має потужні можливості навігації по коду, включаючи пошук символів, класів, методів, переходи за посиланнями та інші. Це значно полегшує орієнтування в великих проектах.

IDE підтримує велику екосистему плагінів, які дозволяють розширювати її функціональність під конкретні потреби розробника.

Хоча IntelliJ IDEA спеціалізується на Java, вона також надає зручну підтримку для інших мов, таких як Kotlin, Groovy, Scala та JavaScript.

Eclipse [13] - це ще одне потужне середовище розробки (IDE) для Java, яке користується широкою популярністю серед розробників. Eclipse відрізняється

великою гнучкістю та розширюваністю за рахунок багатої екосистеми плагінів.

Головні переваги Eclipse:

- широкий набір інструментів для розробки, тестування та налагодження Java-додатків;

- потужні можливості рефакторингу коду та аналізу;

- зручна інтеграція з системами контролю версій, такими як Git;

- підтримка великої кількості фреймворків та бібліотек Java;

- відкритий вихідний код та велике співтовариство розробників.

NetBeans [14] - це ще одне популярне середовище розробки для Java, яке відрізняється своєю простотою та інтуїтивно зрозумілим інтерфейсом. NetBeans є безкоштовним та поширюється під ліцензією CDDL.

Головні особливості NetBeans:

- вбудовані інструменти для швидкої розробки, тестування та розгортання Java-додатків;

- потужна підтримка Java EE, Java SE, JavaFX та інших Java-технологій;

- зручна система управління проектами та модулів;

- інтегровані засоби профілювання та аналізу продуктивності;

- добра підтримка систем контролю версій, таких як Git, Subversion, Mercurial;

- великий набір плагінів для розширення функціоналу.

Складемо таблицю порівняння можливостей середовищ розробки для Java (таблиця 3.1).

Таблиця 3.1 – Порівняння середовищ розробки для Java

Характеристика	IntelliJ IDEA	Eclipse	NetBeans
Підтримка Java	+	+	+
Вбудована підтримка Spring Boot	+	+/-	+/-
Підтримка візуального редагування Spring конфігурацій	+	+	+

Продовження таблиці 3.1.

Продуктивність	+	+/-	+/-
Автодоповнення коду	+	+	+
Сумарний бал	5	4	4

Таким чином, IntelliJ IDEA є найкращим середовищем розробки для Java.

3.2 Варіантний аналіз та обґрунтування вибору мови програмування для клієнтської частини застосунку

Розглянемо середовища веб-розробки. Серед найпопулярніших існують такі середовища веб-розробки, як Visual Studio Code, Webstorm, Atom.

Atom [15] - це безкоштовний, відкритий і крос-платформний редактор коду, розроблений компанією GitHub. Він призначений для веб-розробки.

Atom заснований на модульній архітектурі, що дозволяє легко встановлювати, налаштовувати та розширювати функціональність за допомогою великої кількості пакетів та плагінів.

Atom має мінімалістичний і зручний інтерфейс користувача, який можна налаштувати під свої потреби.Редактор підтримує широкий спектр мов програмування, включаючи JavaScript, HTML, CSS, Python, Ruby, PHP та інші.

Atom прямо інтегрується з системами контролю версій, такими як Git, що спрощує командну роботу над проектами. Він дозволяє працювати одночасно з кількома проектами, файлами та вікнами, що підвищує продуктивність. Редактор пропонує розумне автодоповнення коду, що допомагає швидше писати та редагувати код. Atom включає вбудовані інструменти для налагодження коду, такі як відладчик, консоль та інспектор.

Visual Studio Code (VS Code) [16] - це популярне і потужне середовище розробки з відкритим вихідним кодом, розроблене компанією Microsoft.

VS Code працює на Windows, macOS та Linux, що робить його доступним для розробників незалежно від операційної системи.

Середовище має мінімалістичний і зручний інтерфейс користувача, який

можна налаштувати під власні потреби.

VS Code має великий вибір офіційних та сторонніх розширень, які дозволяють розширювати його можливості для підтримки різних мов, фреймворків та інструментів.

Завдяки використанню технології Electron, VS Code забезпечує швидку роботу та низьке споживання ресурсів.

Середовище добре інтегрується з системами контролю версій, такими як Git, що спрощує командну роботу над проектами. VS Code пропонує розвинений редактор коду з функціями автодоповнення, підсвічування синтаксису, навігації та рефакторингу.

Середовище включає вбудований відладчик, який допомагає знаходити та виправляти помилки в коді. VS Code інтегрується з вбудованим термінальним вікном, що дає змогу виконувати командні операції безпосередньо в IDE. VS Code підтримує широкий спектр мов програмування, включаючи JavaScript, TypeScript, Python, Java, C++, C#, PHP та інші.

WebStorm [17] - це потужне інтегроване середовище розробки (IDE), розроблене компанією JetBrains для веб-розробки.

WebStorm надає всебічну підтримку для JavaScript, TypeScript, React, Angular, Vue.js, Node.js та інших популярних технологій веб-розробки.

IDE пропонує інтелектуальне автозаповнення коду, що допомагає підвищувати продуктивність розробки. Вбудована система рефакторингу: WebStorm має потужні інструменти для безпечного рефакторингу коду, включаючи перейменування змінних, методів та класів. IDE тісно інтегрується з такими фреймворками як React, Angular, Vue.js, Node.js, що спрощує розробку та налагодження веб-додатків.

WebStorm має вбудований інструмент для покрокового налагодження JavaScript-коду з можливістю перегляду стану змінних. Середовище надає зручну інтеграцію з популярними системами контролю версій, такими як Git, Subversion та Mercurial.

WebStorm дозволяє одночасно відкривати кілька файлів та проектів, що

підвищує ефективність роботи. IDE надає велику екосистему плагінів та додаткових інструментів, що дозволяють адаптувати її під конкретні потреби розробника. WebStorm інтегрується з популярними фреймворками тестування, такими як Jest, Mocha, Jasmine, Karma та інші. Завдяки оптимізації продуктивності, WebStorm демонструє швидку роботу навіть з великими проектами.

3.3 Розробка серверної частини додатка

Вибір відповідного веб-клієнта для розробки серверної частини додатку є важливим рішенням, яке може значно вплинути на продуктивність та масштабованість системи. Це особливо актуально, так як в розроблюваному виконуються запити до зовнішнього API для отримання інформації про продукти харчування.

Так як для розробки використовується фреймворк Spring, розглянемо веб-клієнти, що доступні в ньому, а саме RestTemplate і WebClient [18].

RestTemplate: - це синхронний клієнт для виконання HTTP-запитів. Він абстрагує складність створення запитів та обробки відповідей. RestTemplate забезпечує зручні можливості, такі як маршалінг/демаршалінг об'єктів, обробка помилок тощо. Він простий у використанні, але не підтримує асинхронну обробку запитів.

WebClient: - це асинхронний реактивний клієнт для виконання HTTP-запитів. WebClient є частиною реактивного стеку Spring Web. Він повністю не блокуючий, струменевий та оптимізований для високої продуктивності. WebClient забезпечує всі необхідні функції для сучасного асинхронного програмування, але складніший у використанні порівняно з RestTemplate.

Характеристики RestTemplate:

1. Синхронний, блокуючий клієнт, який виконує один HTTP-запит за раз.
2. Простий у використанні та налаштуванні.
3. Забезпечує автоматичне маршалювання/демаршалювання об'єктів у/з різних форматів (JSON, XML тощо).
4. Підтримує зручну обробку помилок та винятків.

5. Можливість використання різних стратегій обробки відповідей.

6. Обмежені можливості для асинхронної обробки запитів.

7. Може призвести до блокування потоків та погіршення продуктивності при високому навантаженні.

Характеристики WebClient:

1. Асинхронний, неблокуючий клієнт, здатний обробляти багато запитів одночасно.

2. Складніший у використанні та налаштуванні порівняно з RestTemplate.

3. Також забезпечує автоматичне маршалювання/демаршалювання об'єктів.

4. Реактивний підхід на основі проектів Reactor та RxJava.

5. Підтримує обробку помилок та винятків у реактивному стилі.

6. Можливість створювати складні ланцюжки операцій над HTTP-запитами.

7. Оптимізований для високої продуктивності та ефективного використання ресурсів.

8. Краще масштабується при високому навантаженні.

У порівнянні з RestTemplate, WebClient більш складний у використанні, але він дозволяє легко справлятися з високим навантаженням додатка завдяки асинхронній та неблокуючій природі. Це дозволяє ефективно розподіляти ресурси та уникати блокування потоків, забезпечуючи кращу продуктивність та досвід користувачів.

Його використання дозволить уникнути блокування та забезпечити кращу продуктивність і досвід користувачів при великій кількості одночасних запитів.

3.4 Розробка структурних схем інтерфейсу

Структурні схеми інтерфейсу необхідні дизайнерам та розробникам при розробці та реалізації інтерфейсу користувача застосунку. Структурні схеми являють собою спрощений вигляд інтерфейсу, на ній прописуються складові вікон графічного інтерфейсу користувача.

Головне вікно застосунку дозволяє переглядати загальну інформацію щодо нутріативних цінностей раціону харчування користувача за сьогодні, надає

функціонал для перегляду інформації про продукти харчування, здійснювати їх пошук, та відображає історію останніх дій користувача.

Структурна схема головного вікна застосунку наведена на рисунку 3.1.

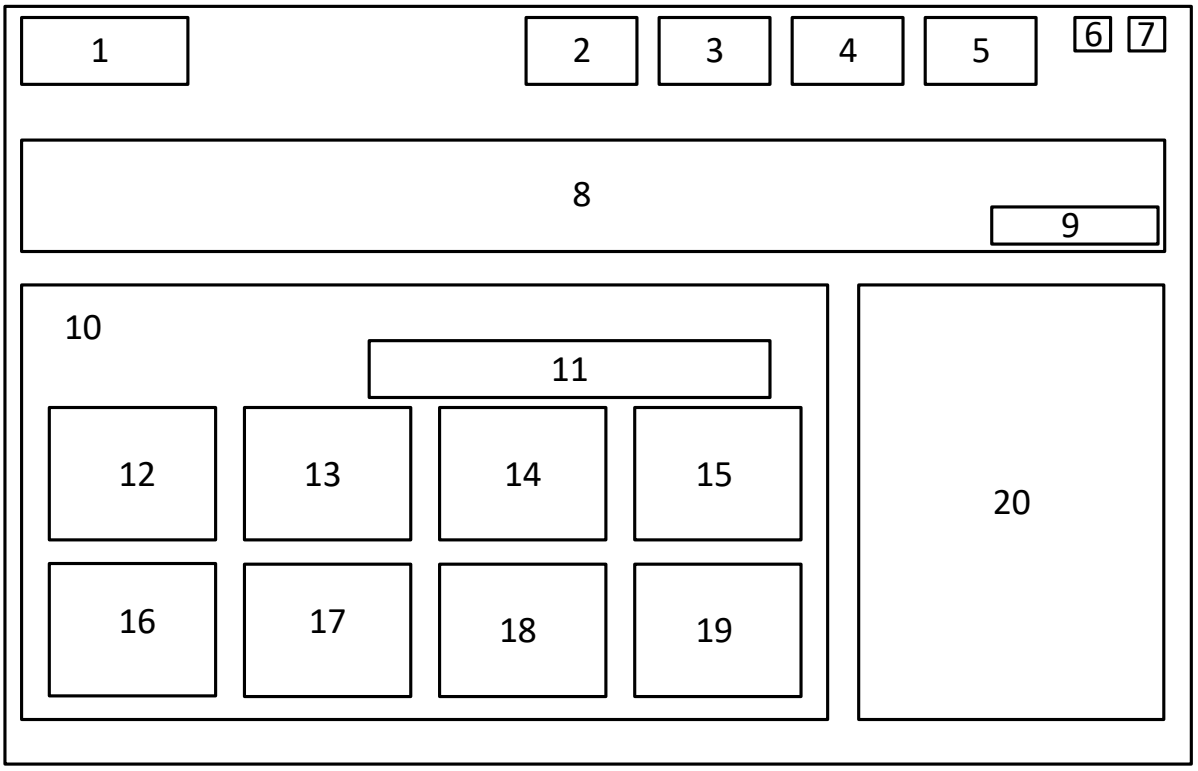


Рисунок 3.1 – Структурна схема головного вікна програми

Елементи головного вікна:

1. Назва програми.
2. Кнопка панелі інструментів.
3. Кнопка списку продуктів.
4. Кнопка статистики.
5. Кнопка профілю.
6. Логотип профілю.
7. Кнопка пошуку.
8. Панель інформації щодо нутріативної цінності раціону харчування.
9. Кнопка для додавання продуктів.
10. Панель зі списком продуктів.
11. Панель пошуку

12-19. Записи продуктів.

20. Панель активності.

Структурна схема вікна реєстрації наведена на рисунку 3.2.

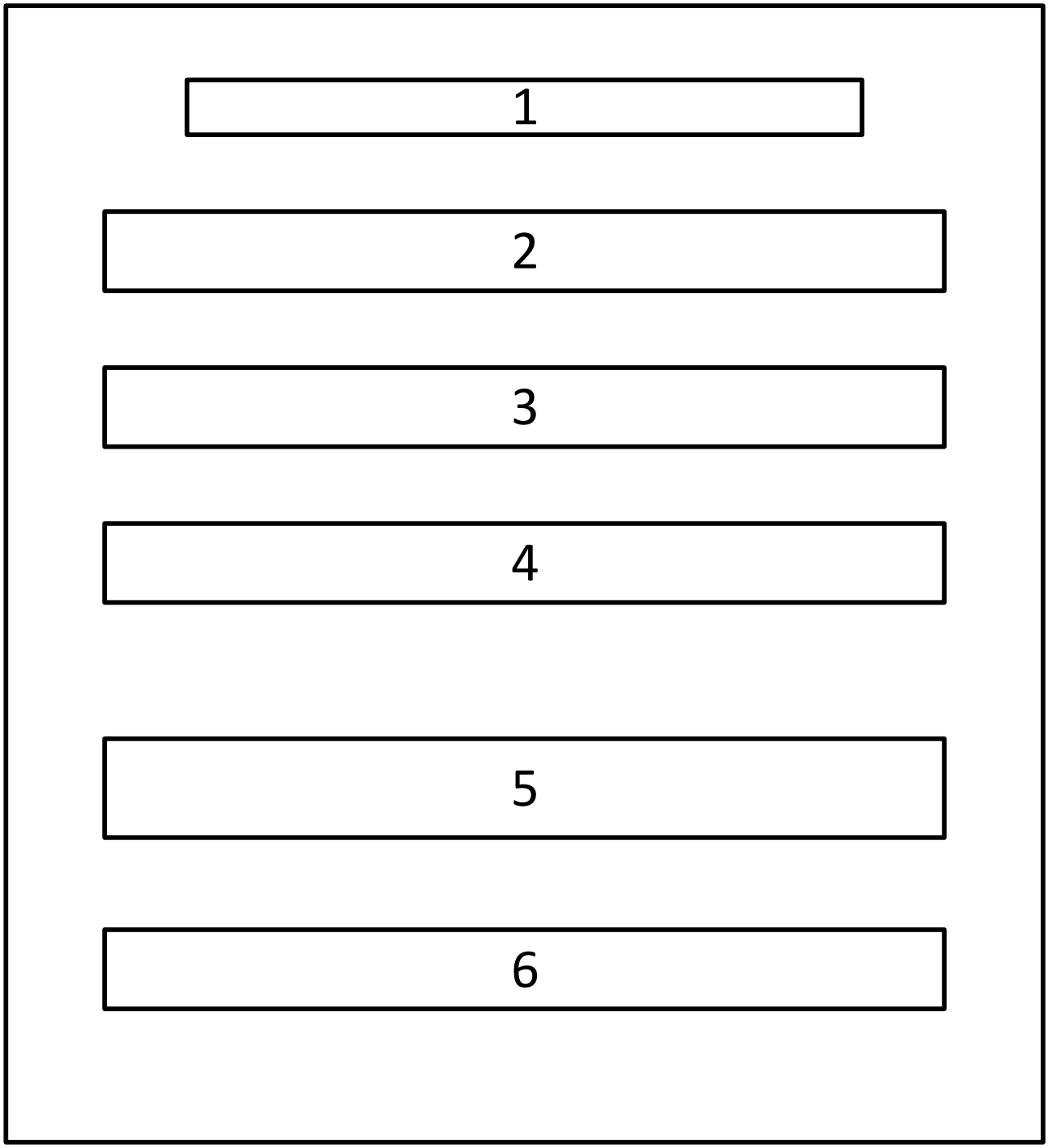


Рисунок 3.2 – Структурна схема вікна реєстрації

Елементи вікна реєстрації:

1. Кнопка переходу на вікно авторизації.

2. Поле для введення імені.
3. Поле для введення електронної адреси.
4. Поле для введення пароля.
5. Поле для проходження reCAPTCHA.
6. Кнопка створення аккаунту.

Вікно реєстрації дозволяє користувачу створити аккаунт, за допомогою якого він користуватиметься системою і в якому зберігатиметься його персональна інформація, раціон харчування та історія дій.

При реєстрації від користувача вимагається пройти перевірку reCAPTCHA.

reCAPTCHA - це безкоштовний інструмент від компанії Google, призначений для захисту вебсайтів від шкідливих ботів та спаму. Це система підтвердження людськості, яка використовує виклики для перевірки того, чи є користувач людиною, а не автоматизованою програмою.

reCAPTCHA працює шляхом аналізу поведінки користувача та використання передових алгоритмів машинного навчання для визначення рівня ризику. Якщо система не може однозначно визначити, чи є користувач людиною, вона може запропонувати додаткові перевірки, такі як вибір зображень, що відповідають певному критерію, або введення тексту з зображення.

Завдяки простоті використання та ефективності, reCAPTCHA став широко використовуватися на вебсайтах по всьому світу. Це допомагає підвищити безпеку та покращити досвід користувачів, запобігаючи шкідливій діяльності ботів та спаму.

Вікно авторизації використовується для входу користувача в систему.

Для цього, користувач вводить такі дані, як логін та пароль.

Також, з цього вікна він може здійснити перехід на вікно реєстрації, якщо він не має аккаунта в застосунку або бажає створити новий.

Окрім цього, він має можливість відновити пароль, якщо не пам’ятає його.

Структурна схема вікна авторизації наведена на рисунку 3.3.

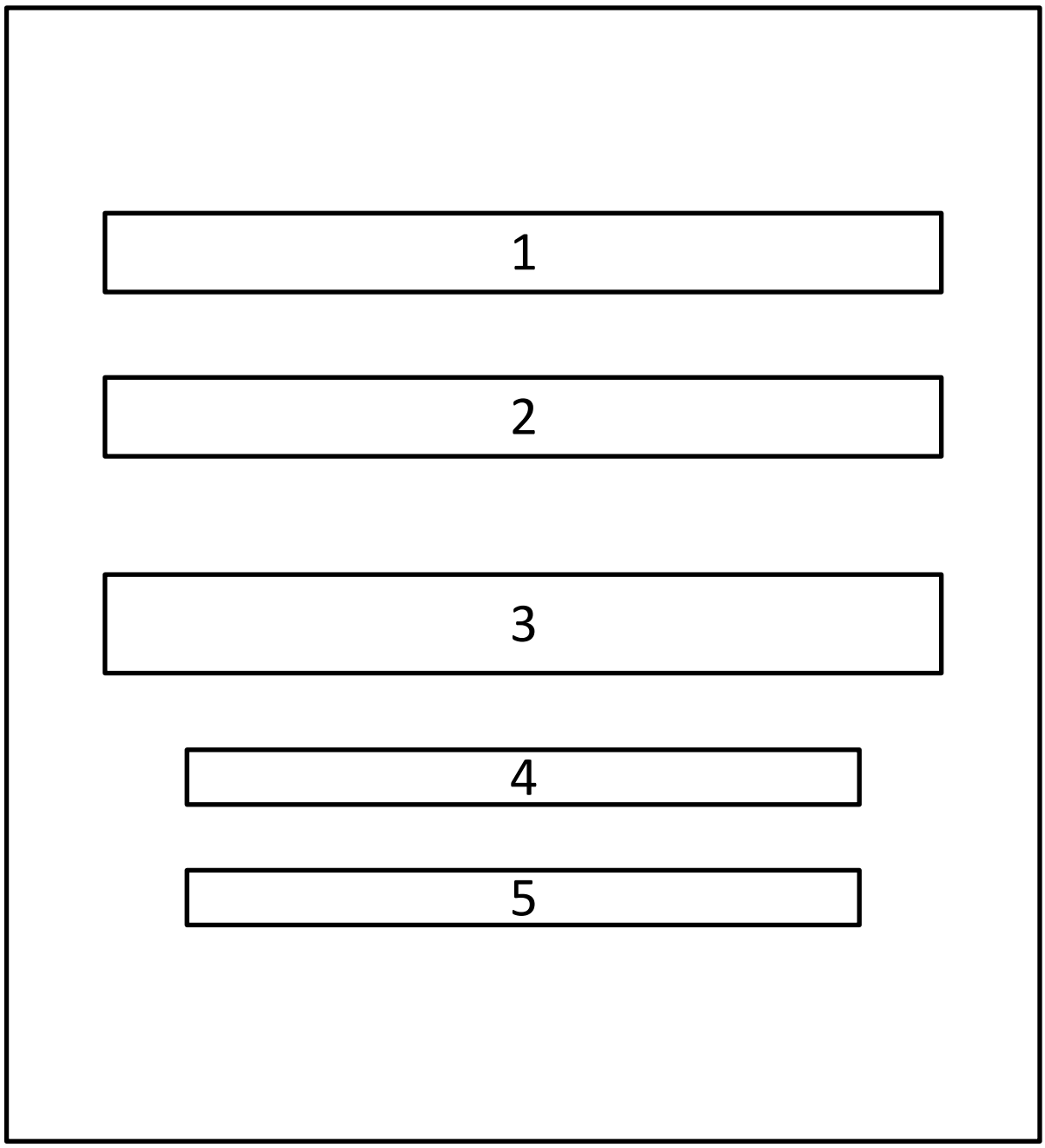


Рисунок 3.3 – Вікно авторизації

Елементи вікна авторизації:

1. Поле введення електронної адреси.
2. Поле для введення паролю.
3. Кнопка для входу в аккаунт.
4. Кнопка для відновлення паролю.
5. Кнопка для переходу на вікно реєстрації.

Вікно «Мої страви» дозволяють переглядати нутріативну цінність спожитих

продуктів за обраний день. Він має календар для перегляду раціону харчування за певний день, та розподіляє раціон на сніданок, обід, перекус та вечерю.

Структурна схема вікна «Мої страви» наведено на рисунку 3.4.

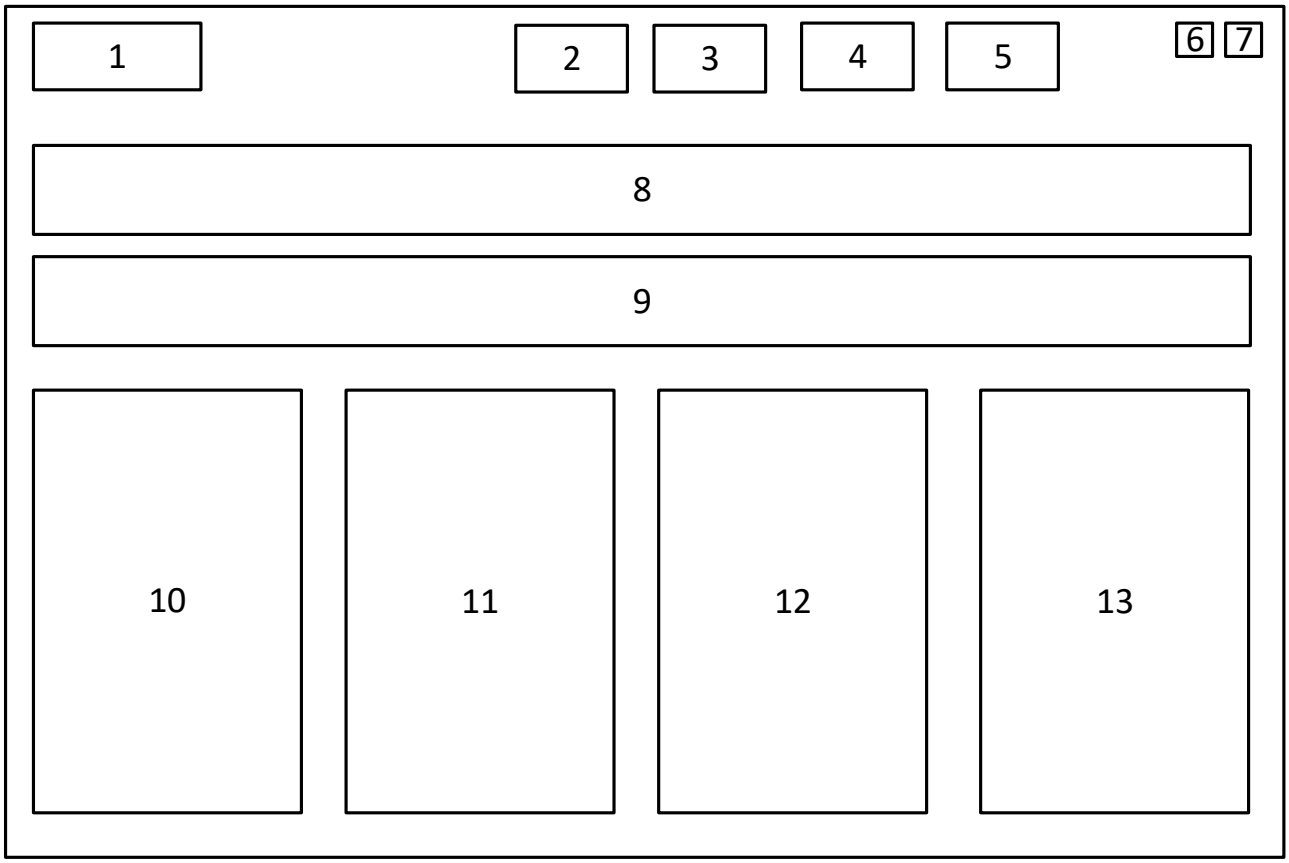


Рисунок 3.4 – Структурна схема вікна «Мої страви»

1. Назва програми.
2. Кнопка панелі інструментів.
3. Кнопка списку продуктів.
4. Кнопка статистики.
5. Кнопка профілю.
6. Логотип профілю.
7. Кнопка пошуку.
8. Календар.
9. Розрахунок нутріативних цінностей раціону харчування за обраний день.
10. Опис сніданку.

11. Опис обіду.
12. Опис перекусів.
13. Опис вечері.

Структурна схема вікна аналізу раціону харчування наведена на рисунку 3.5.

Це вікно розраховує нутріативні цінності раціону харчування за певний день та порівнює їх зі стандартними значеннями.

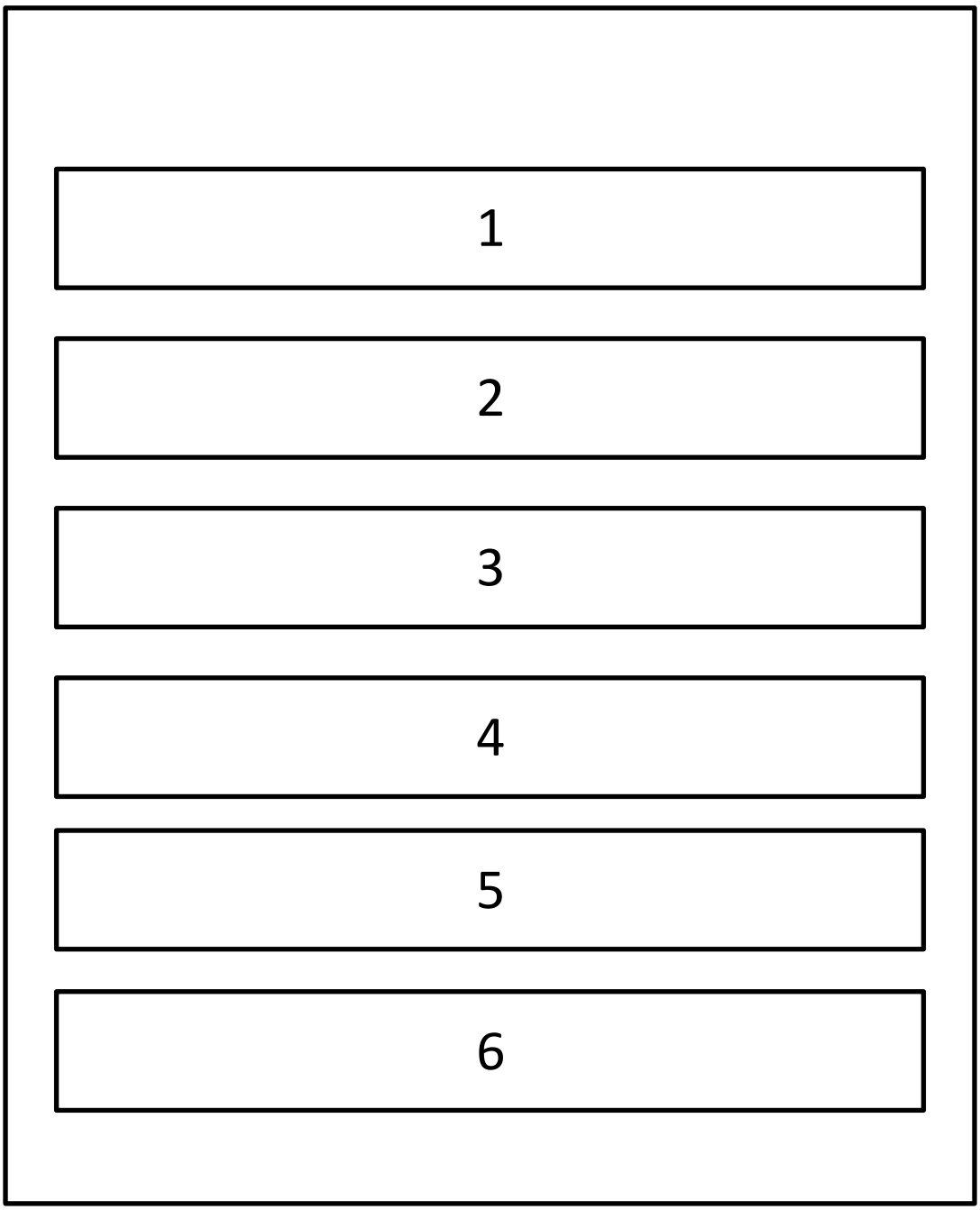


Рисунок 3.5 – Структурна схема вікна аналізу раціону харчування

Елементи вікна аналізу раціону харчування:

1. Кількість калорій.
2. Кількість карбогідратів.
3. Кількість жирів.
4. Кількість насичених жирів.
5. Кількість протеїнів.
6. Кількість цукру.

Вікно додавання нового продукту дозволяє завантажити ті продукти харчування, яких немає в системі. Користувач має самостійно внести інформацію про нутріативні цінності цього продукта, так як він невідомий системі.

Структурна схема вікна додавання нового продукту наведена на рисунку 3.6.

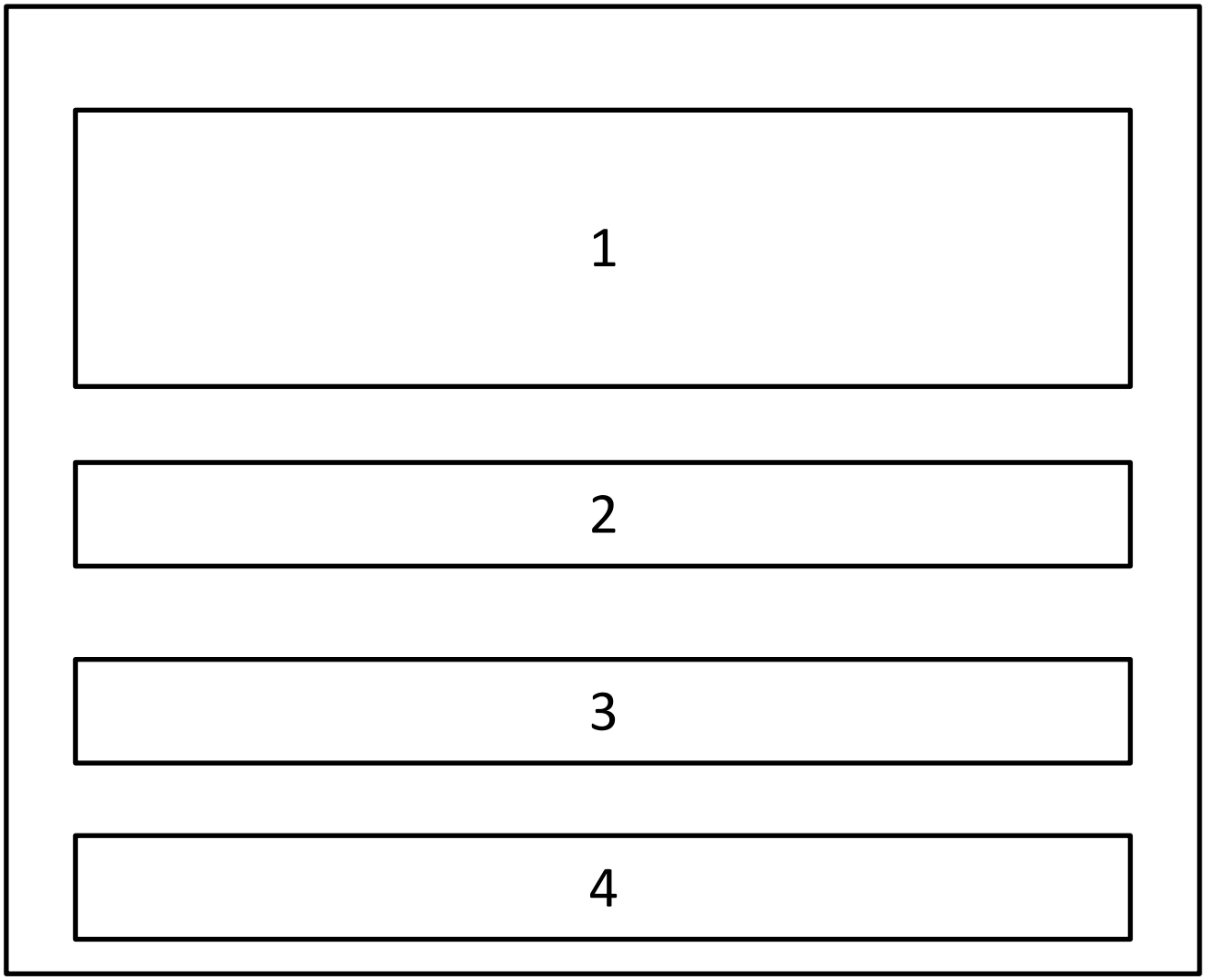


Рисунок 3.6 – Структурна схема вікна додавання нового продукту

Елементи вікна додавання нового продукту:

1. Поле для завантаження фото.
2. Назва продукту.
3. Кількість калорій.
4. Кнопка «Додати до раціону».

Вікно профілю користувача відображає загальну інформацію про користувача, таку як фото, ім'я, електронну пошту, вік, а також його фізичні параметри. Також, тут проводиться розрахунок показників тіла користувача. Структурну схему вікна профілю користувача подано на рисунку 3.7.

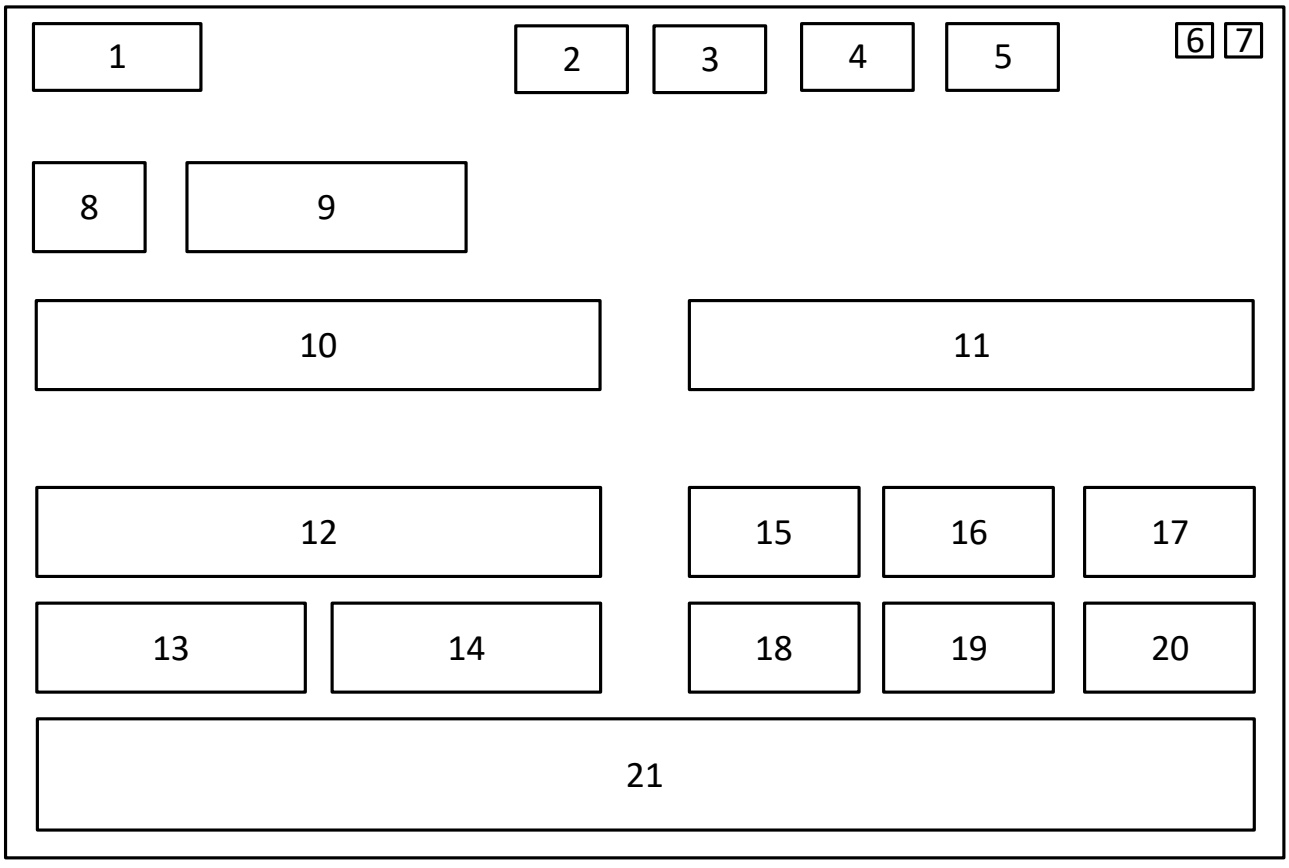


Рисунок 3.7 – Структурна схему вікна профілю користувача

Елементи вікна профілю користувача:

1. Назва програми.
2. Кнопка панелі інструментів.
3. Кнопка списку продуктів.

4. Кнопка статистики.
5. Кнопка профілю.
6. Логотип профілю.
7. Кнопка пошуку.
8. Фото користувача.
9. Ім'я користувача.
10. Електронна адреса користувача.
11. Номер телефону користувача.
12. Дата народження користувача.
13. Вага користувача.
14. Зріст користувача.
15. Обхват грудей.
16. Обхват талії.
17. Обхват бедер.
18-20. Панель з цілями користувача щодо його ваги.
21. Розрахунок показників тіла користувача.

3.5 Висновки

В цьому розділі було проведено варіантний аналіз та обґрунтування засобів розробки програмного застосунку. Розроблено алгоритми роботи застосунка, побудовано структурні схеми інтерфейсу застосунку. Розроблено програмний застосунок для аналізу та визначення раціону людини.

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Аналіз методів тестування

Однією з функцій тестування є регулярне описання особливостей досліджуваної системи, передача інформації про неї та її поведінку. Під час розробки розробники часто використовують методологію розробки через тестування (Test Driven Development) для створення модульних тестів. Команди розробників використовують приймальні тести для складання специфікацій системи та механізм неперервної інтеграції для запобігання регресії. Незалежно від їхньої корисності, потрібні тести вищого рівня, щоб запобігти виникненню дефектів. На рисунку 4.1 представлена піраміда автоматизації тестування, яка відображає всі необхідні тести для професійної розробки.

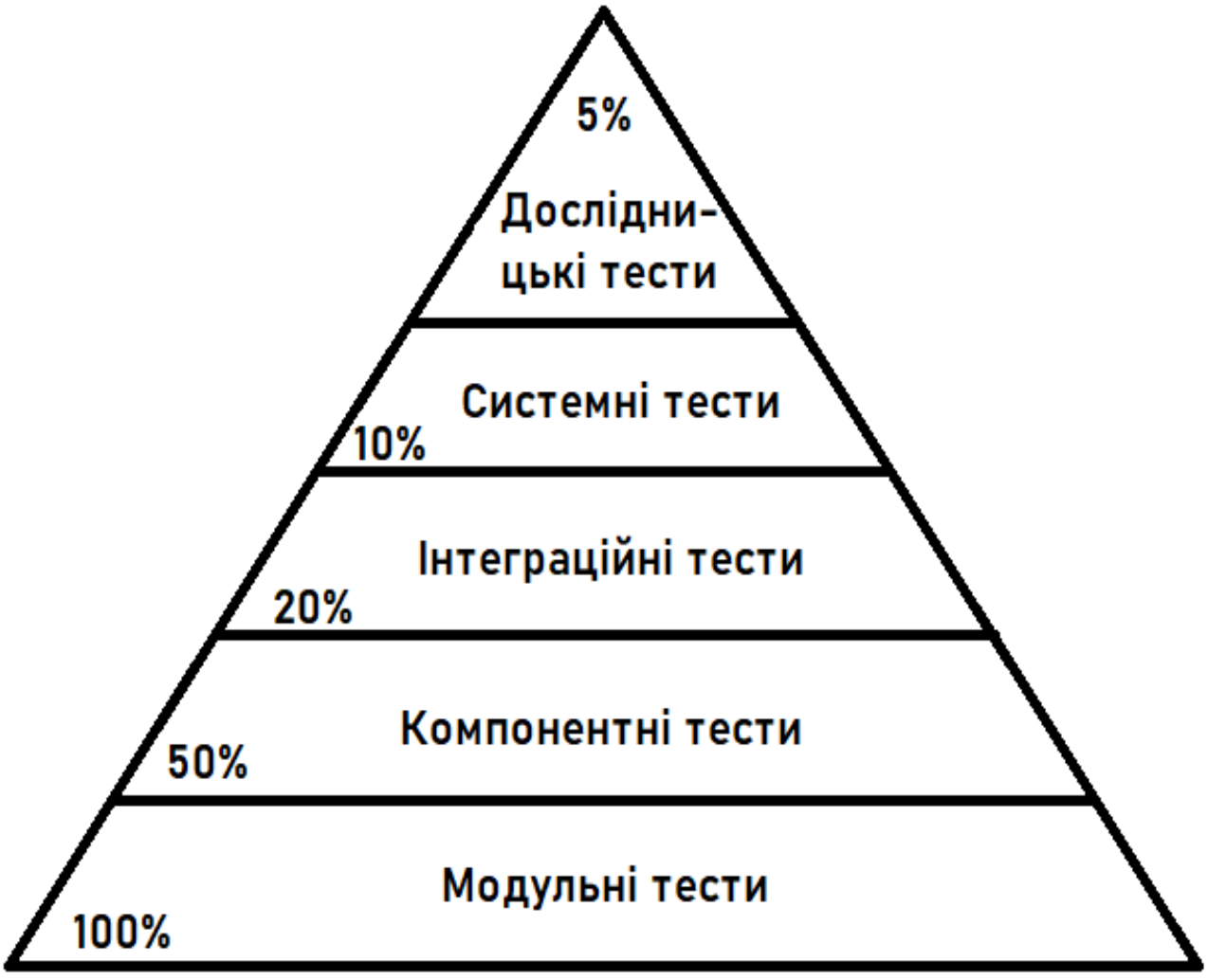


Рисунок 4.1 – Піраміда автоматизації тестування

Модульні тести знаходяться в основі піраміди. Їх створюють програмісти для програмістів за допомогою мови програмування системи. Ці тести визначають специфікації системи на найнижчому рівні та забезпечують виконання початкових планів програмістів в контексті безперервної інтеграції. Модульні тести повинні охоплювати код максимально близько до 100%, з реальним покриттям, а не фальшивими тестами. Для модульного тестування використовується технологія JUnit 5. Модульні тести покривають окремі методи програми. JUnit 5 — це сучасний фреймворк для модульного тестування Java, який є наступником популярного JUnit 4. Він був розроблений з урахуванням найкращих практик попередніх версій та додає нові можливості для поліпшення процесу тестування. Приклад успішного проходження та непроходження unit-тесту в JUnit наведено на рисунку 4.2.

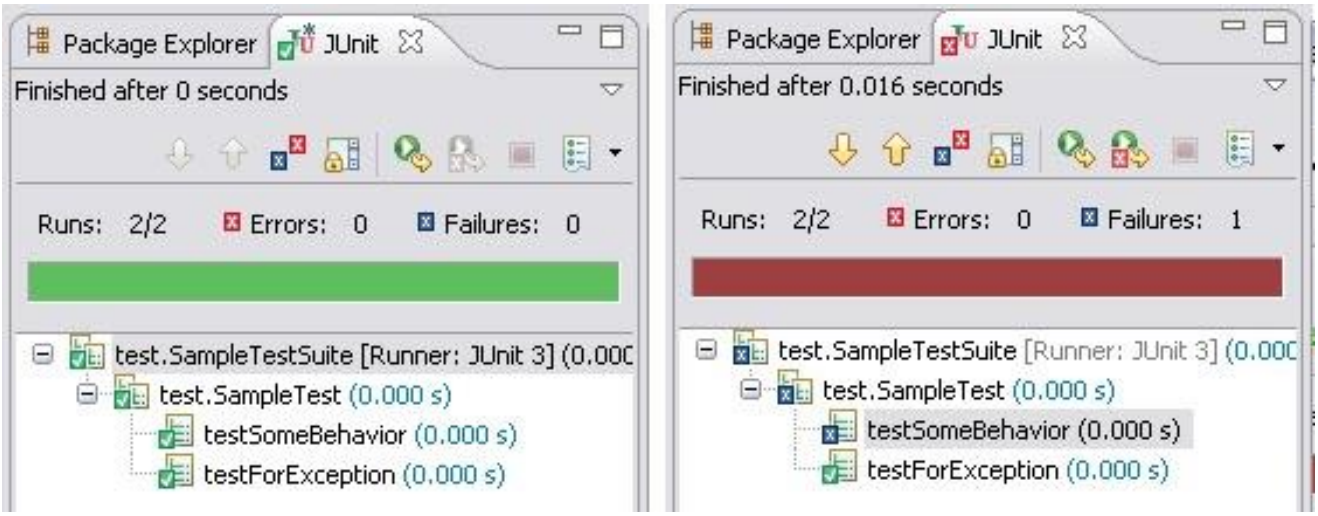


Рисунок 4.2 – Приклад запуску тесту в JUnit

Компонентні тести включають частину приймальних тестів, зазвичай написаних для окремих компонентів системи. Вони перевіряють бізнес-правила, інкапсульовані в компонентах системи. Компонентні тести ізолюють окремий компонент, передають йому вхідні дані, отримують результати та перевіряють їх відповідність. Для ізоляції інших компонентів використовуються макети та тест-дублери. Ці тести розробляються спеціалістами з контролю якості за допомогою розробників, і пишуться так, щоб їх могли читати та інтерпретувати бізнес-користувачі.

Компонентні тести покривають приблизно половину системи та орієнтовані на оптимістичні ситуації і очевидні граничні умови. Технології: JUnit 5 + WireMock.

Компонентні тести перевіряють окремі мікросервіси, де взаємодія з іншими мікросервісами мокується. Тести відтворюють бізнес-правила, викликають вхідні точки та перевіряють результати інтеграції, наприклад, чи збереглися дані в базі або чи повернулися правильні дані.

WireMock – це інструмент для створення симульованих HTTP-сервісів (моків), який широко використовується для тестування мікросервісів. Він дозволяє розробникам і тестувальникам створювати і керувати моками, що імітують поведінку реальних сервісів.

Архітектура WireMock наведена на рисунку 4.3.

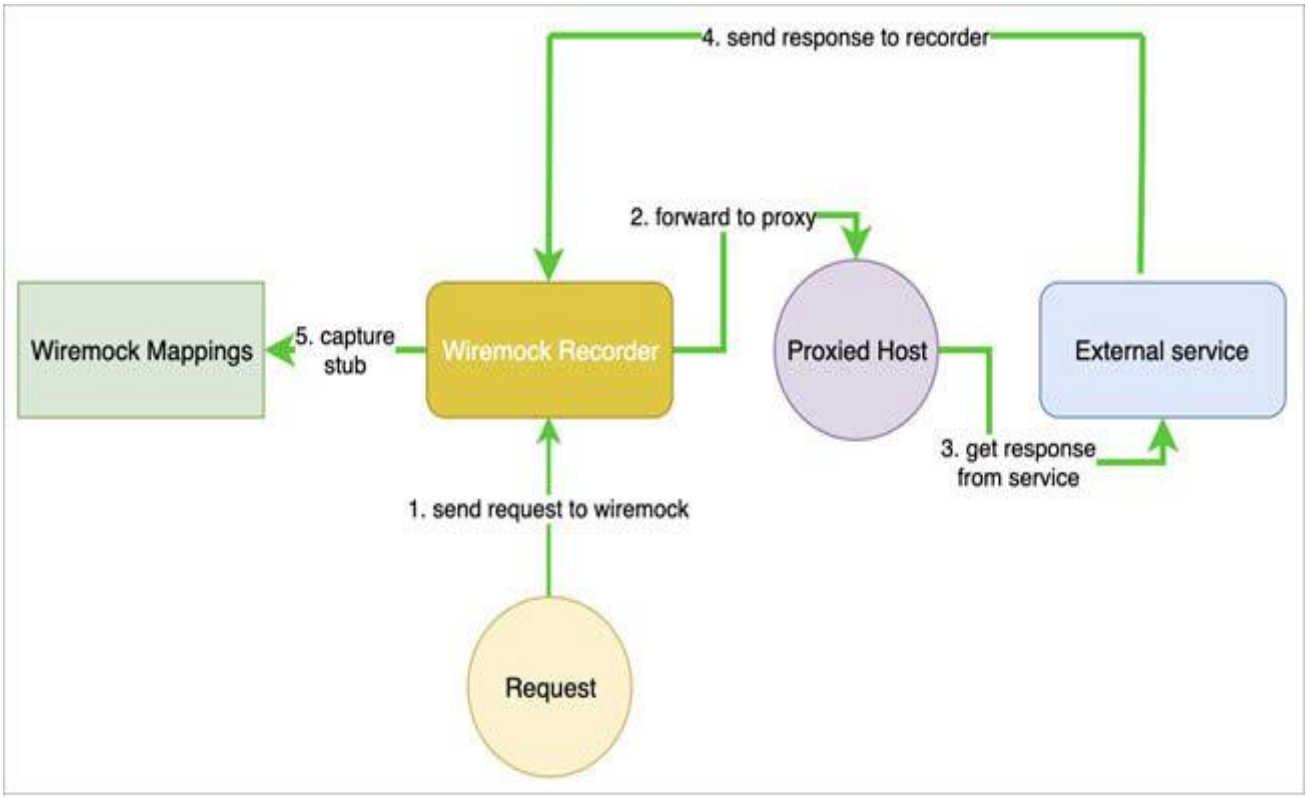


Рисунок 4.3 – Архітектура WireMock

Інтеграційні тести є актуальними для великих систем з багатьма компонентами. Вони групують компоненти системи та ізолюють окремі компоненти, замінюючи їх макетами та тест-дублерами. Ці тести перевіряють

правильність поведінки між компонентами, зосереджуючись на продуктивності та пропускній здатності. Інтеграційні тести зазвичай пишуться на тій же мові та в тому ж середовищі, що й компонентні тести, але виконуються періодично через їх тривалість. Для цього використовуються технології Docker, JUnit 5. Інтеграційні тести перевіряють взаємодію між мікросервісами, піднімаючи Docker-контейнери з усіма компонентами та проганяючи тести для перевірки обміну даними між мікросервісами.

Архітектура Docker продемонстрована на рисунку 4.4.

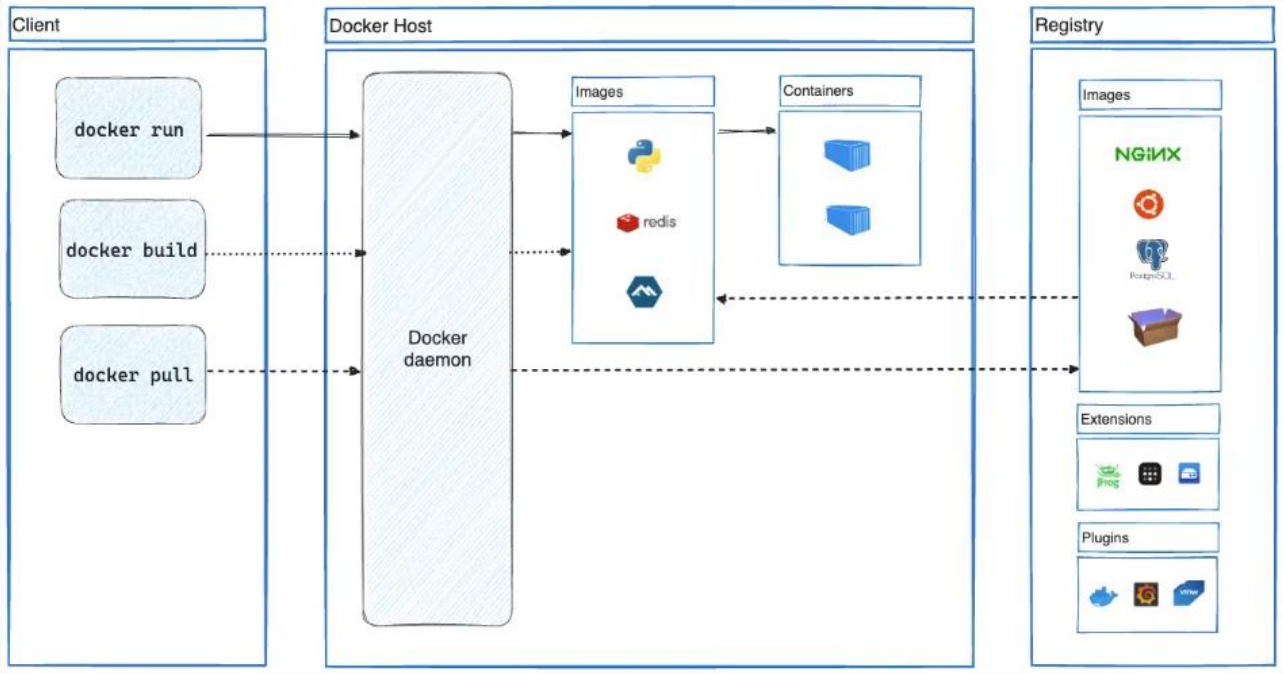


Рисунок 4.4 – Архітектура Docker

Docker є ключовим інструментом для проведення інтеграційних тестів у розробці програмного забезпечення. Інтеграційні тести перевіряють взаємодію між різними компонентами системи, і Docker дозволяє створювати ізольовані середовища, які допомагають забезпечити надійність та відтворюваність цих тестів.

Системні тести перевіряють роботу всієї інтегрованої системи, представляючи граничний випадок інтеграційних тестів. Вони перевіряють правильність взаємодії компонентів системи. Ці тести пишуться системними архітекторами та провідними технічними спеціалістами, і покривають 10%

системи, оскільки призначені для перевірки конструкції, а не поведінки системи. Для цього використовуються технології Karate Framework та Docker. Системні тести перевіряють API через відкритий інтерфейс без знання деталей реалізації. Вони піднімають всі сервіси в Docker-контейнері, роблять запити до API та перевіряють відповіді, автоматизуючи те, що можна зробити вручну через Postman або Swagger.

Karate Framework — це потужний інструмент для автоматизації тестування API, створений для того, щоб забезпечити простоту та ефективність тестування сервісів REST і SOAP. З точки зору системних тестів, Karate Framework надає ряд переваг та функцій, що роблять його особливо корисним для тестування складних інтеграційних сценаріїв.

Дослідницькі тести не автоматизуються та виконуються вручну, їх мета – дослідження системи та підтвердження її очікуваної поведінки. Вони потребують людського розуму та досвіду, і їх проведення не підлягає плануванню. У деяких групах є спеціалісти для виконання таких тестів, а в інших групах організовуються «дні полювання на помилки», коли всі співробітники намагаються знайти дефекти в системі. Покриття коду не є метою дослідницького тестування; метою є перевірка поведінки системи в умовах реального використання.

Методологія TDD є потужним інструментом визначення та перевірки дотримання вимог, а приймальні тести допомагають забезпечити відповідність цим вимогам. Однак, вони є лише частиною загальної стратегії тестування. Для досягнення мети "контроль якості не повинен виявляти дефекти" команди розробників повинні тісно співпрацювати зі службою контролю якості для створення ієрархії модульних, компонентних, інтеграційних, системних та дослідницьких тестів. Тести слід виконувати якомога частіше для забезпечення максимального зворотного зв'язку та постійної якості системи.

4.2 Тестування програми

Вікно реєстрації в застосунку наведено на рисунку 4.5.

Користувачу необхідно ввести електрону адресу, ім’я та придумати пароль.

Кнопка для сворення аккаунту стане активною після заповнення усіх необхідних даних та проходження перевірки reCAPTCHA.

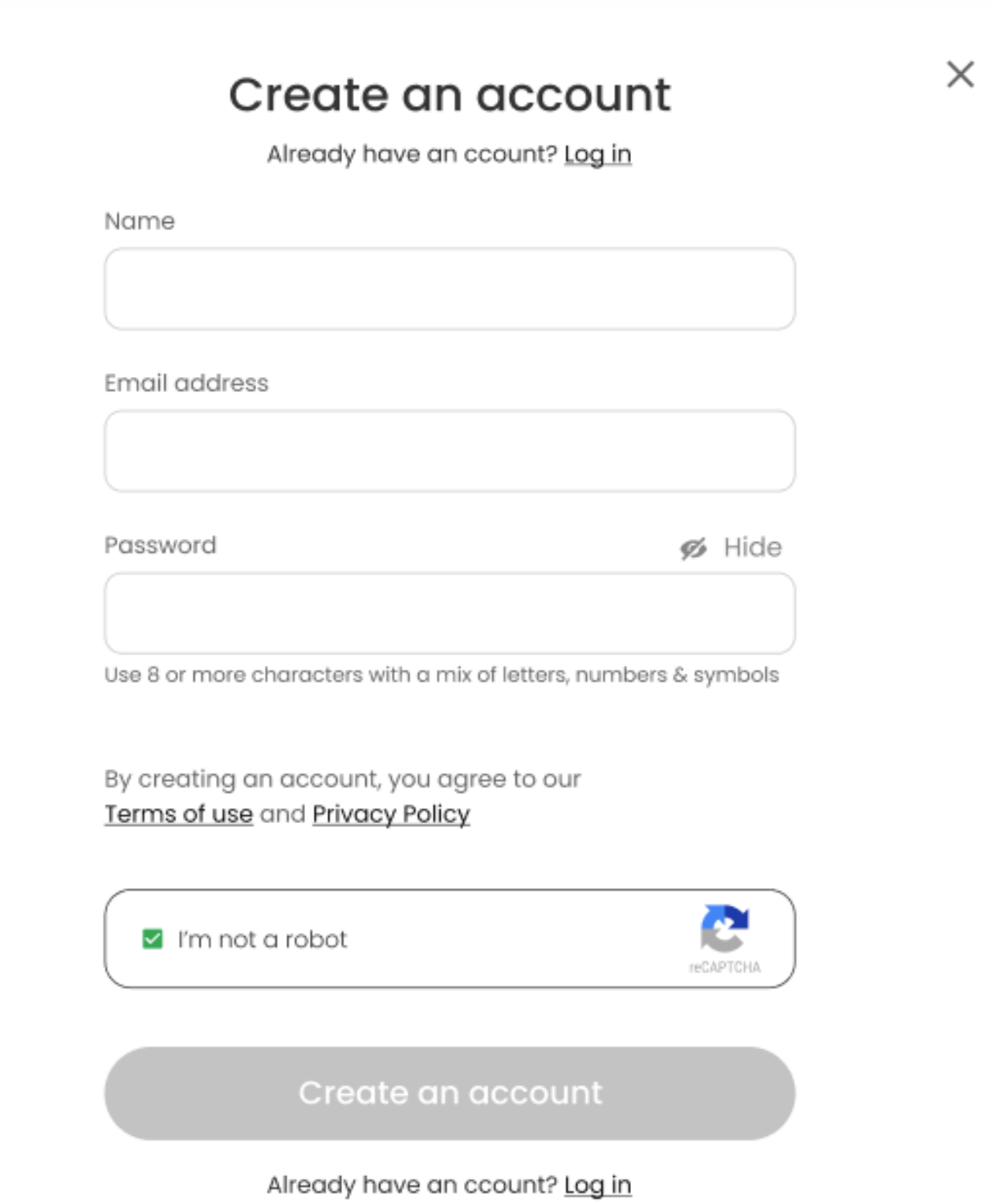


Рисунок 4.5 – Вікно реєстрації

Вікно авторизації наведено на рисунку 4.6.

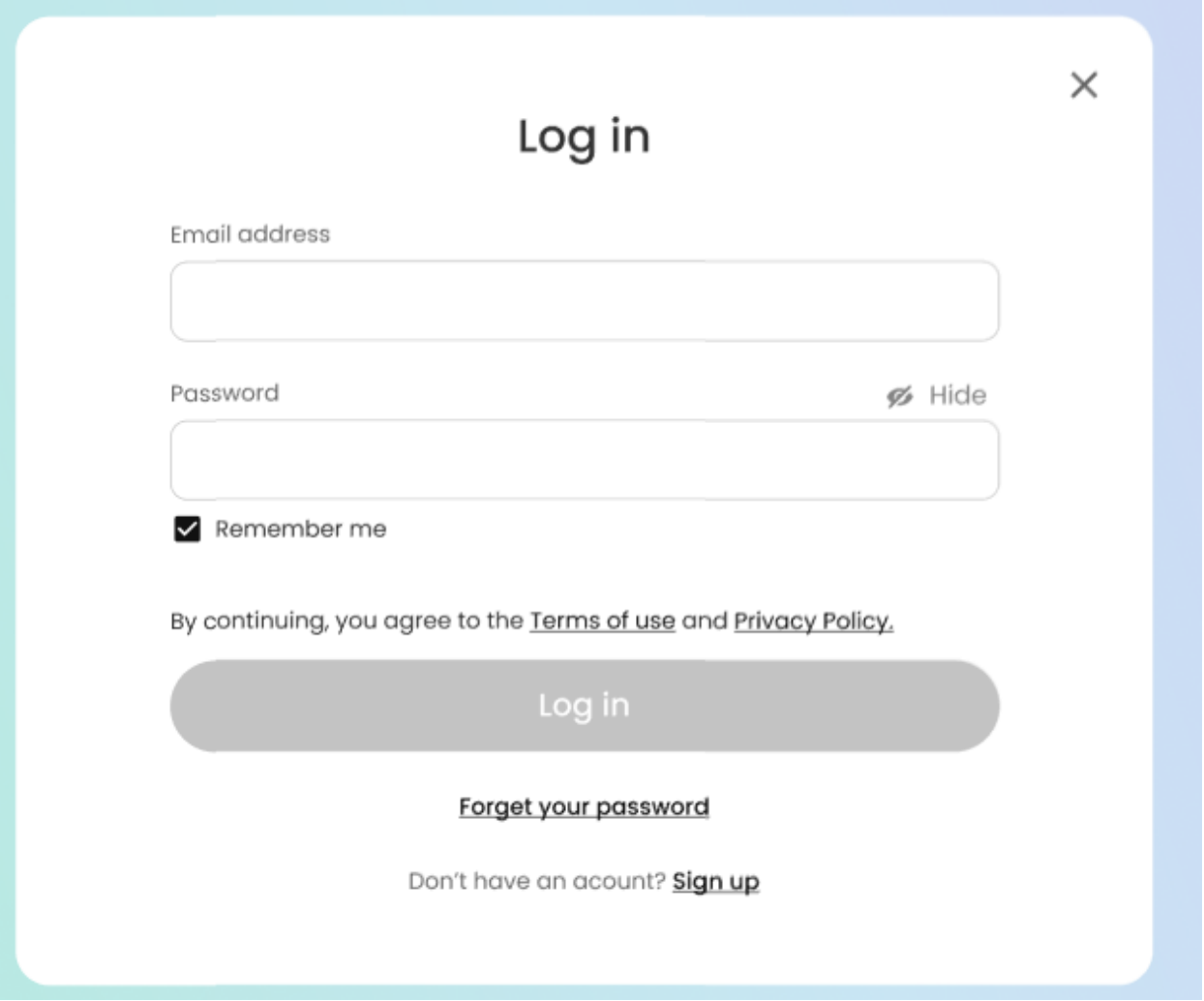


Рисунок 4.6 – Вікно авторизації

Після успішної авторизації, відбувається перехід на головну сторінку застосунку.

На рисунку 4.7 продемонстровано головну сторінку програми.

Головна сторінка відображає кількість калорій за поточний день повністю та розподілену кількість за типами прийому їжі.

Нижче відображені продукти, які збережені у базі даних застосунку з можливістю пошуку та створення власного продукту.

У правій частині вікна відображається активність користувача, а саме історія доданих у раціон продуктів за останній час. Це зменшує вірогідність помилкового додавання продукту в раціон повторно та необхідність переходити в інше вікно для

перегляду проміжного стану раціону.

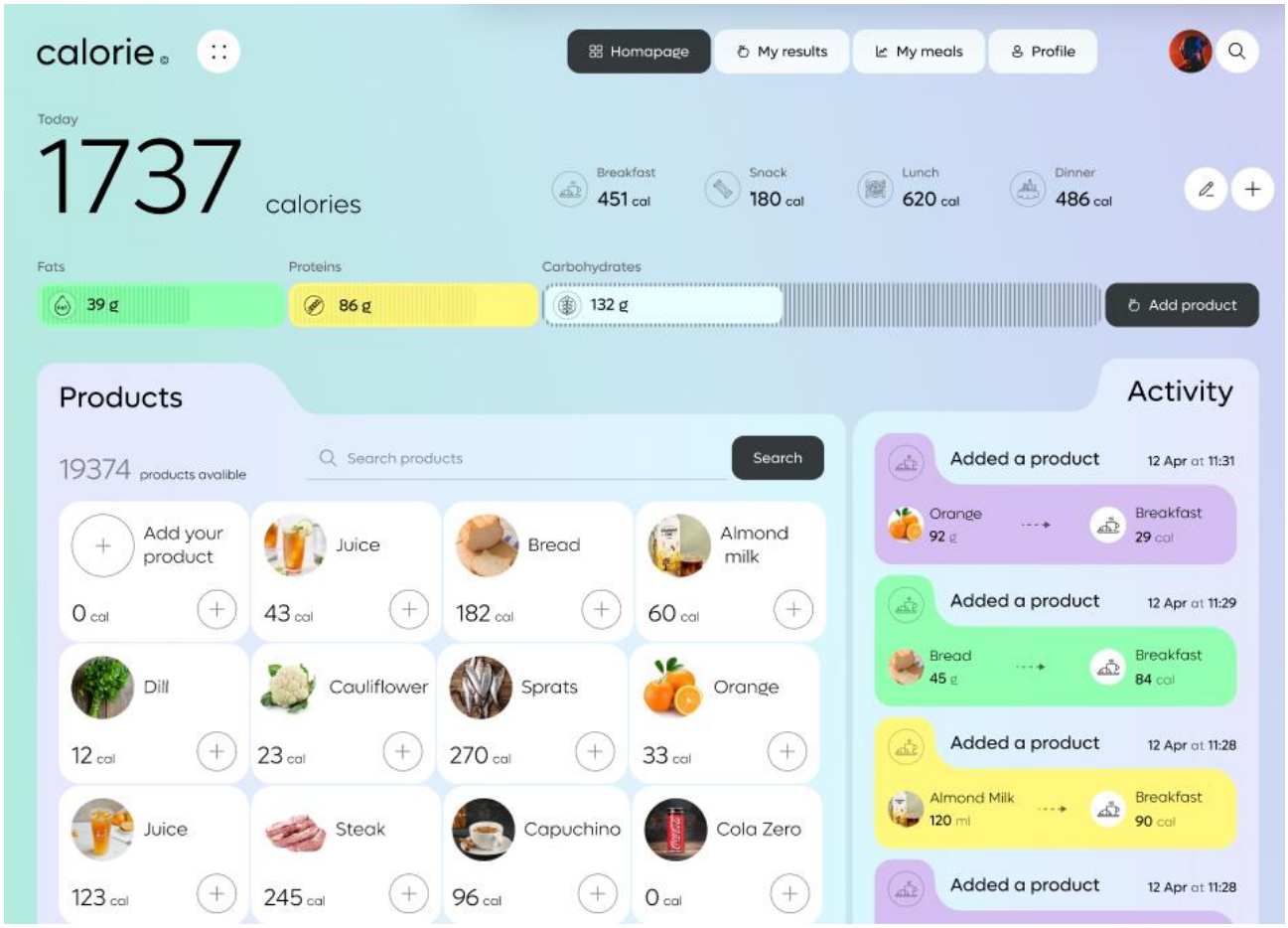


Рисунок 4.7 – Головна сторінка додатка

Окрім пошуку існуючих продуктів, користувач має можливість створити власний продукт описавши його властивості самостійно. Це може бути корисно для складних страв калорійність яких залежить від рецепту та приготування і може бути прорахована лише власноруч.

Після створення продукту, він відображається так само, як і інші продукти у вікні пошуку. Усі дані про індивідуальні продукти зберігаються окремо для кожного користувача, тому вони не будуть відображатися при пошуку з іншого аккаунту.

Додамо новий продукт до системи. Для цього, натиснемо «Add your product». Після цього, відкриється нове діалогове вікно для додавання інформації про продукт (рисунок 4.8).

В цьому вікні, завантажимо фото продукту, назву та кількість калорій.

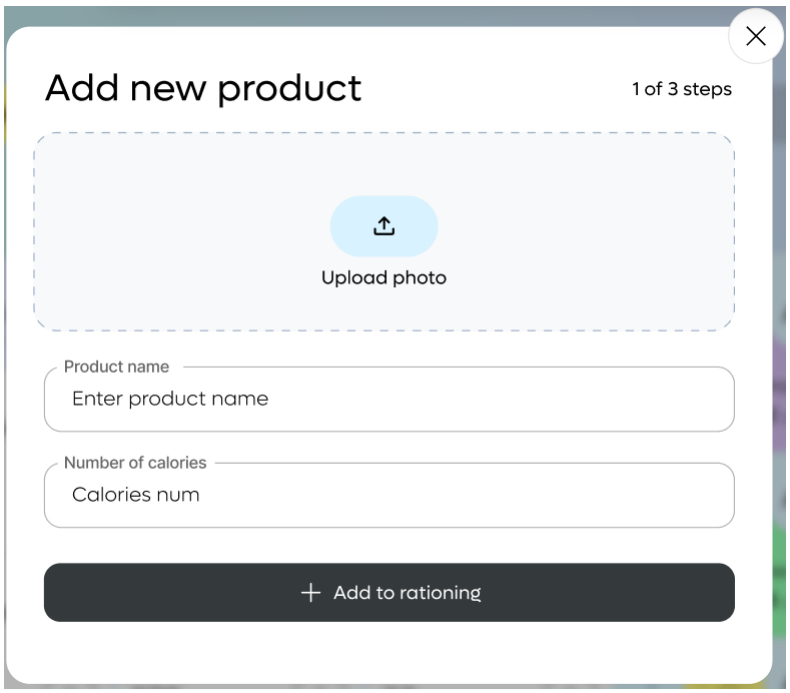


Рисунок 4.8 – Вікно додавання нового продукту (крок 1)

Після цього, введемо деталі про цей продукт: кількість білків, жирів, вуглеводів та цукру (рисунок 4.9).

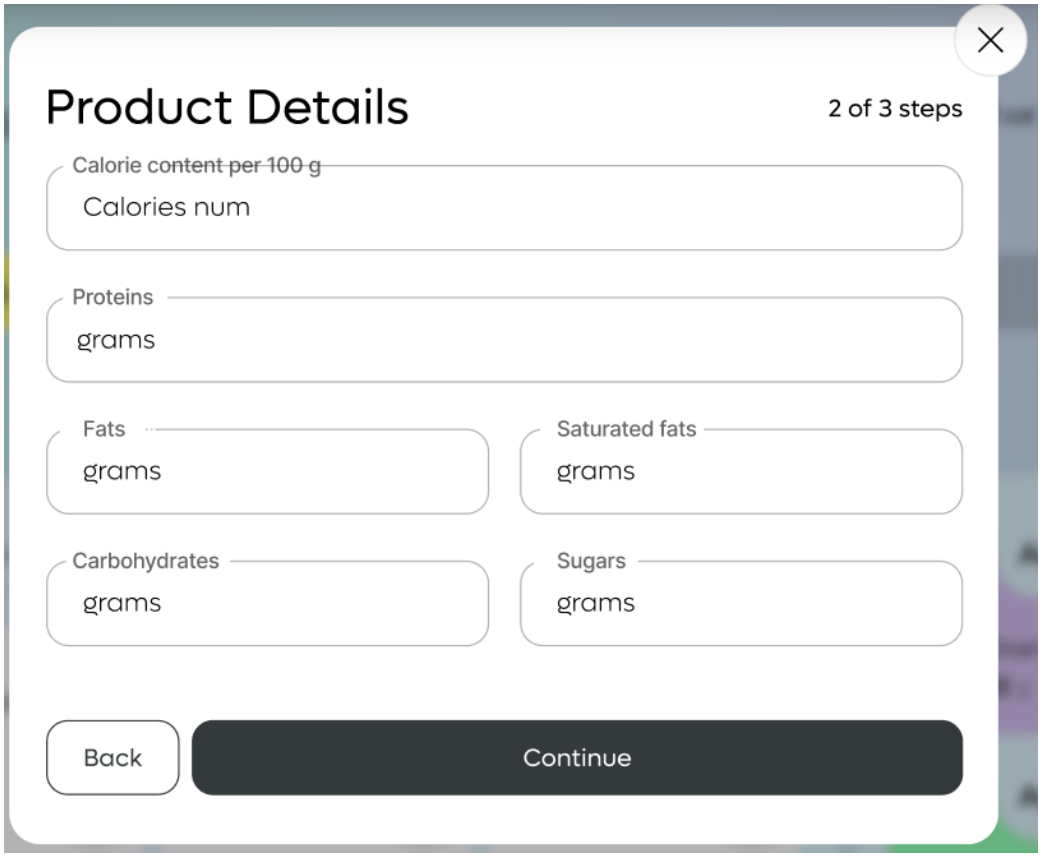


Рисунок 4.9 - Вікно додавання нового продукту (крок 2)

Кінцевим етапом додавання нового продукту є додавання його до свого раціону харчування – дата та тип страви (рисунок 4.10).

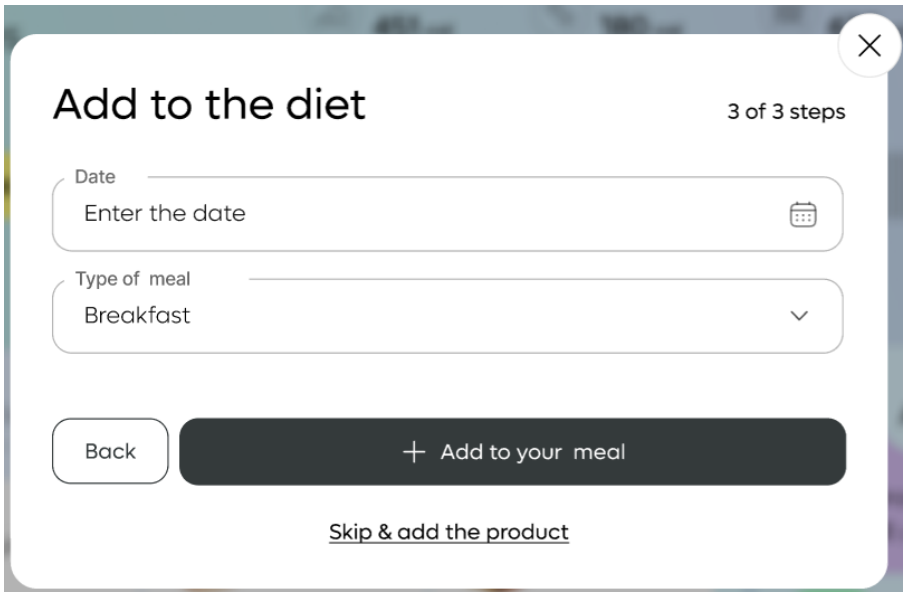


Рисунок 4.10 - Вікно додавання нового продукту (крок 3)

Доданий продукт зображено на рисунку 4.11.

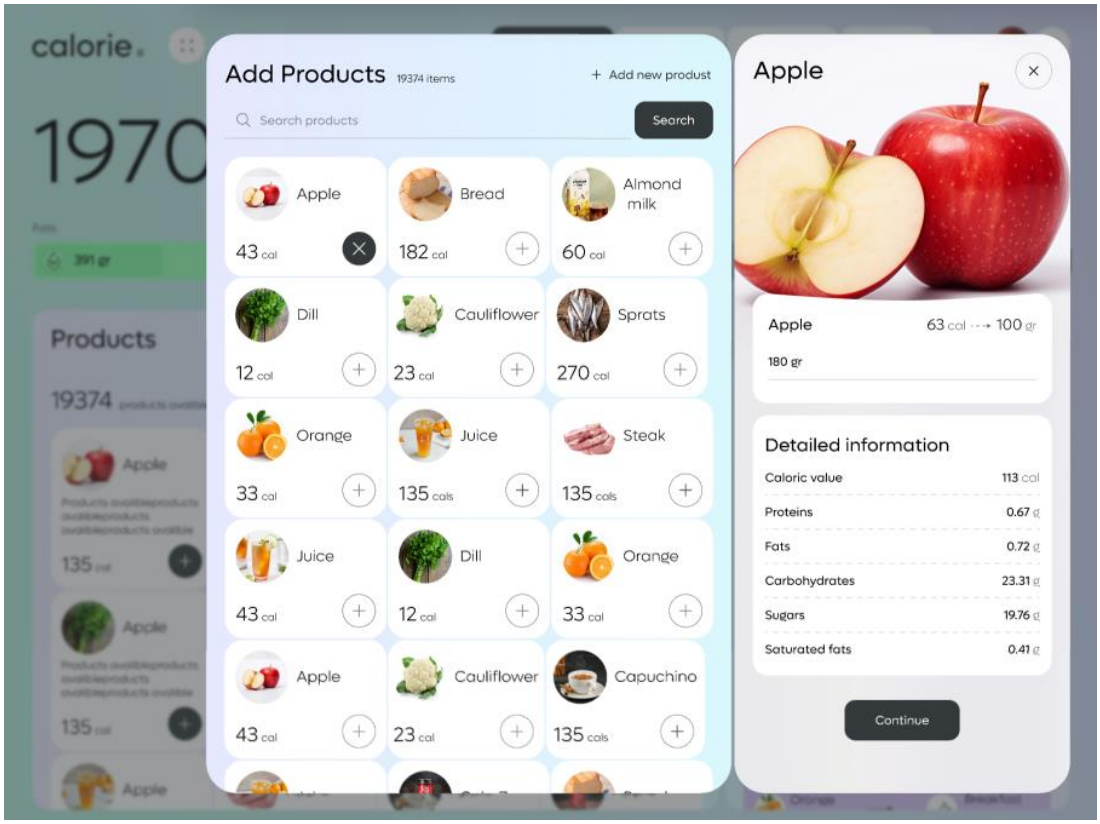


Рисунок 4.11 – Доданий продукт

При додаванні продуктів, ми можемо обрати варіанти їх сортування залежно від того, які продукти більш необхідні для збалансування раціону (рисунок 4.12).

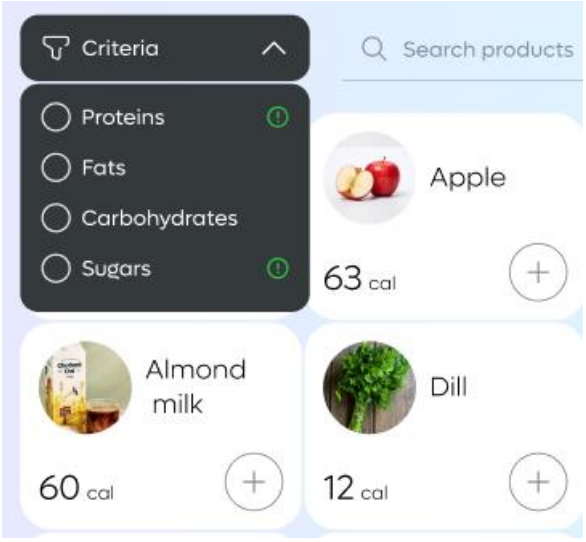


Рисунок 4.12 – Вибір параметрів сортування

Якщо у раціоні харчування спостерігається відхилення вмісту деяких речовин від встановленої користувачем цілі по вазі та норму їх споживання (протеїнів, жирів, карбогідратів або цукрів), система виводить поради щодо вибору певного виду сортування для того, аби запропонувати продукти, які допоможуть нормалізувати раціон харчування (рисунок 4.13).

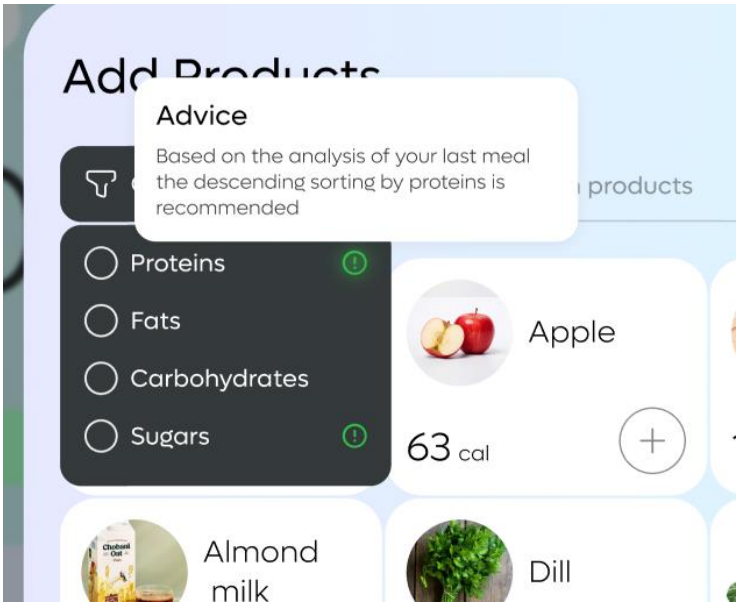


Рисунок 4.13 – Поради щодо застосування методів сортування

На рисунку 4.14 продемонстровано застосування сортування продуктів харчування за зростанням вмісту протеїнів, тобто, найпершим буде демонструватися продукт з найменшим вмістом протеїнів.

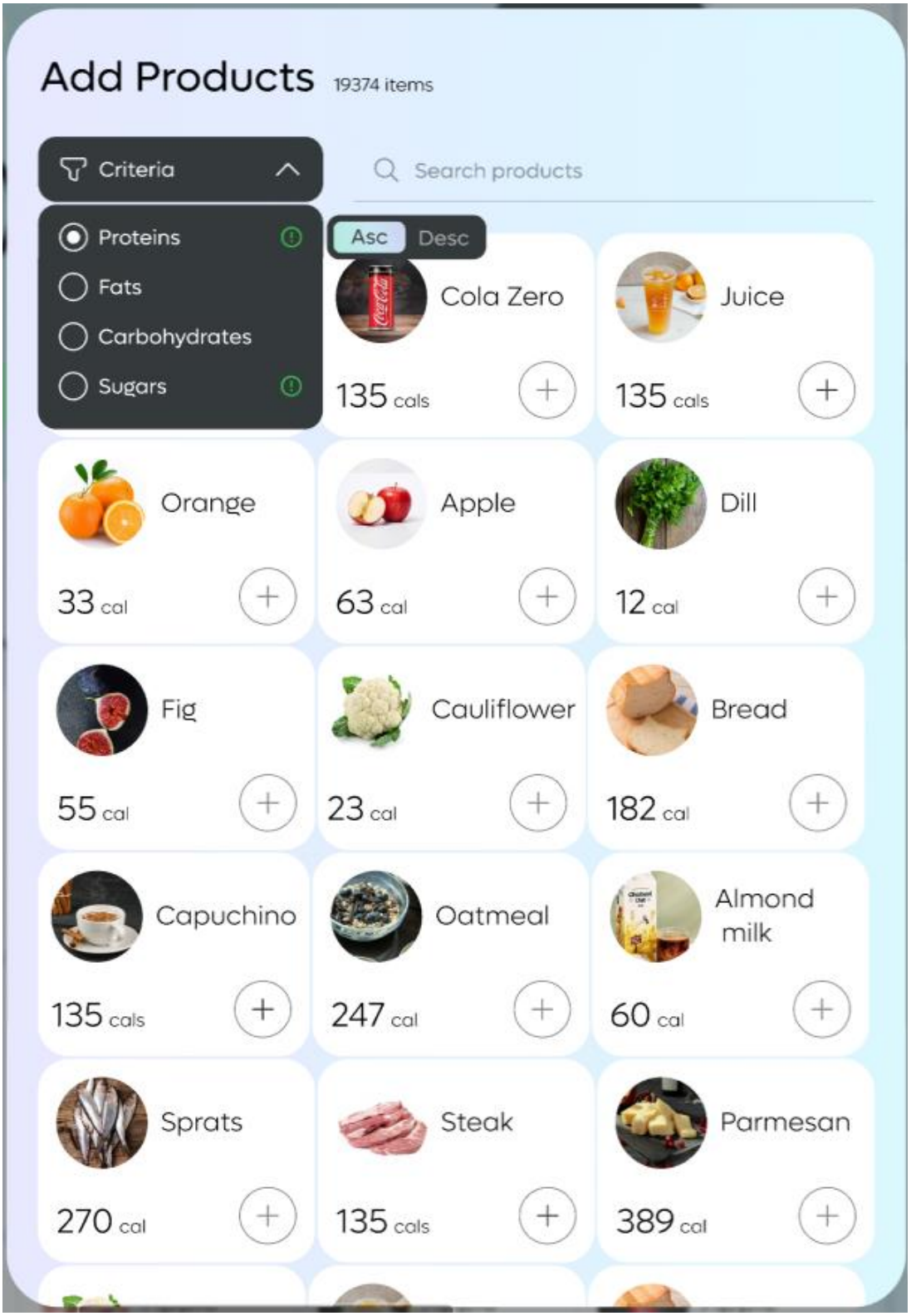


Рисунок 4.14 – Сортування за зростанням вмісту протеїнів

На рисунку 4.15 продемонстровано сортування продуктів харчування за спаданням вмісту протеїнів. Тобто, першими будуть демонструватися продукти з найвищим вмістом протеїнів.

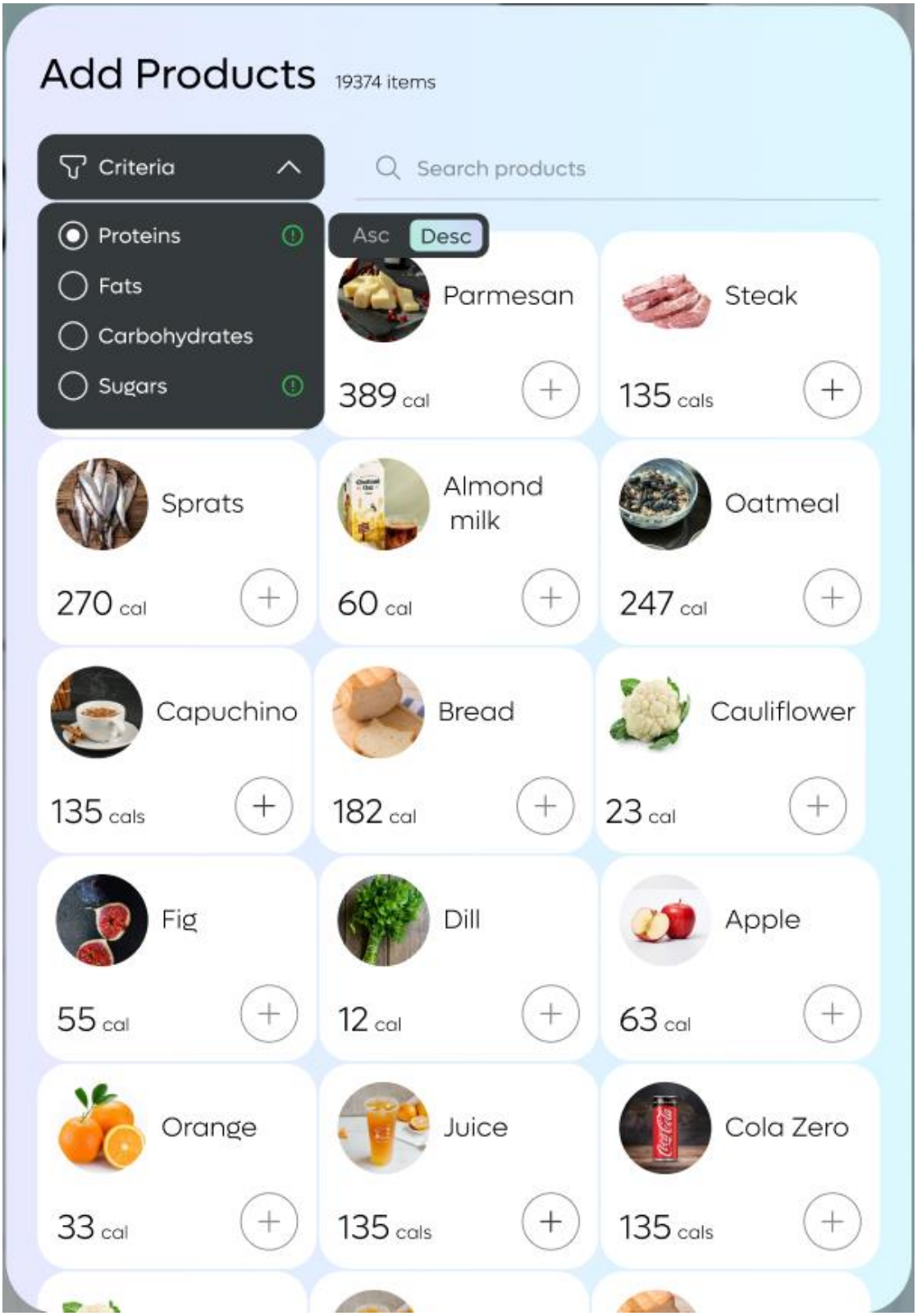


Рисунок 4.15 – Сортування за спаданням вмісту протеїнів

Відкриємо вкладку «Мої страви» (рисунок 4.16).

Дана вкладка відображає раціон користувача за поточний день. Користувач може обрати дату скористувавшись календарем у правому верхньому кутку екрану або натиснувши на дати поблизу обраної дати.

Якщо дані раціону відсутні, усі значення будуть порожніми та аналіз даних за цей день буде недоступним.

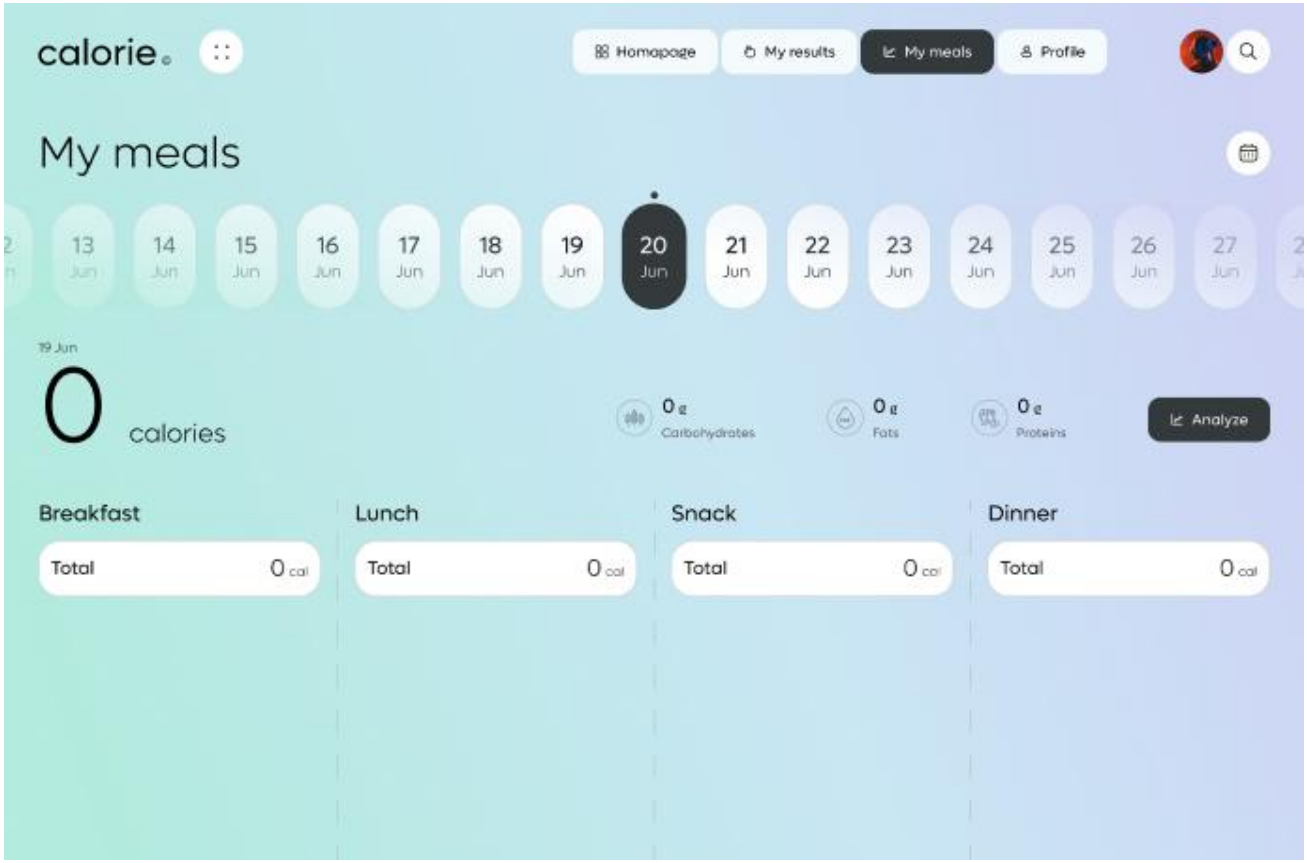


Рисунок 4.16 – Вкладка «Мої страви»

Додамо продукти у раціон харчування на обраний день. Результат зображено на рисунку 4.17.

Раціон розділено на чотири стандартних типи прийому їжі, калорійність яких розраховується окремо. Її можна побачити під назвою прийому їжі у полі “Total”. Сума цих значень утворює загальну кількість спожитих за день калорій та відображається на рівні з нутрієнтами вжитими за день. Дані по кількості нутрієнтів відображаються у грамах та рахуються за увесь денний раціон разом

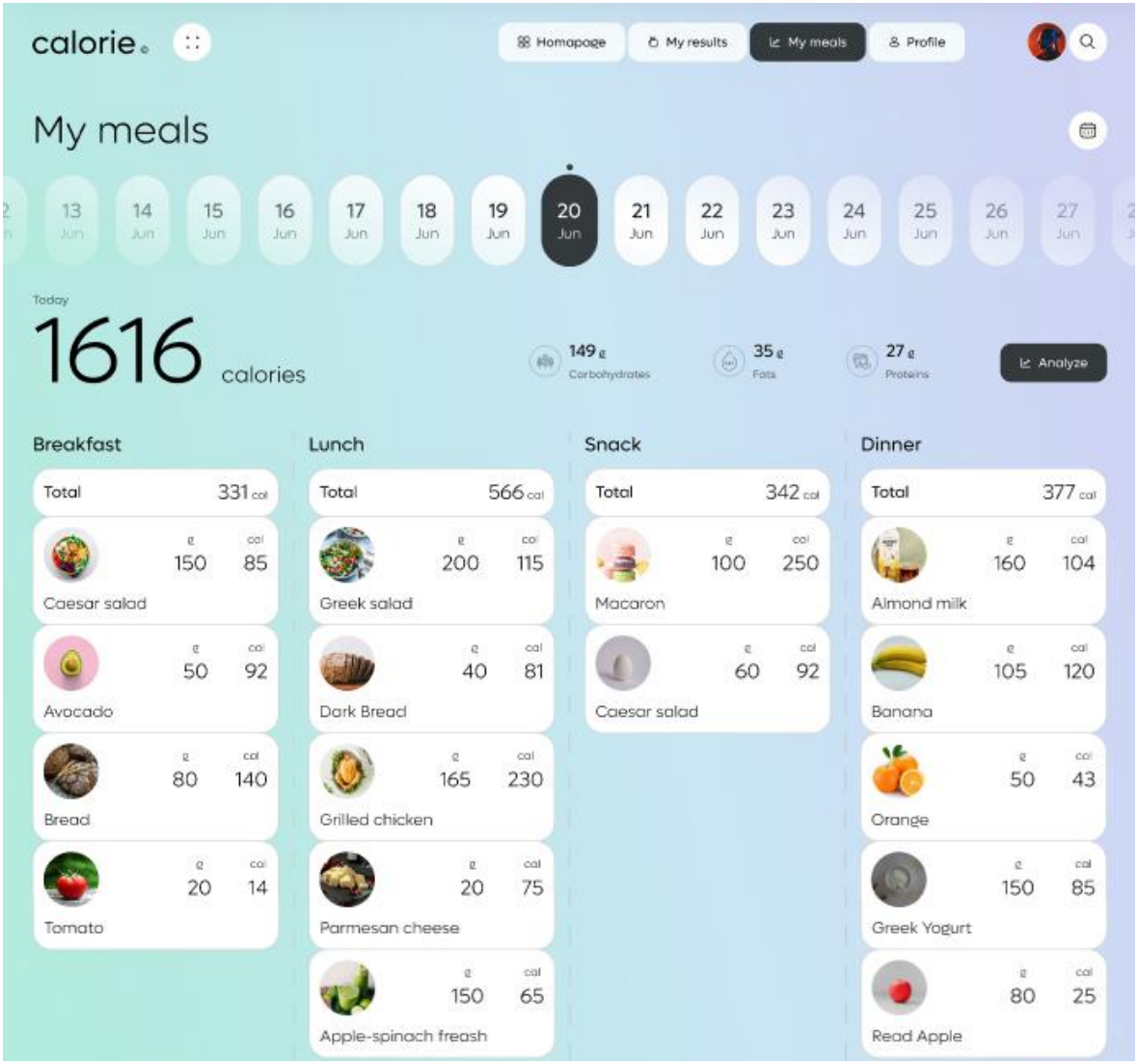


Рисунок 4.17 – Заповнений раціон харчування

Натиснемо кнопку «Analyze» для аналізу раціону харчування. Результат аналізу наведено на рисунку 4.18.

Результат аналізу розділено на дві частини: фактичні та рекомендовані дані раціону за день.

Зліва відобраються калорії та нутрієнти спожиті користувачем, що розраховані на основі даних про продукти в раціоні. Зправа відображаться нормальні значення відповідних показників, які вирахувані на основі введених користувачем даних у профілі. Це значення з якими користувач може порівнювати свій раціон та корегувати для покращення якості харчування.

Справа від розділу з аналізом раціону харчування виводяться поради від штучного інтелекту, які базуються на даних користувача, таких як стать, вік, вага та

цілі щодо зміни ваги. Штучний інтелект може запропонувати змінити вміст певних речовин в раціоні харчування користувача.

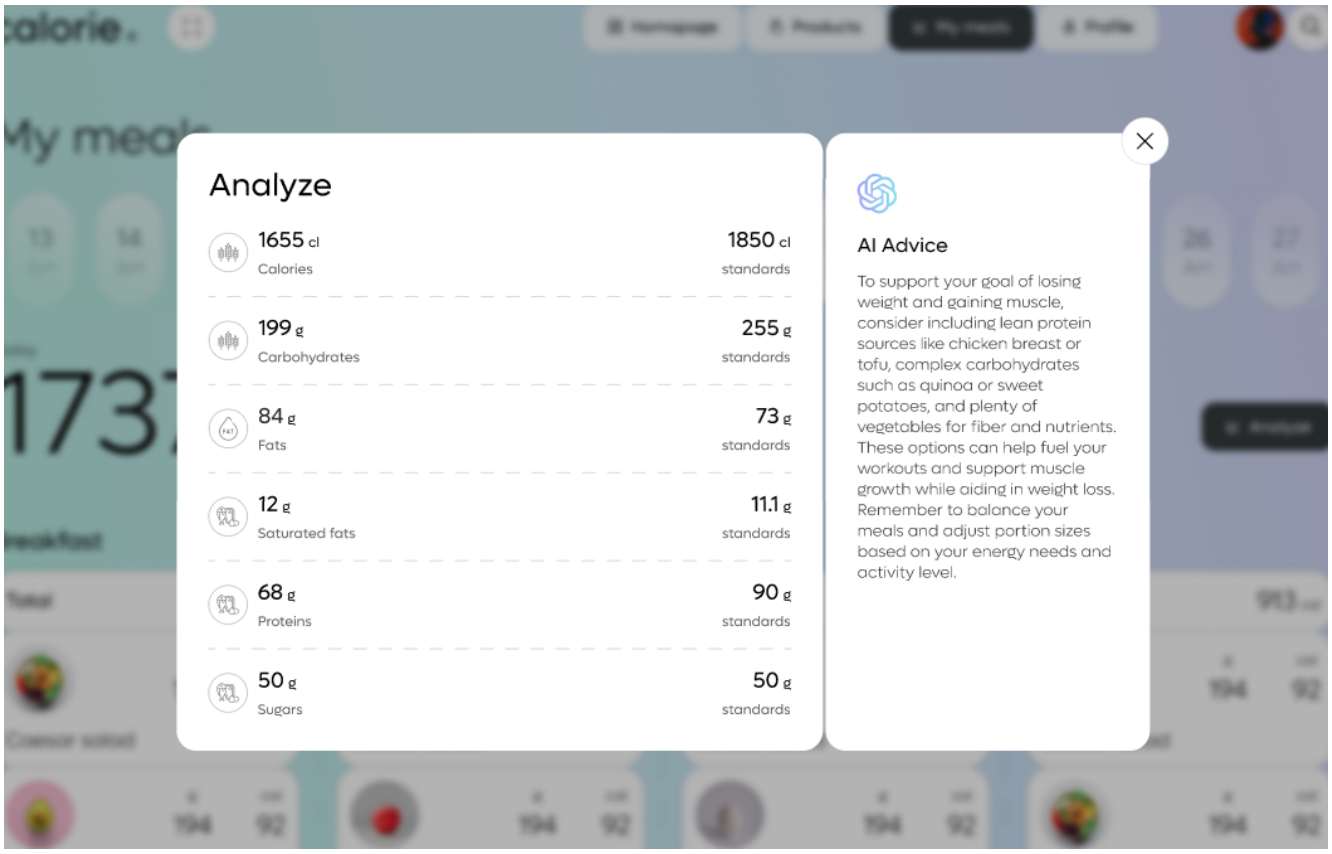


Рисунок 4.18 – Аналіз раціону харчування

Відкриємо вкладку з відомостями про користувача (рисунок 4.19).

На цій сторінці відображена персональна інформація користувача та параметри його тіла.

Обов'язкові поля позначені зірочкою та мають бути заповнені при реєстрації. У профілі відображаються актуальні дані про фізичні показники користувача які він може редагувати та відстежувати.

Поле «Your goal» - ціль користувача пов’язана з його показниками ваги. Користувач може обрати між трьома варіантами: підтримання, зниження, або набір ваги, або не обрати жодного. Цей показник впливає на кількість калорій та поживних речовий, яку рекомендовано вживати користувачеві, вона генерується при аналізі профілю.

Якщо користувач не вказав свою ціль, рекомендована калорійність його

раціону калькулюється в залежності від індексу маси тіла, який в свою чергу вираховується на основі обов’язкових полів.

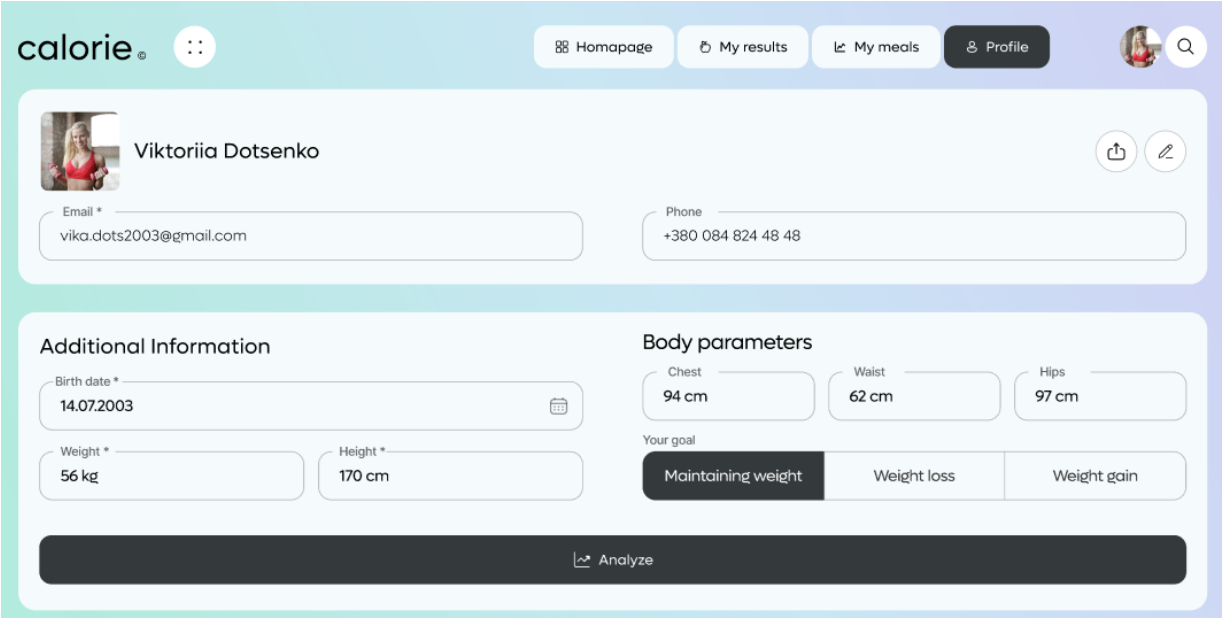


Рисунок 4.19 – Аккаунт користувача

Проведемо аналіз показників тіла на основі цих параметрів. Результат аналізу наведено на рисунку 4.20.

Після розрахунку індексів маси тіла та розподілу жиру по тілуб результативні дані порівнюють із нормою. Якщо показник нижче або вижче за норму, його значення виділяється червоним та з'являється позначка, що відображає напрямок відхилення показника. Інші параметри є рекомендованими для користувача, він може опиратись на них при побудові раціону харчування.

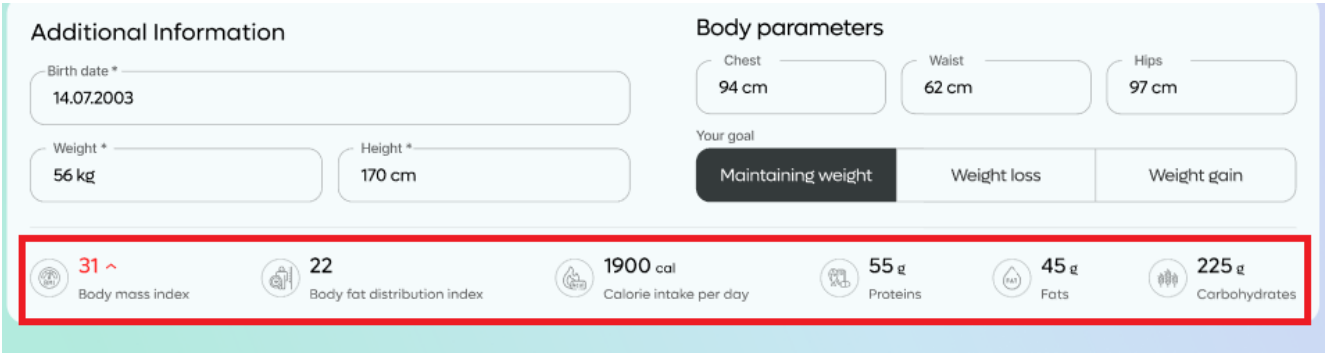


Рисунок 4.20 – Розрахунок показників для користувача

Таким чином, було продемонстровано, що застосунок виконує поставлені перед ним на початку розробки задачі та працює без помилок.

4.3 Висновки

В цьому розділі, було проведено огляд методів тестування програмного забезпечення. Було проведено тестування програмного застосунку для аналізу та визначення раціону харчування людини. Було наведено приклади її використання, тестування продемонструвало працездатність та коректну роботу програми.

ВИСНОВКИ

В результаті виконання бакалаврської дипломної роботи, було реалізовано застосунок для визначення та аналізу раціону харчування людини. Він забезпечує процедури реєстрації користувача, записи його антропометричних параметрів, розрахунок показників тіла, запис раціону харчування, пошук інформації про нутріативні цінності продуктів, складання раціону харчування залежно від потреб користувача та ведення історії змін показників тіла людини.

Для розробки було використано фрейморки Spring та React, мови програмування Java та JavaScript, систему управління базами даних PostgreSQL, інструмент для обміну даних між міксосервісами Kafka. Бакалаврська дипломна робота оформлена згідно методичних вказівок [21].

У першому розділі, було проведено аналіз стану питання розробки застосунку для визначення та аналізу раціону харчування людини, порівняльний аналіз аналогів, аналіз методів розв’язання задачі. На їх основі було доведено доцільність та актуальність власної розробки, поставлено задачі дослідження.

У другому розділі, було розроблено модель застосунку, метод отримання інформації про поживні цінності продуктів харчування, загальний алгоритм роботи програми та діаграми станів.

У третьому розділі, було проведено варіантний аналіз засобів розробки, розроблено алгоритми роботи застосунку, розроблено структурні схеми інтерфейсу застосунку, розроблено програмний застосунок.

У четвертому розділі, було проведено аналіз методів тестування, проведено тестування розробленого застосунка. Тестування довело її працездатність та коректність роботи. Розроблений застосунок виконує поставлені перед ним на початку розробки задачі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Принципи раціонального здорового харчування [Електронний ресурс] – Режим доступу: https://www.bsmu.edu.ua/blog/3243-printsipi-ratsionalnogo-zdorovogo-harchuvannya/ – Дата доступу: 04.05.2024.

2. Доценко В.С. Exploring Of Java HTTP Client Implementations// Матеріали конференції «Стан, досягнення та перспективи інформаційних систем і технологій» (2024), Одеса, Видавництво ОНТУ, 2024 р. – 498 с.

3. MyFitnessPal [Електронний ресурс] – Режим доступу: https://www.myfitnesspal.com/ – Дата доступу: 18.04.2024.

4. Lose It! [Електронний ресурс] – Режим доступу: https://www.loseit.com/ (дата звернення: 18.04.2024) – Дата доступу: 18.04.2024.

5. Fooducate [Електронний ресурс] – Режим доступу: https://www.fooducate.com/ – Дата доступу: 18.04.2024.

6. Node.js [Електронний ресурс] – Режим доступу: https://nodejs.org/en – Дата доступу: 21.04.2024.

7. Django [Електронний ресурс] – Режим доступу: https://www.djangoproject.com/ – Дата доступу: 21.04.2024.

8. Spring [Електронний ресурс] – Режим доступу: (дата звернення: 21.04.2024)

9. PostgreSQL [Електронний ресурс] – Режим доступу: https://www.postgresql.org/ – Дата доступу: 25.04.2024.

10. MySQL [Електронний ресурс] – Режим доступу: https://www.mysql.com/ – Дата доступу: 25.04.2024.

11. MongoDB [Електронний ресурс] – Режим доступу: https://www.mongodb.com/ – Дата доступу: 25.04.2024.

12. Kafka [Електронний ресурс] – Режим доступу: https://kafka.apache.org/ – Дата доступу: 26.04.2024.

13. IntelliJ IDEA [Електронний ресурс] – Режим доступу: https://www.jetbrains.com/idea/ – Дата доступу: 26.04.2024.

14. Eclipse [Електронний ресурс] – Режим доступу: https://www.eclipse.org/downloads/ – Дата доступу: 26.04.2024.

15. Netbeans [Електронний ресурс] – Режим доступу: https://netbeans.apache.org/front/main/index.html – Дата доступу: 26.04.2024.

16. Atom [Електронний ресурс] – Режим доступу: https://editor.cc/ – Дата доступу: 28.04.2024.

17. Visual Studio Code [Електронний ресурс] – Режим доступу: https://code.visualstudio.com/ – Дата доступу: 28.04.2024.

18. Webstorm [Електронний ресурс] – Режим доступу: https://www.jetbrains.com/webstorm/ – Дата доступу: 28.04.2024.

19. What is functional testing? [Електронний ресурс] – Режим доступу: https://www.opentext.com/what-is/functional-testing – Дата доступу: 14.05.2024.

20. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів спеціальності 121 "Інженерія програмного забезпечення" / Уклад. О. Н. Романюк, Г. О. Черноволик – Вінниця: ВНТУ, 2022. – 52с.

ДОДАТКИ

ДОДАТОК А

Технічне завдання

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

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О.Н.
«_12_»__березня__2024 р.

Технічне завдання
на бакалаврську дипломну роботу «Розробка програмного застосунку для визначення та аналізу раціону харчування людини»
за спеціальністю
121 – Інженерія програмного забезпечення

Керівник бакалаврської дипломної роботи:
_____ к.т.н., доц. О.М. Хошаба
«_12_»__березня__2024 року.

Виконав:
_____ студент гр. 1ПІ-20б В.С. Доценко
«_12_»__березня__2024 року.

м. Вінниця – 2024 рік

1. Найменування та галузь застосування

Бакалаврська дипломна робота: «Розробка програмного застосунку для визначення та аналізу раціону харчування людини».

Галузь застосування – веб-технології.

2. Підстава для розробки.

Завдання на роботу, яке затверджене на засіданні кафедри програмного забезпечення – протокол №17 від 20 березня 2024 р.

3. Мета та призначення розробки.

Метою роботи є покращення можливостей визначення раціону харчування людини шляхом розробки та використання спеціалізованого програмного застосунку який дозволяє автоматизувати процес аналізу раціону конкретної людини та сформувати рекомендації щодо раціону .

4. Вихідні дані для проведення НДР

1. Принципи раціонального здорового харчування [Електронний ресурс] – Режим доступу до ресурсу: https://www.bsmu.edu.ua/blog/3243-printsipi-ratsionalnogo-zdorovogo-harchuvannya/ – Дата доступу: 24.04.2024.

2. Kafka [Електронний ресурс] – Режим доступу до ресурсу: https://kafka.apache.org/ – Дата доступу: 26.04.2024.

3. PostgreSQL [Електронний ресурс] – Режим доступу до ресурсу: https://www.postgresql.org/ – Дата доступу: 26.04.2024.

4. What is functional testing? [Електронний ресурс] – Режим доступу до ресурсу: https://www.opentext.com/what-is/functional-testing – Дата доступу: 29.04.2024.

5. Технічні вимоги

Вхідні дані — інформація про користувача.

Вихідні дані — раціон харчування, фізичні показники користувача, нутріативні цінності раціону харчування.

6. Конструктивні вимоги.

Інтерфейс програми повинен відповідати естетичним та ергономічним вимогам, повинна бути зручною в обслуговуванні та керуванні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

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

–пояснювальна записка до бакалаврської дипломної роботи;

–технічне завдання;

–лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи
1	Аналіз стану розвитку програм для визначення та аналізу раціону харчування людини	14.03.24 – 24.03.24
2	Розробка структури та алгоритмів програмного застосунку	24.03.24 – 06.04.24
3	Варіантний аналіз та вибір засобів розробки	06.04.24 – 11.04.24
4	Розробка програмного застосунку	12.04.24 – 29.04.24
5	Тестування програмного застосунку	29.04.24 – 11.05.24
6	Оформлення матеріалів до захисту БДР	12.05.24– 30.05.24

10. Порядок контролю та прийняття.

Усі етапи бакалаврської роботи контролюються науковим керівником згідно плану по виконанню роботи. Прийняття бакалаврської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком захисту.

ДОДАТОК Б

Протокол перевірки на наявність текстових запозичень

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Розробка програмного застосунку для визначення та аналізу раціону харчування людини.

Тип роботи: БДР

Підрозділ : кафедра програмного забезпечення, ФІТКІ

Науковий керівник: к.т.н., доц. каф. ПЗ Хошаба О.М.

Оригінальність	96.6%
Схожість	3.4%

Аналіз звіту подібності

Особа, відповідальна за перевірку _____ Черноволик Г. О.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи _____ Доценко В.С.

Керівник роботи _____ Хошаба О.М.

ДОДАТОК В

Лістинг програми

```
import org.springframework.stereotype.Service;

@Service
public class NutritionService {

    public NutritionalDataDTO calculateNutritionalData(Product product, double weightInGrams) {
        NutritionalDataDTO data = new NutritionalDataDTO();
        data.setWeightInGrams(weightInGrams);
        data.setCalories(calculateNutrient(weightInGrams, product.getCalories()));
        data.setProteins(calculateNutrient(weightInGrams, product.getProteins()));
        data.setFats(calculateNutrient(weightInGrams, product.getFats()));
        data.setCarbs(calculateNutrient(weightInGrams, product.getCarbs()));
        data.setSugars(calculateNutrient(weightInGrams, product.getSugars()));

        return data;
    }

    private double calculateNutrient(double weight, double nutrientPer100g) {
        return (weight / 100) * nutrientPer100g;
    }
}

package com.kenshoo;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
```

```
import java.util.ArrayList;
import java.util.List;
import java.util.Optional;

@Service
public class ProductService {

  private final ProductApiClientService productApiClientService;
  private final UserProductRepository userProductRepository;
  private final LocalProductRepository productRepository;

  @Autowired
  public ProductService(ProductApiClientService productApiClientService,
ProductRepository productRepository, UserProductRepository
userProductRepository) {
    this.productApiClientService = productApiClientService;
    this.productRepository = productRepository;
    this.userProductRepository = userProductRepository;
  }

  public List<ProductDTO> searchProducts(ProductsSearchRequest request) {
    Optional<List<Product>> productListOptional =
userProductRepository.findProductsByUserAndName(request.getUser(),
request.getName());

    List<Product> productList = productListOptional.orElseGet(() -> {
      Optional<List<Product>> productListOptionalByName =
productRepository.findByName(request.getName());
      return productListOptionalByName.orElseGet(() -> {
```

```
            List<Product> productsFromApi =
productApiClientService.getProductList(request);
            if (productsFromApi.isEmpty()) {
               throw new ProductNotFoundException("Products not found for user "
+ request.getUser() + " or name " + request.getName());
            }
            localProductRepository.saveAll(productsFromApi);
            return productsFromApi;
         });
      });

      return convertToDTOList(productList);
   }

   private List<ProductDTO> convertToDTOList(List<Product> productList) {
      List<ProductDTO> productDTOList = new ArrayList<>();
      for (Product product : productList) {
         ProductDTO productDTO = new ProductDTO();
         productDTO.setId(product.getId());
         productDTO.setName(product.getName());
         productDTOList.add(productDTO);
      }
      return productDTOList;
   }
}
public enum Nutrient {
   PROTEINS,
   FATS,
   CARBOHYDRATES,
   SUGARS
```

```
}

public enum SortDirection {
  ASC,
  DESC
}

@Data
public class RecommendedSorting {
  private Nutrient nutrient;
  private SortDirection direction;

  public RecommendedSorting(Nutrient nutrient, SortDirection direction) {
     this.nutrient = nutrient;
     this.direction = direction;
  }
}

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
public class UserDietAnalysisService {

  @Autowired
  private UserDietAnalysisRepository userDietAnalysisRepository;

  @Autowired
  private UserRepository userRepository;
```

```
  public UserDietAnalysis getUserDietAnalysis(Long userId) {
     return userDietAnalysisRepository.findByUserId(userId);
  }

  public Set<RecommendedSorting> needsRecommendation(UserDietAnalysis analysis, User user) {
     double deviationThreshold = 0.05;
     double normalProteins = user.getNormalProteins();
     double normalFats = user.getNormalFats();
     double normalCarbohydrates = user.getNormalCarbohydrates();
     double normalSugars = user.getNormalSugars();

     Set<RecommendedSorting> recommendedSortings = new HashSet<>();

     checkDeviation(normalProteins, analysis.getProteins(), deviationThreshold, Nutrient.PROTEINS)
           .ifPresent(direction -> recommendedSortings.add(new RecommendedSorting(Nutrient.PROTEINS, direction)));

     checkDeviation(normalFats, analysis.getFats(), deviationThreshold, Nutrient.FATS)
           .ifPresent(direction -> recommendedSortings.add(new RecommendedSorting(Nutrient.FATS, direction)));

     checkDeviation(normalCarbohydrates, analysis.getCarbohydrates(), deviationThreshold, Nutrient.CARBOHYDRATES)
           .ifPresent(direction -> recommendedSortings.add(new RecommendedSorting(Nutrient.CARBOHYDRATES, direction)));

     checkDeviation(normalCarbohydrates, analysis.getCarbohydrates(),
```

```
deviationThreshold, Nutrient.SUGARS)
            .ifPresent(direction -> recommendedSortings.add(new
RecommendedSorting(Nutrient.SUGARS, direction)));

    return recommendedSortings;
  }

  private Optional<SortDirection> checkDeviation(double normalValue, double
analysisValue, double deviationThreshold, Nutrient nutrient) {
    if (Math.abs((analysisValue - normalValue) / normalValue) >
deviationThreshold) {
      return Optional.of(analysisValue > normalValue ? SortDirection.DESC :
SortDirection.ASC);
    }
    return Optional.empty();
  }
}

public class AIQuestionGenerator {

  public static String createMessage(User user) {
    String template = "Hi, please suggest to me three products that I should add to
my meal. I am a %s, my age is %d, my height is %d cm and weight is %d kg. My
goal is to %s weight.";
    return String.format(template, user.getGender(), user.getAge(),
user.getHeight(), user.getWeight(), user.getGoal());
  }
}

import org.json.JSONArray;
```

```
import org.json.JSONObject;
import org.springframework.web.reactive.function.client.WebClient;
import
org.springframework.web.reactive.function.client.WebClientResponseException;
import reactor.core.publisher.Mono;

public class ChatGPTClient {

  private final WebClient webClient;

  public ChatGPTClient() {
    this.webClient = WebClient.builder()
        .baseUrl(API_URL)
        .defaultHeader("Content-Type", "application/json")
        .defaultHeader("Authorization", "Bearer " + API_KEY)
        .build();
  }

  public String sendMessage(String message) {
    try {
      String jsonInputString = new JSONObject()
          .put("model", "gpt-3.5-turbo")
          .put("messages", new JSONArray()
              .put(new JSONObject()
                  .put("role", "user")
                  .put("content", message)))
          .toString();

      Mono<String> response = webClient.post()
          .bodyValue(jsonInputString)
```

```
                .retrieve()
                .bodyToMono(String.class);

        String responseBody = response.block();
        JSONObject jsonResponse = new JSONObject(responseBody);
        JSONArray choices = jsonResponse.getJSONArray("choices");
        if (choices.length() > 0) {
            return
choices.getJSONObject(0).getJSONObject("message").getString("content");
        } else return null;
      } catch (Exception e) {
        throw new RuntimeException("Failed to make AI request", e);
      }
   }
}

import javax.persistence.*;

@Entity
@Data
public class MealProduct {
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private Long id;
   private String name;
   private double calories;
   private double grams;
   private double fats;
   private double carbs;
   private double sugars;
```

```
    private double proteins;

    @Enumerated(EnumType.STRING)
    private MealType mealType;
}

public enum MealType {
    BREAKFAST,
    LUNCH,
    SUPPER,
    SNACK
}
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
public class MealReportService {
    private final MealReportRepository mealReportRepository;
    private final ProductService productService;
    private final NutritionService nutritionService;

    @Autowired
    public MealReportService(MealReportRepository mealReportRepository,
ProductService productService, NutritionService nutritionService) {
        this.mealReportRepository = mealReportRepository;
        this.productService = productService;
        this.nutritionService = nutritionService;
    }

    public MealReport getMealReportByUser(User user) {
```

```
        return mealReportRepository.findByUser(user)
                .orElseThrow(() -> new MealReportNotFoundException("MealReport not found for user: " + user));
    }

    public MealReport addProductToMealReport(Long mealReportId, Long productId, double grams, MealType mealType) {
        MealReport mealReport = findMealReportById(mealReportId);
        Product product = findProductById(productId);

        NutritionalDataDTO nutritionalData = nutritionService.calculateNutritionalData(product, grams);

        MealProduct mealProduct = createMealProduct(product.getName(), grams, nutritionalData, mealType);
        mealReport.getMealProducts().add(mealProduct);
        mealReport.setTotalCalories(mealReport.getTotalCalories() + nutritionalData.getCalories());

        return mealReportRepository.save(mealReport);
    }

    private MealReport findMealReportById(Long mealReportId) {
        return mealReportRepository.findById(mealReportId)
                .orElseThrow(() -> new MealReportNotFoundException("MealReport not found: " + mealReportId));
    }

    private Product findProductById(Long productId) {
        return productService.findProductById(productId)
```

```
            .orElseThrow(() -> new ProductNotFoundException("Product not found: " + productId));
    }

    private MealProduct createMealProduct(String productName, double grams, NutritionalDataDTO nutritionalData, MealType mealType) {
        MealProduct mealProduct = new MealProduct();
        mealProduct.setName(productName);
        mealProduct.setCalories(nutritionalData.getCalories());
        mealProduct.setGrams(grams);
        mealProduct.setFats(nutritionalData.getFats());
        mealProduct.setCarbs(nutritionalData.getCarbs());
        mealProduct.setSugars(nutritionalData.getSugars());
        mealProduct.setProteins(nutritionalData.getProteins());
        mealProduct.setMealType(mealType);
        return mealProduct;
    }
}

public class MealReportNotFoundException extends RuntimeException {
    public MealReportNotFoundException(String message) {
        super(message);
    }
}

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

@RestController
```

```
@RequestMapping("/api/meal-reports")
public class MealReportController {
    private final MealReportService mealReportService;

    @Autowired
    public MealReportController(MealReportService mealReportService) {
        this.mealReportService = mealReportService;
    }

    @PostMapping("/{mealReportId}/add-product")
    public ResponseEntity<MealReport> addProductToMealReport(@PathVariable
Long mealReportId,
                                                @RequestParam String productName,
                                                @RequestParam double grams,
                                                @RequestParam MealType mealType) {
        MealReport                          updatedMealReport                    =
mealReportService.addProductToMealReport(mealReportId,          productName,
grams, mealType);
        return ResponseEntity.ok(updatedMealReport);
    }

    @GetMapping("/user/{user}")
    public    ResponseEntity<MealReport>    getMealReportByUser(@PathVariable
String user) {
        MealReport mealReport = mealReportService.getMealReportByUser(user);
        return ResponseEntity.ok(mealReport);
    }
}

import org.apache.kafka.clients.producer.ProducerConfig;
```

```
import org.apache.kafka.common.serialization.StringSerializer;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.kafka.core.DefaultKafkaProducerFactory;
import org.springframework.kafka.core.KafkaTemplate;
import org.springframework.kafka.core.ProducerFactory;
import org.springframework.kafka.support.serializer.JsonSerializer;

import java.util.HashMap;
import java.util.Map;

@Configuration
public class KafkaProducerConfig {

   @Value("${kafka.bootstrap-servers}")
   private String bootstrapServers;

   @Bean
   public Map<String, Object> producerConfigs() {
      Map<String, Object> props = new HashMap<>();
      props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG,
bootstrapServers);
      props.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG,
StringSerializer.class);
      props.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG,
JsonSerializer.class);
      return props;
   }
```

```
    @Bean
    public ProducerFactory<String, ProductDataRequestDTO> producerFactory() {
        return new DefaultKafkaProducerFactory<>(producerConfigs());
    }

    @Bean
    public KafkaTemplate<String, ProductDataRequestDTO> kafkaTemplate() {
        return new KafkaTemplate<>(producerFactory());
    }
}

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.kafka.core.KafkaTemplate;
import org.springframework.stereotype.Service;

@Service
public class KafkaProducerService {

    @Value("${kafka.topic.product-data-request}")
    private String productDataRequestTopic;

    private final KafkaTemplate<String, ProductDataRequestDTO> kafkaTemplate;

    @Autowired
    public                              KafkaProducerService(KafkaTemplate<String,
ProductDataRequestDTO> kafkaTemplate) {
        this.kafkaTemplate = kafkaTemplate;
    }
```

```
    public void sendProductDataRequest(ProductDataRequestDTO
productDataRequestDTO) {
        kafkaTemplate.send(productDataRequestTopic, productDataRequestDTO);
    }
}

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.kafka.annotation.KafkaListener;
import org.springframework.stereotype.Service;

@Service
public class KafkaConsumerService {

    private final ProductSearchApiService productSearchApiService;

    @Autowired
    public KafkaConsumerService(ProductSearchApiService
productSearchApiService) {
        this.productSearchApiService = productSearchApiService;
    }

    @KafkaListener(topics = "${kafka.topic.product-data-request}", groupId =
"${kafka.group-id}")
    public void consume(ProductDataRequestDTO productDataRequestDTO) {
        productSearchApiService.sendProductDataRequest(productDataRequestDT
O);
    }
}
```

ДОДАТОК Г

ІЛЮСТРАТИВНА ЧАСТИНА

Розробка програмного застосунку для визначення та аналізу раціону харчування людини

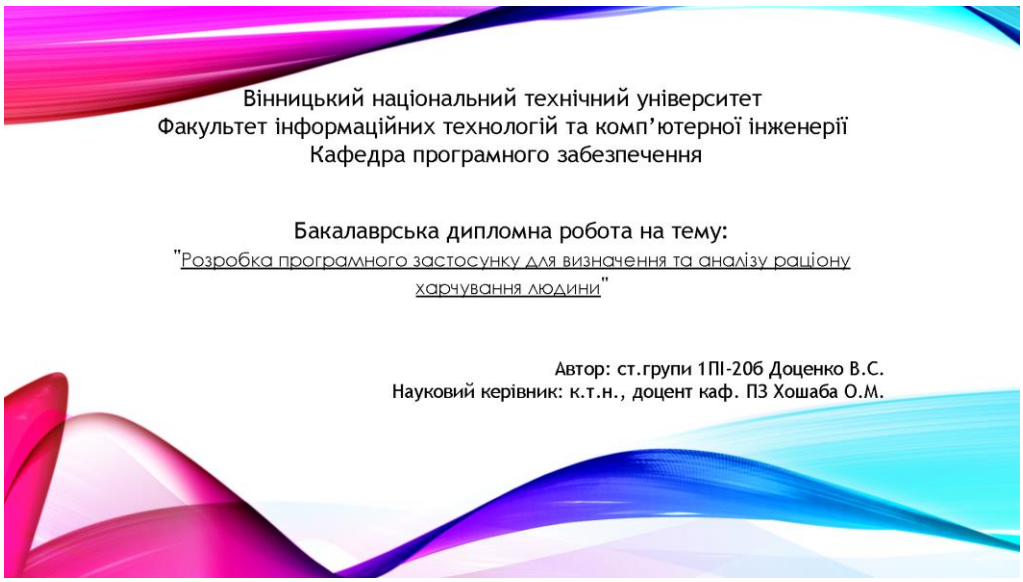


Рисунок Г.1 – Титульний слайд

Рисунок Г.2 – Актуальність теми

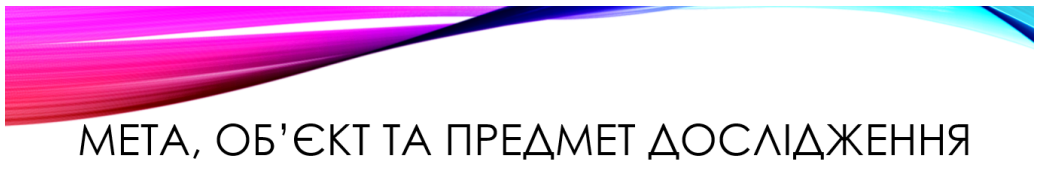
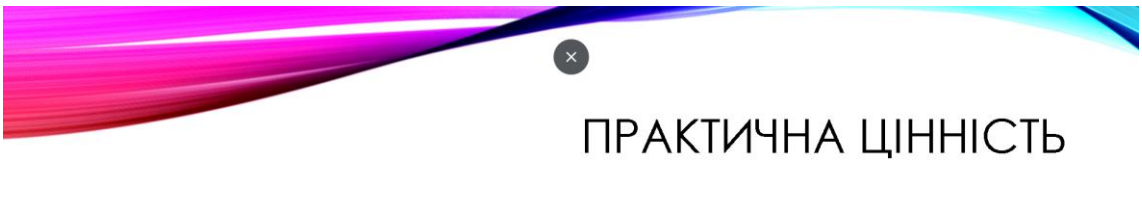


Рисунок Г.3 – Мета, об'єкт та предмет дослідження

Рисунок Г.4 – Завдання дослідження

Рисунок Г.5 – Новизна одержаних результатів



Практична цінність одержаних результатів полягає у можливості використання розробленого додатка для аналізу раціону харчування, перегляду інформації про нутріативні цінності продуктів харчування, ведення історії змін параметрів тіла та формування здорового раціону харчування ,що відповідає потребам користувача.

Рисунок Г.6 – Практична цінність

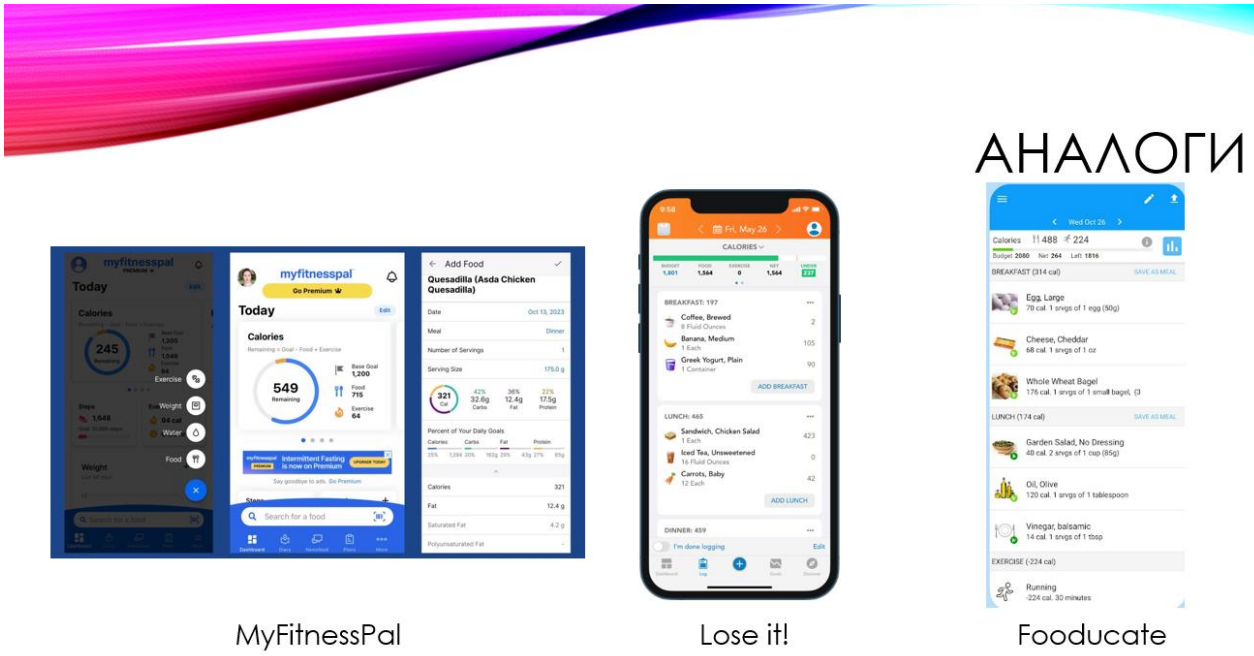


Рисунок Г.7 – Аналоги

ПОРІВНЯЛЬНИЙ АНАЛІЗ АНАЛОГІВ

Характеристика	MyFitnessPal	Lose it!	Fooducate	Власна розробка
Аналіз нутрієнтного складу продуктів	1	1	1	1
Розрахунок параметрів тіла користувача	1	1	0	1
Ведення історії раціону користувача	1	1	1	1
Ведення історії змін ваги користувача	1	0	1	1
Персональні рекомендації	0	0	0	1
Додавання своїх продуктів користувачем	0	0	1	1
Сумарний бал	4	3	4	6

Рисунок Г.8 – Порівняльний аналіз аналогів

ВИКОРИСТАНІ ТЕХНОЛОГІЇ

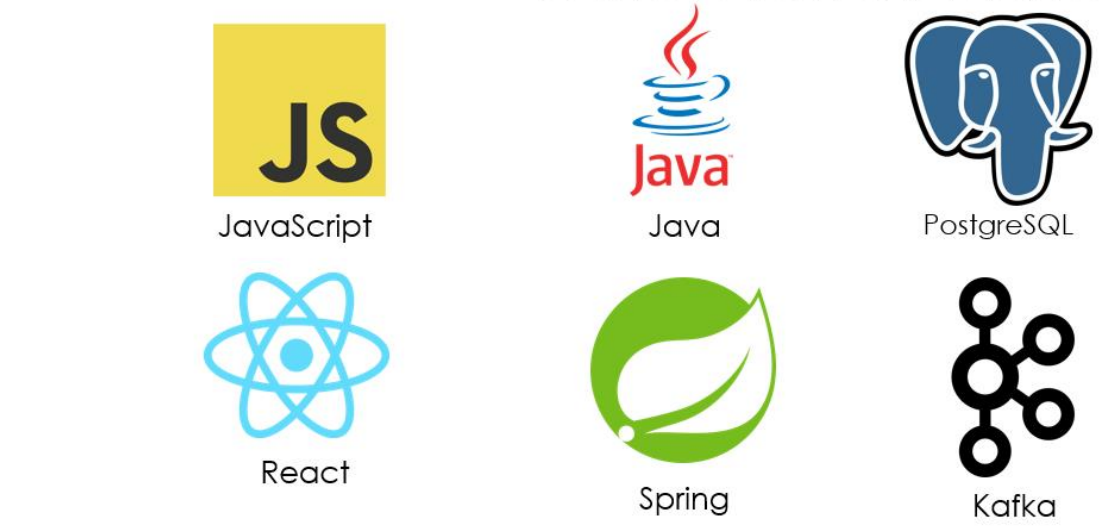


Рисунок Г.9 – Використані технології

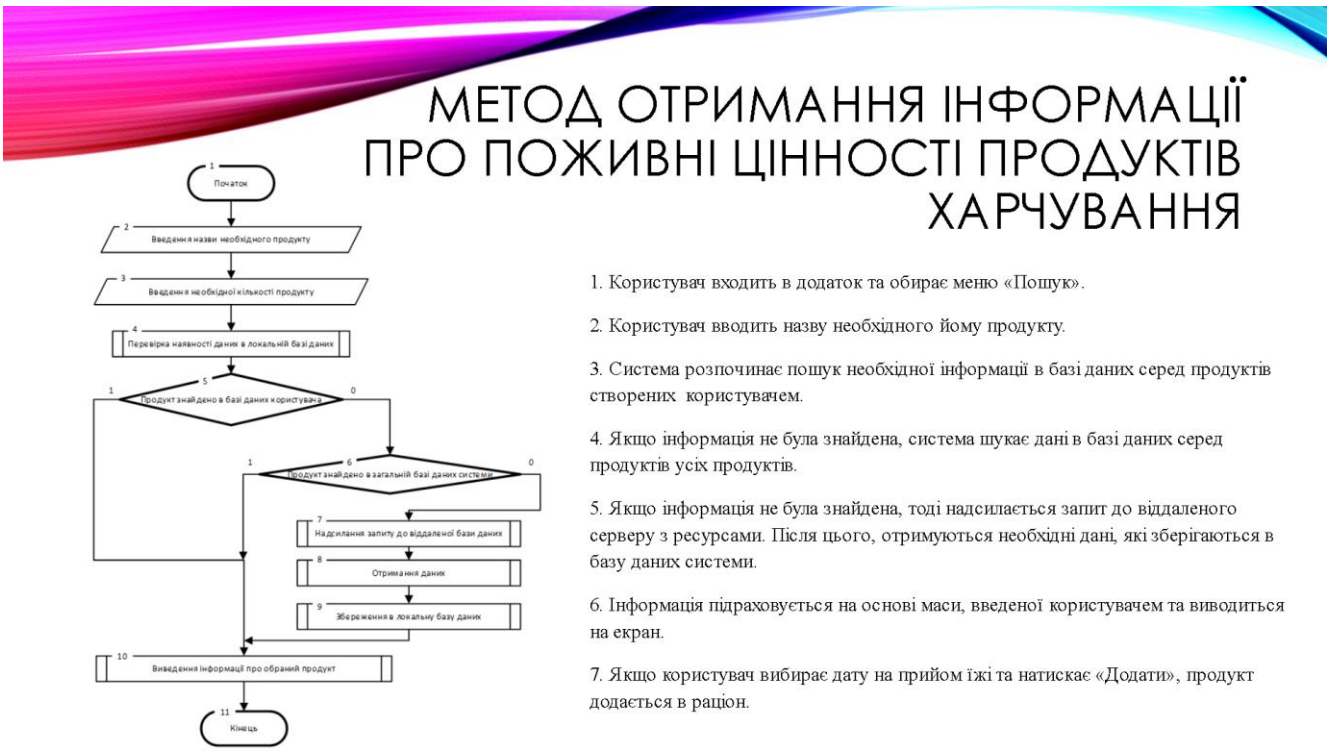


Рисунок Г.10 – Метод отримання інформації про поживні цінності продуктів харчування

Рисунок Г.11 – Метод пошуку продуктів харчування

Рисунок Г.12 – Метод аналізу раціону харчування

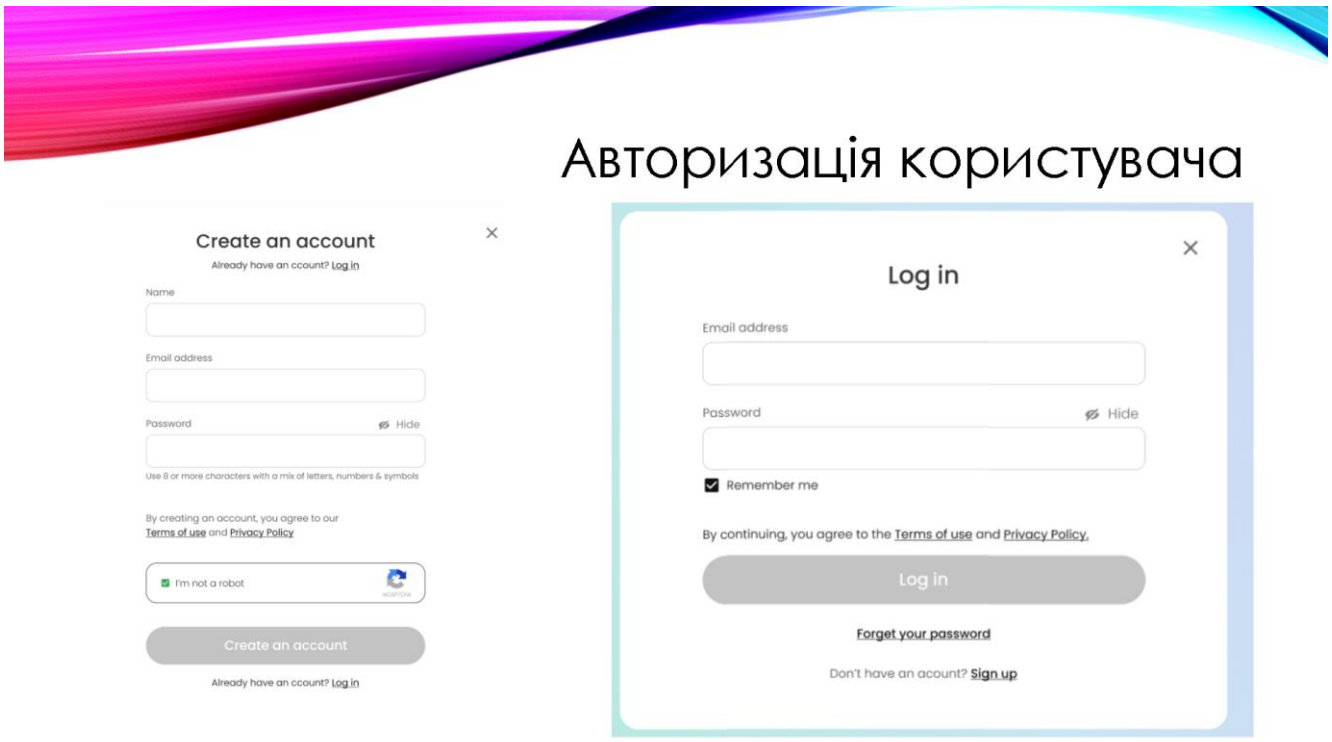


Рисунок Г.13 – Вікна реєстрації та авторизації

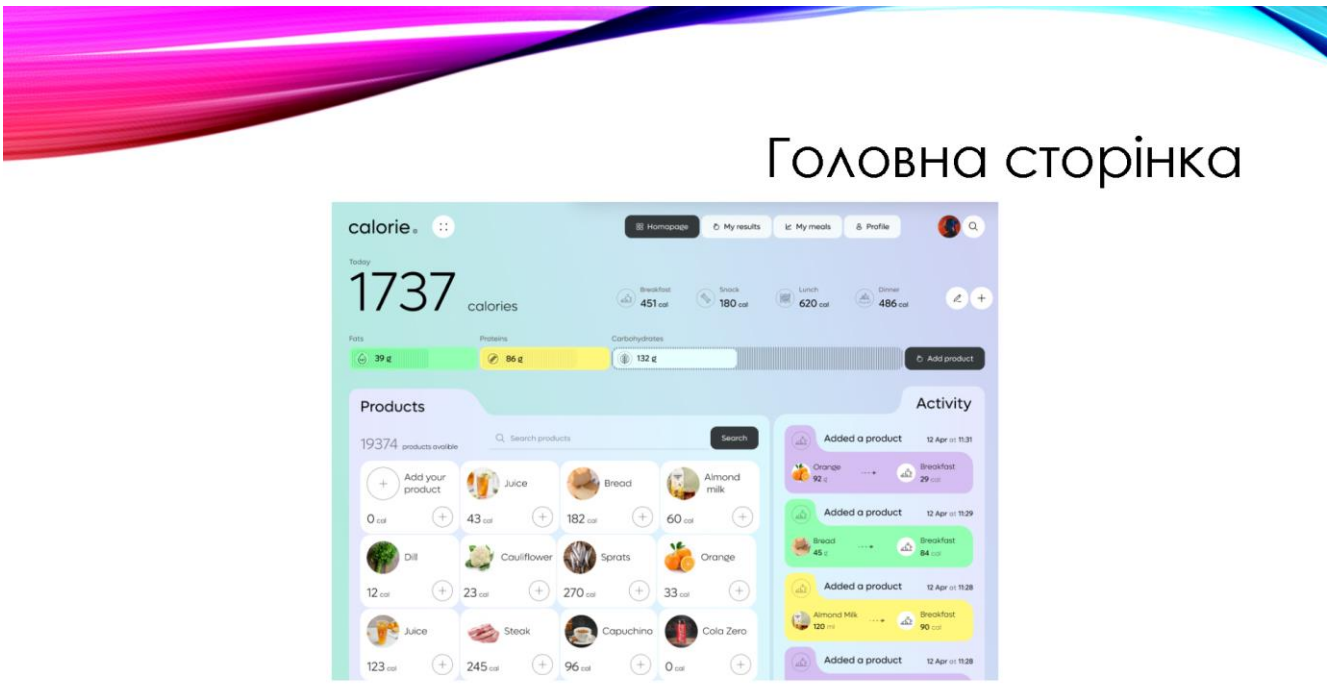


Рисунок Г.14 – Профіль користувача

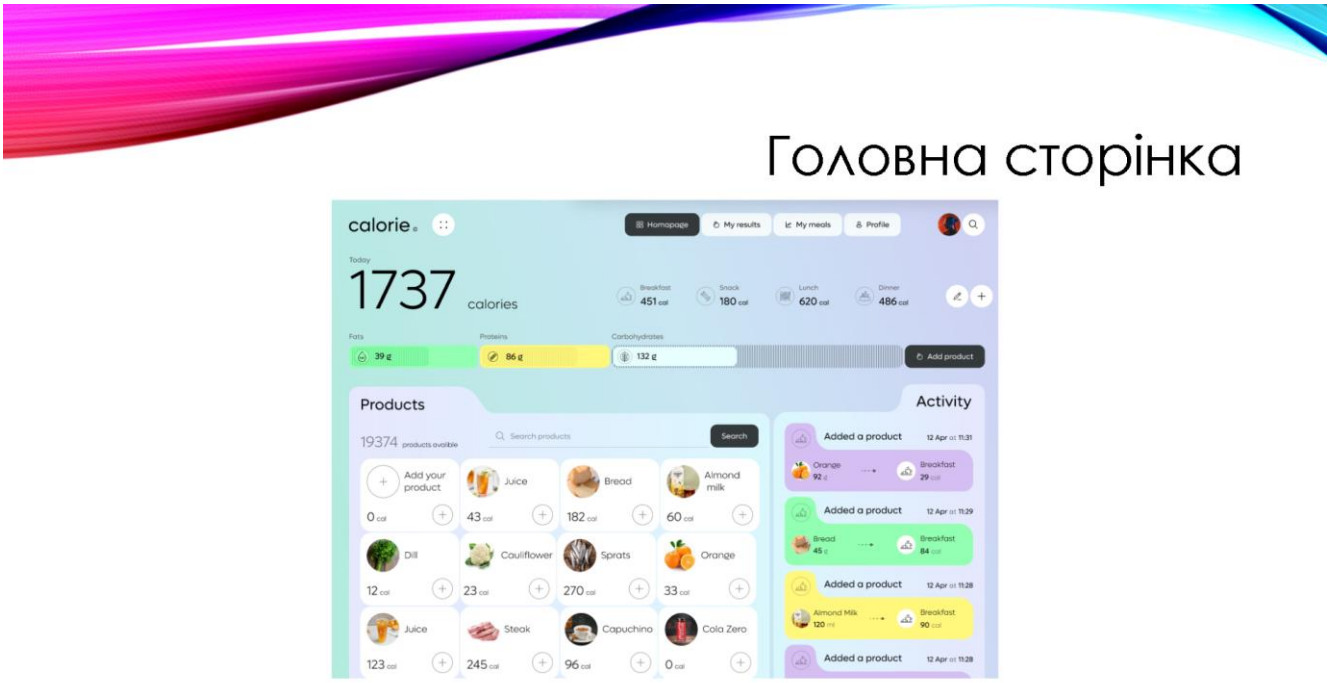


Рисунок Г.15 – Головна сторінка застосунку

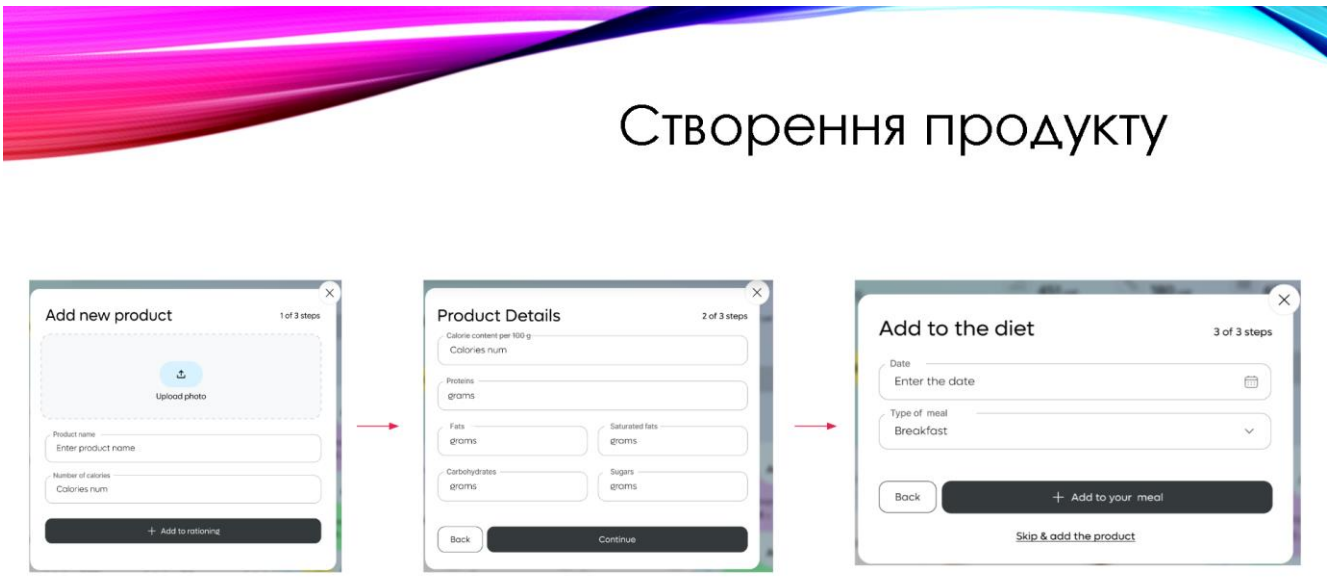


Рисунок Г.16 – Створення продукту

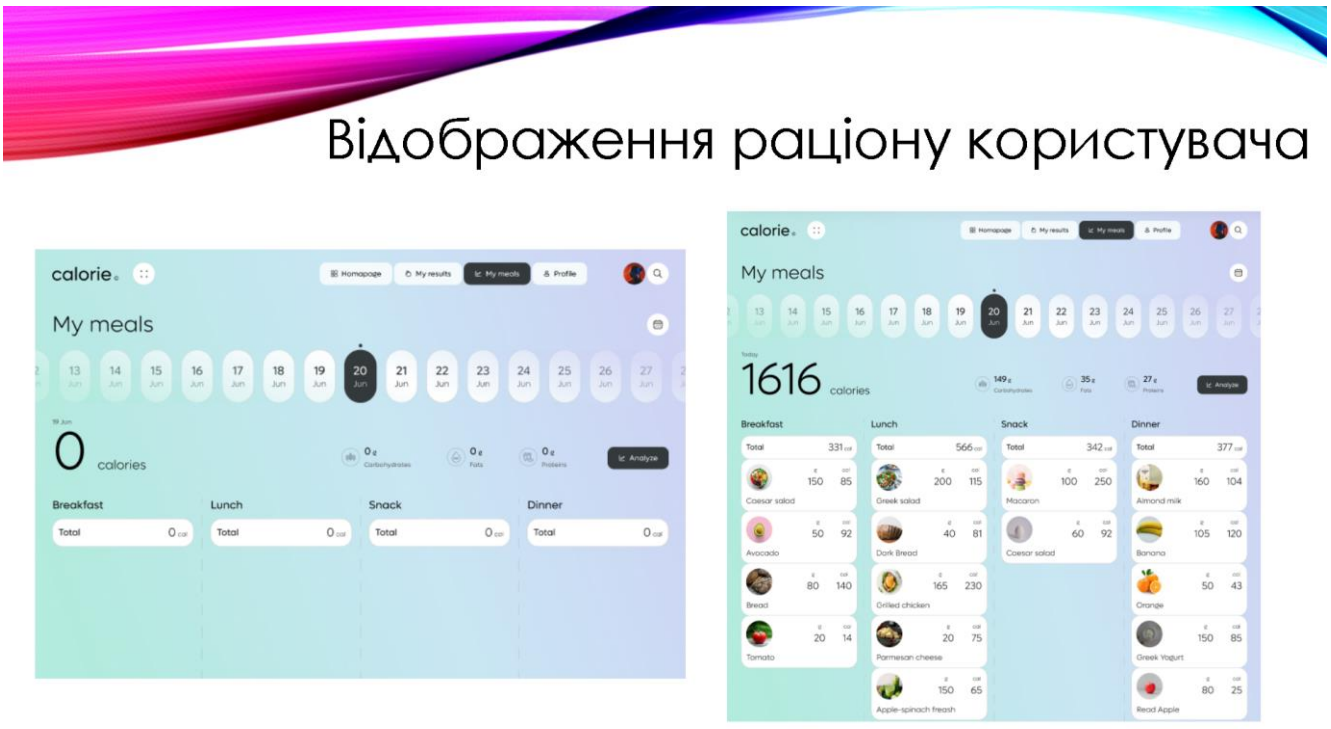


Рисунок Г.17 – Відображення раціону харчування

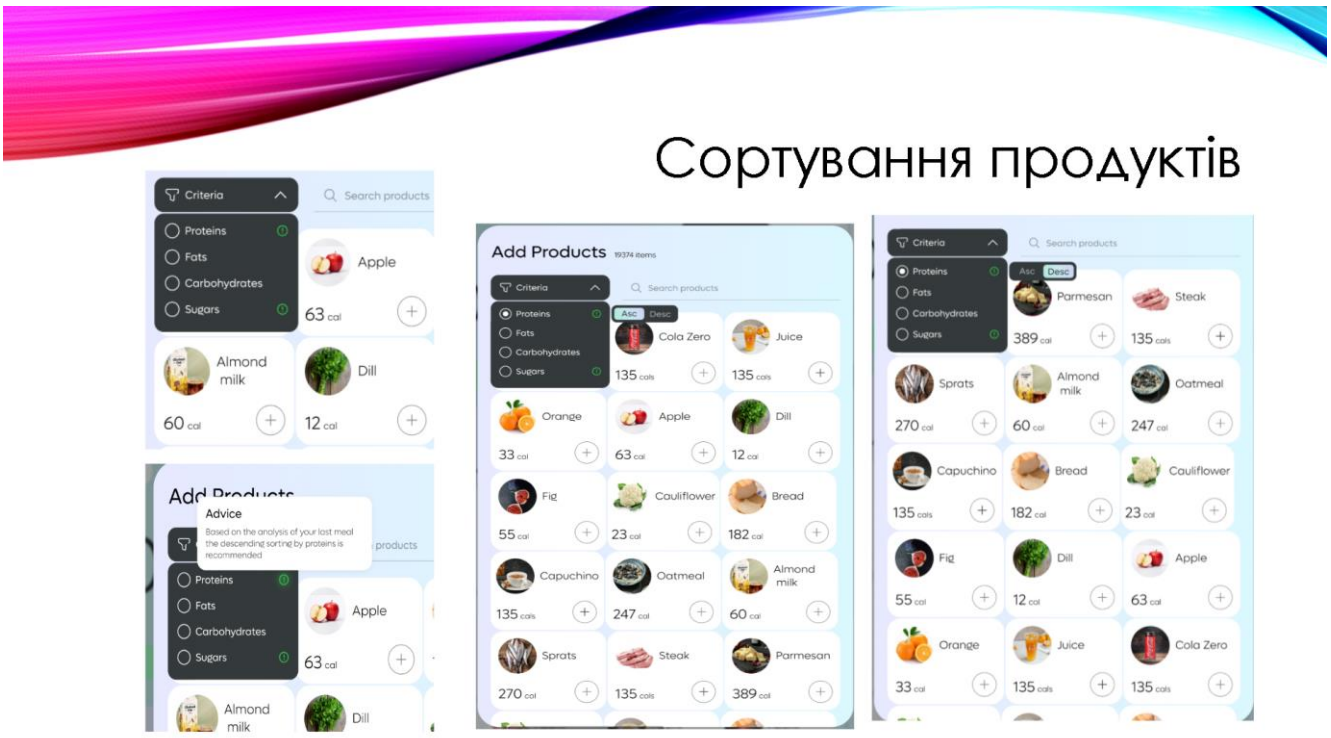


Рисунок Г.18 – Сортування продуктів

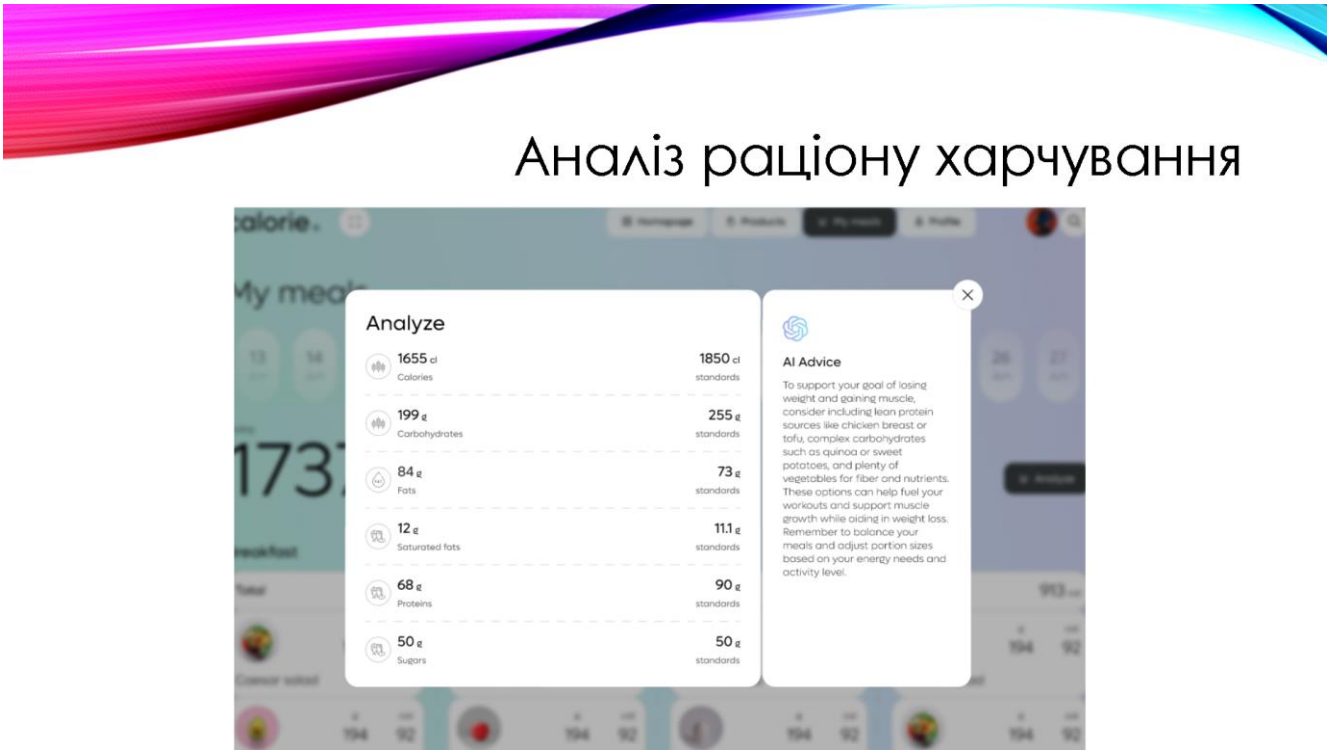


Рисунок Г.19 – Аналіз раціону харчування

Рисунок Г.20 – Висновки

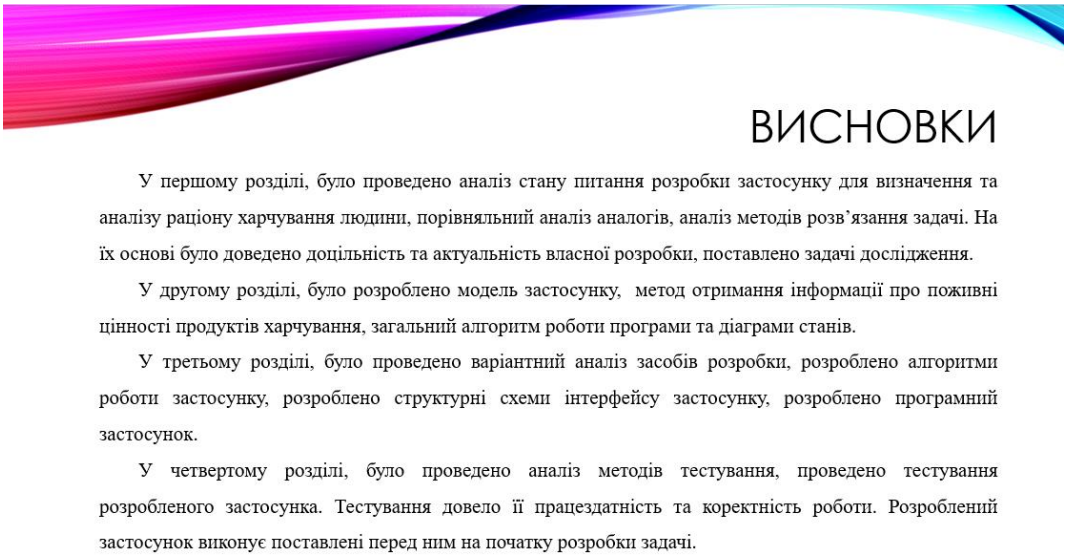


Рисунок Г.21 – Висновки (продовження)

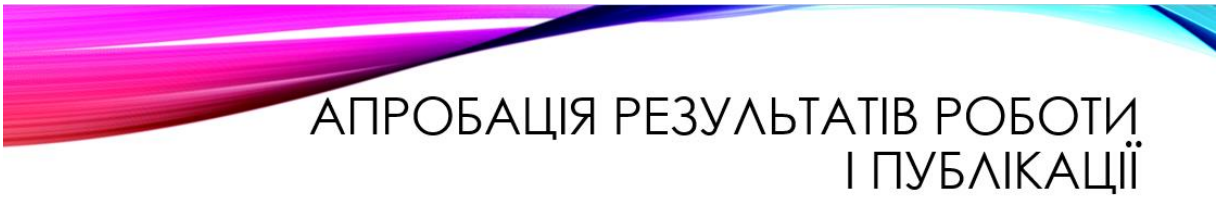


Рисунок Г.22 – Апробація результатів роботи і публікації