НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

БІОРЕСУРСІВ І

ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ПОГОДЖЕНО	**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ**
Декан факультету	**Завідувач кафедри комп'ютерних наук**
_____**Ігор БОЛБОТ**	_____**Белла ГОЛУБ**
(підпис)	(підпис)
«___» _____2026 р.	«___» _____2026 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Програмне забезпечення для моніторингу калорій та харчування»
Спеціальність «121 Інженерія програмного забезпечення»
Освітньо-професійна програма «Інженерія програмного забезпечення»

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

Керівник бакалаврської
кваліфікаційної роботи _____ **Юрій Міловідов**
(підпис)
ст.викладач

Виконав _____ **Микола Бобко**
(підпис)

Київ-2026

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук

к.т.н., доцент _____Белла ГОЛУБ
(підпис)

«___» _____2025 р.

ЗАВДАННЯ ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ ЗДОБУВАЧУ

Бобку Миколі Михайловичу

Спеціальність «121 Інженерія програмного забезпечення»
Освітньо-професійна програма «Інженерія програмного забезпечення» Тема бакалаврської кваліфікаційної роботии

« Програмне забезпечення для моніторингу калорій та харчування»

затверджена наказом від «19 грудня» 2025 р. № 3204«С» Термін

подання завершеної роботи на кафедру «_____» _____2026 р.

Вихідні дані до бакалаврської кваліфікаційної роботи:
Програмний прототип системи моніторингу калорій та харчування

Перелік питань, що підлягають дослідженню:
Методи, моделі та програмні засоби створення системи моніторингу калорійності, поживної цінності продуктів і відповідності раціону індивідуальним цілям користувача

Дата видачі завдання «___» _____2026 р.

Керівник бакалаврської кваліфікаційної робота _____ **Юрій МІЛОВІДОВ**
(підпис)

Завдання взяв до виконання _____ **Микола БОБКО**
(підпис)

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

– БД: база даних.
– ІС: інформаційна система.
– ПЗ: програмне забезпечення.
– БЖВ: білки, жири, вуглеводи.
– CRUD: створення, перегляд, оновлення та видалення даних.
– UI: інтерфейс користувача.
– UX: користувацький досвід.
– API: програмний інтерфейс застосунку.
– REST: архітектурний підхід до побудови вебсервісів.
– HTTP: протокол передавання гіпертексту.
– HTTPS: захищений протокол передавання гіпертексту.
– HTML: мова розмітки вебсторінок.
– CSS: каскадні таблиці стилів.
– JS: JavaScript, мова програмування для вебзастосунків.
– Node.js: середовище виконання JavaScript на сервері.
– Express.js: фреймворк Node.js для створення серверної частини вебзастосунку.
– npm: менеджер пакетів для Node.js.
– SQLite: вбудована реляційна система керування базами даних.
– SQL: мова структурованих запитів до баз даних.
– sql.js: бібліотека для роботи з SQLite у середовищі JavaScript.
– CSV: формат збереження табличних даних.
– PDF: формат електронного документа.
– URL: уніфікований покажчик ресурсу.
– bcrypt: алгоритм хешування паролів.
– session: механізм збереження стану користувача між запитами.

– middleware: проміжний програмний компонент оброблення запитів.

– dashboard: інформаційна панель користувача.

– NutriTrack: розроблена інформаційна система моніторингу калорій та харчування.

ВСТУП

Раціональне харчування, контроль енергетичної цінності раціону та необхідність регулярного відстеження харчових звичок зумовлюють підвищені вимоги до програмних засобів, які забезпечують моніторинг калорійності, поживної цінності продуктів і відповідності фактичного харчування індивідуальним цілям користувача.

Процес моніторингу калорій та харчування охоплює комплекс взаємопов’язаних дій, включаючи ведення профілю користувача, визначення добової норми калорій, внесення прийомів їжі, вибір продуктів із бази даних, розрахунок білків, жирів, вуглеводів та загальної енергетичної цінності раціону. За відсутності спеціалізованого програмного забезпечення ці дії часто виконуються вручну або за допомогою розрізнених таблиць і нотаток, що призводить до неточностей у підрахунках, втрати історії харчування та ускладнює аналіз динаміки змін.

Традиційні підходи до контролю харчування, як правило, не забезпечують цілісного інформаційного супроводу користувача на всіх етапах формування раціону. Відсутність централізованого обліку прийомів їжі, бази продуктів, персональних цілей, добових норм і аналітичних показників ускладнює оцінювання фактичного стану харчування та не дозволяє своєчасно виявляти перевищення або недостатність споживання калорій і поживних речовин.

Використання сучасних програмних систем для моніторингу калорій та харчування відкриває можливості для автоматизації процесів обліку раціону, розрахунку харчової цінності продуктів, формування статистики та надання рекомендацій користувачу. Інтеграція механізмів зберігання, оброблення та аналізу даних дозволяє підвищити точність розрахунків, забезпечити зручність ведення харчового щоденника та створити основу для персоналізованого контролю харчових звичок.

Метою кваліфікаційної роботи є проєктування та розробка програмного забезпечення для моніторингу калорій та харчування, яке забезпечує автоматизований облік прийомів їжі, розрахунок калорійності та харчової цінності раціону, контроль добових норм і формування аналітичної інформації для користувача.

Для досягнення поставленої мети в роботі передбачається розв’язання таких **завдань:**

1. дослідити предметну область моніторингу калорій та харчування, визначити основні процеси обліку раціону й аналізу харчових показників;
2. проаналізувати існуючі програмні рішення для контролю харчування, визначити їх переваги та недоліки;
3. сформулювати функціональні, нефункціональні та безпекові вимоги до розроблюваного програмного забезпечення;
4. розробити UML-моделі, що описують структуру, поведінку та взаємодію основних компонентів системи;
5. спроєктувати логічну та фізичну модель даних для зберігання інформації про користувачів, продукти, прийоми їжі, цілі та аналітичні показники;
6. реалізувати програмний прототип системи моніторингу калорій та харчування;
7. провести тестування програмних модулів і оцінити коректність роботи системи під час обліку раціону, розрахунку калорійності та формування результатів.

Об’єктом дослідження є процес обліку, контролю та аналізу харчування користувача.

Предметом дослідження є методи, моделі та програмні засоби створення системи моніторингу калорійності, поживної цінності продуктів і відповідності раціону індивідуальним цілям користувача.

Наукова новизна роботи полягає у розробленні підходу до побудови програмного забезпечення, яке поєднує облік прийомів їжі, розрахунок

калорійності, аналіз харчової цінності та контроль добових норм в єдиному інформаційному середовищі. Запропоноване рішення дає змогу підвищити точність ведення харчового щоденника та забезпечити користувача узагальненою інформацією про стан його раціону.

Методи дослідження включають застосування системного аналізу для вивчення предметної області, UML-моделювання для формалізації функцій і структури системи, методи об’єктно-орієнтованого проєктування для побудови програмної архітектури, а також методи тестування програмного забезпечення для перевірки коректності реалізованих функцій. Для організації даних використовується структурований підхід до зберігання інформації про користувачів, продукти, прийоми їжі, цілі харчування та результати розрахунків.

Практична значущість одержаних результатів полягає у можливості використання розробленого програмного забезпечення для щоденного контролю харчування, ведення персонального харчового щоденника, розрахунку калорійності раціону та аналізу споживання білків, жирів і вуглеводів. Система може бути застосована користувачами, які прагнуть упорядкувати власне харчування, контролювати добову норму калорій, відстежувати прогрес і приймати обґрунтовані рішення щодо коригування раціону.

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області програмного забезпечення для моніторингу калорій та харчування

Предметна область програмного забезпечення для моніторингу калорій та харчування охоплює процеси обліку щоденного раціону користувача, розрахунку енергетичної та поживної цінності продуктів, контролю добових норм споживання і формування аналітичної інформації щодо харчових звичок. Основним призначенням такої системи є автоматизація ведення харчового щоденника, зменшення кількості ручних розрахунків і надання користувачу зручного інструменту для контролю власного раціону.

Актуальність розроблення програмного забезпечення для моніторингу калорій та харчування зумовлена тим, що значна частина користувачів не має зручного способу системно фіксувати прийоми їжі, відстежувати кількість спожитих калорій, білків, жирів і вуглеводів, а також порівнювати фактичні показники з індивідуальними цілями. Використання паперових записів, електронних таблиць або неструктурованих нотаток не забезпечує достатньої точності, швидкості оброблення даних і можливості подальшого аналізу.

У межах предметної області ключовим об’єктом є **запис прийому їжі**, який містить інформацію про продукт, масу порції, дату й час споживання, калорійність та показники БЖВ. На основі цих даних система розраховує добовий підсумок, визначає відхилення від установленої норми, формує статистику за період і надає користувачу узагальнену інформацію про стан харчування.

Основними учасниками процесу є користувач і адміністратор. Користувач створює профіль, задає персональні параметри, визначає цілі харчування, додає прийоми їжі та переглядає результати. Адміністратор забезпечує супровід бази продуктів, модерацію даних і контроль коректності інформації, що використовується системою для розрахунків.

На рисунку 1.1 наведено DFD-діаграму 0-го рівня, яка відображає загальну взаємодію користувача, адміністратора, зовнішньої бази харчової цінності, сервісу сповіщень і внутрішніх сховищ даних. Діаграма узагальнює основні інформаційні потоки без деталізації внутрішньої логіки окремих модулів.

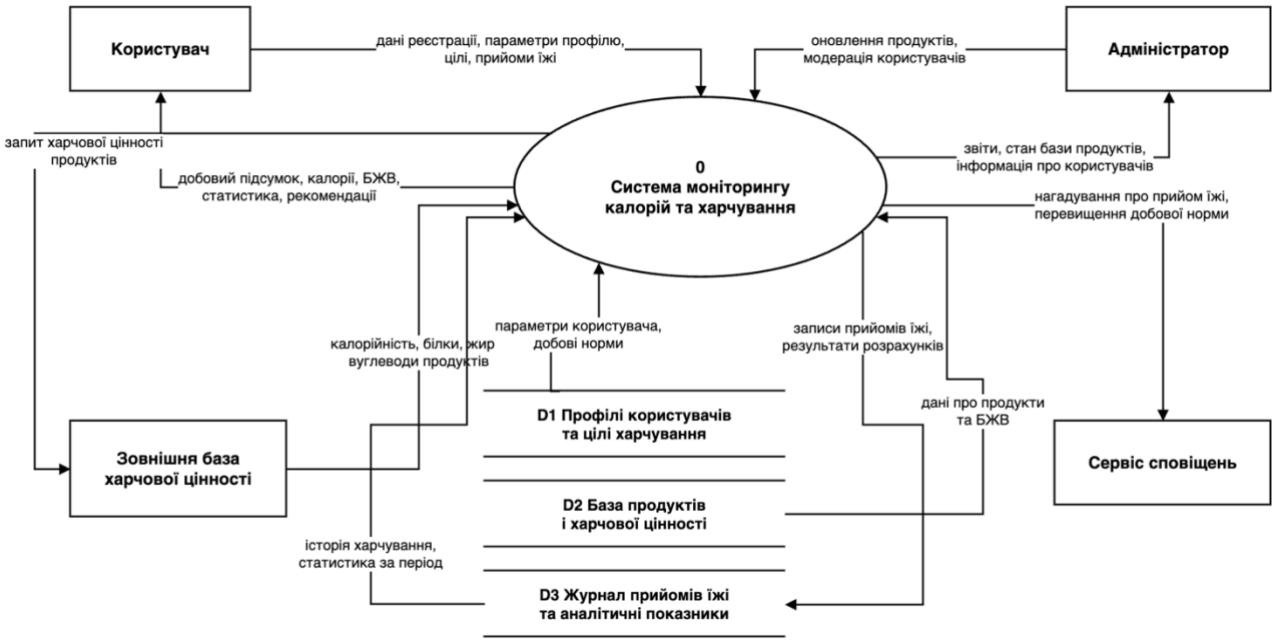

Рисунок 1.1 – DFD-діаграма 0-го рівня програмного забезпечення для моніторингу калорій та харчування

У системі передбачається централізоване збереження профілів користувачів, бази продуктів, записів прийомів їжі та аналітичних показників. Такий підхід дозволяє забезпечити цілісність даних, уникнути дублювання інформації та створити основу для подальшого розширення функціональності, наприклад додавання рекомендацій, нагадувань, планування раціону або порівняння результатів за різні періоди.

Таблиця 1.1 відображає основні процеси предметної області, які мають бути підтримані розроблюваним програмним забезпеченням.

Таблиця 1.1 – Основні процеси предметної області системи моніторингу калорій та харчування

№	Процес	Короткий зміст	Результат виконання
1	Реєстрація та ведення профілю користувача	Внесення особистих параметрів користувача, зокрема віку, статі, маси тіла, зросту, рівня активності та харчової цілі	Сформований профіль користувача з індивідуальними параметрами
2	Визначення добової норми	Розрахунок або встановлення цільового добового споживання калорій, білків, жирів і вуглеводів	Добові нормативні значення для подальшого контролю
3	Ведення бази продуктів	Зберігання інформації про назву продукту, калорійність, білки, жири, вуглеводи та одиниці вимірювання	Актуальна база продуктів для швидкого додавання до раціону
4	Додавання прийому їжі	Вибір продуктів, зазначення порції, дати, часу та типу прийому їжі	Створений запис у журналі харчування
5	Розрахунок харчової цінності	Автоматичне обчислення калорійності та БЖВ для окремого прийому їжі й добового раціону	Розраховані показники калорійності та поживної цінності
6	Аналіз добового раціону	Порівняння фактичного споживання з установленими нормами	Визначення залишку або перевищення добової норми
7	Формування статистики	Узагальнення даних за день, тиждень або інший період	Аналітичні показники та динаміка харчування
8	Надсилання сповіщень	Інформування користувача про прийом їжі, перевищення норми або необхідність оновлення записів	Своєчасне повідомлення користувача
9	Адміністрування системи	Модерація користувачів, оновлення бази продуктів і контроль коректності даних	Підтримання працездатності та актуальності системи

Інформаційна система повинна забезпечувати не лише збереження введених даних, а й їхню змістовну обробку. Зокрема, після додавання продуктів до прийому їжі система має автоматично визначати сумарну калорійність, кількість білків, жирів і вуглеводів, після чого оновлювати добову статистику користувача. Це дає змогу швидко оцінити, наскільки фактичний раціон відповідає поставленій меті.

Окреме значення має база продуктів і харчової цінності. Вона виступає джерелом довідкових даних, на основі яких здійснюються всі подальші розрахунки. Доцільно передбачити можливість як використання наявної бази продуктів, так і додавання нових позицій користувачем або адміністратором. Це підвищує гнучкість системи та дозволяє адаптувати її до реальних харчових звичок користувача.

Аналітична складова предметної області передбачає формування добових і періодичних підсумків. Користувач повинен мати змогу переглядати кількість спожитих калорій, структуру раціону за БЖВ, історію харчування та прогрес відносно встановленої цілі. Такий підхід перетворює систему з простого засобу обліку на інструмент підтримки прийняття рішень щодо коригування харчування.

Основні інформаційні об’єкти предметної області наведено в таблиці 1.2.

Таблиця 1.2 – Основні інформаційні об’єкти системи

№	Об’єкт	Призначення
1	Користувач	Зберігає облікові дані та забезпечує персоналізовану роботу із системою
2	Профіль користувача	Містить фізичні параметри, рівень активності та індивідуальні налаштування
3	Ціль харчування	Визначає бажаний режим: підтримання ваги, зниження маси або набір маси
4	Продукт	Описує харчовий продукт, його калорійність і показники БЖВ
5	Прийом їжі	Фіксує факт споживання продуктів у певний час
6	Журнал харчування	Накопичує історію прийомів їжі користувача
7	Аналітичний показник	Відображає підсумкові значення калорійності, БЖВ і прогресу
8	Сповіщення	Забезпечує нагадування та інформування користувача про важливі події
9	Адміністратор	Виконує супровід системи, модерацію даних і оновлення довідників

Отже, предметна область програмного забезпечення для моніторингу калорій та харчування поєднує процеси персонального обліку, довідкового зберігання продуктів, автоматизованого розрахунку харчової цінності та аналітичного оцінювання раціону. Розроблення такої системи дозволяє

підвищити зручність контролю харчування, зменшити ймовірність помилок у розрахунках і забезпечити користувача актуальною інформацією для щоденного прийняття рішень щодо власного раціону.

1.2 Аналіз існуючих рішень

Для визначення функціональних можливостей розроблюваного програмного забезпечення було проаналізовано сучасні системи моніторингу калорій та харчування. Основну увагу приділено застосункам, які забезпечують ведення харчового щоденника, облік калорій, розрахунок білків, жирів і вуглеводів, формування цілей харчування, перегляд статистики та підтримку бази продуктів.

Серед найбільш поширених рішень у цій предметній області доцільно виділити MyFitnessPal, YAZIO, Lifesum, FatSecret та Cronometer. Ці системи мають схожу базову логіку роботи, однак відрізняються рівнем деталізації харчових показників, способом побудови інтерфейсу, наявністю рекомендацій, планів харчування та можливостями аналітики.

На рисунку 1.2 подано інтерфейс застосунку MyFitnessPal, який є одним із найбільш відомих сервісів для контролю харчування. Система поєднує лічильник калорій, харчовий щоденник, облік фізичної активності, води, ваги та періодичного голодування. Користувач може додавати продукти, переглядати залишок калорій за день, контролювати макронутрієнти та відстежувати прогрес. Перевагою MyFitnessPal є велика база продуктів і розвинена інтеграція з фітнес-сервісами. Недоліком є перевантаженість функціями та обмеження частини можливостей у безкоштовній версії.

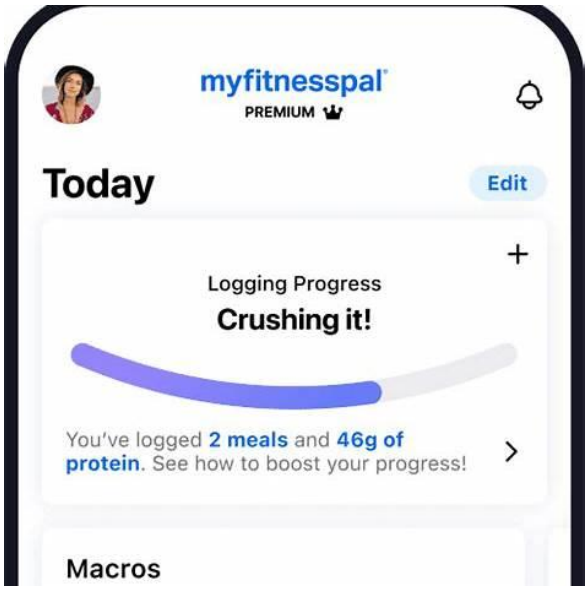

Рисунок 1.2 – Інтерфейс застосунку MyFitnessPal

На рисунку 1.3 наведено інтерфейс YAZIO - мобільного застосунку для підрахунку калорій, контролю харчування та досягнення цілей, пов'язаних зі зниженням або підтриманням маси тіла. Система підтримує облік прийомів їжі, формування плану харчування, контроль калорійного дефіциту та використання AI-функцій для спрощення введення даних. Перевагою YAZIO є простий інтерфейс і орієнтація на щоденне використання. Водночас частина розширених можливостей, зокрема поглиблені плани та персональні рекомендації, доступна переважно в платній версії.

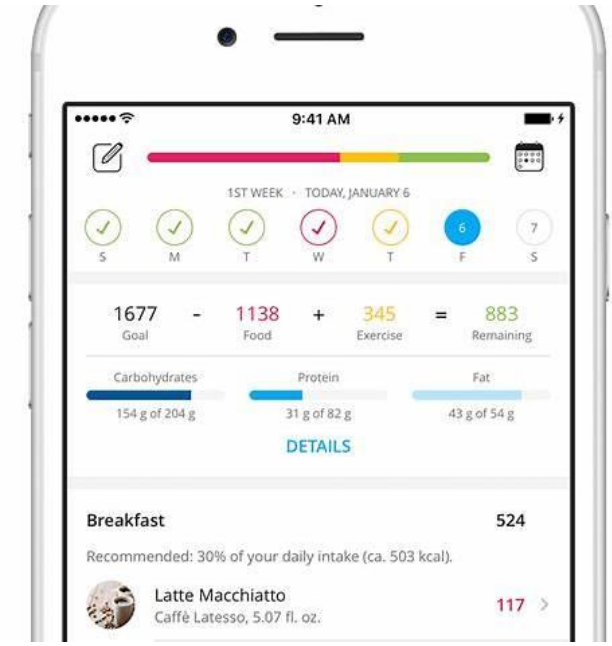

Рисунок 1.3 – Інтерфейс застосунку YAZIO

На рисунку 1.4 представлено Lifesum - застосунок для контролю харчування, який поєднує підрахунок калорій, трекер макронутрієнтів, планування раціону, рецепти, контроль води та персоналізовані рекомендації. Система орієнтована не лише на фіксацію спожитих продуктів, а й на формування корисних харчових звичок. Її перевагою є наявність готових планів харчування, списків покупок і зручної візуалізації прогресу. Недоліком є залежність значної частини функцій від підписки та менша гнучкість для користувачів, яким потрібен простий локальний облік без зайвих сервісів.

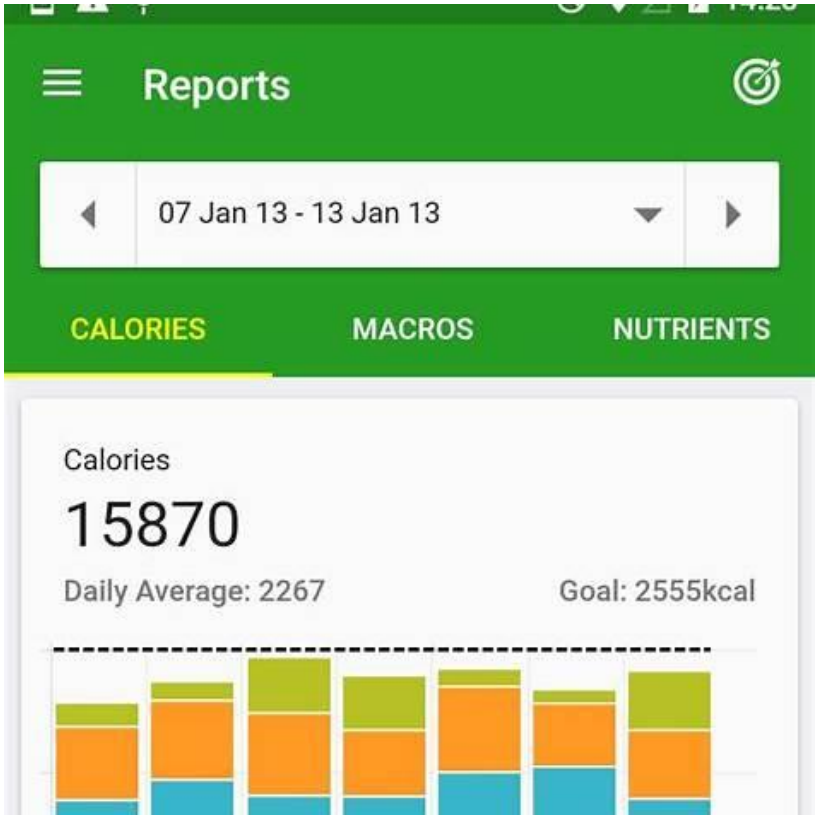

Рисунок 1.4 – Інтерфейс застосунку Lifesum

На рисунку 1.5 наведено інтерфейс FatSecret, який забезпечує облік їжі, тренувань, ваги, макронутрієнтів і перегляд дієтичного календаря. Система має велику базу продуктів, підтримує сканування штрихкодів, ведення журналу прогресу та формування звітів. Перевагою FatSecret є доступність значної частини базових функцій без оплати. Водночас інтерфейс системи менш адаптований до гнучкого персонального налаштування, а частина розширених інструментів доступна у Premium-версії.

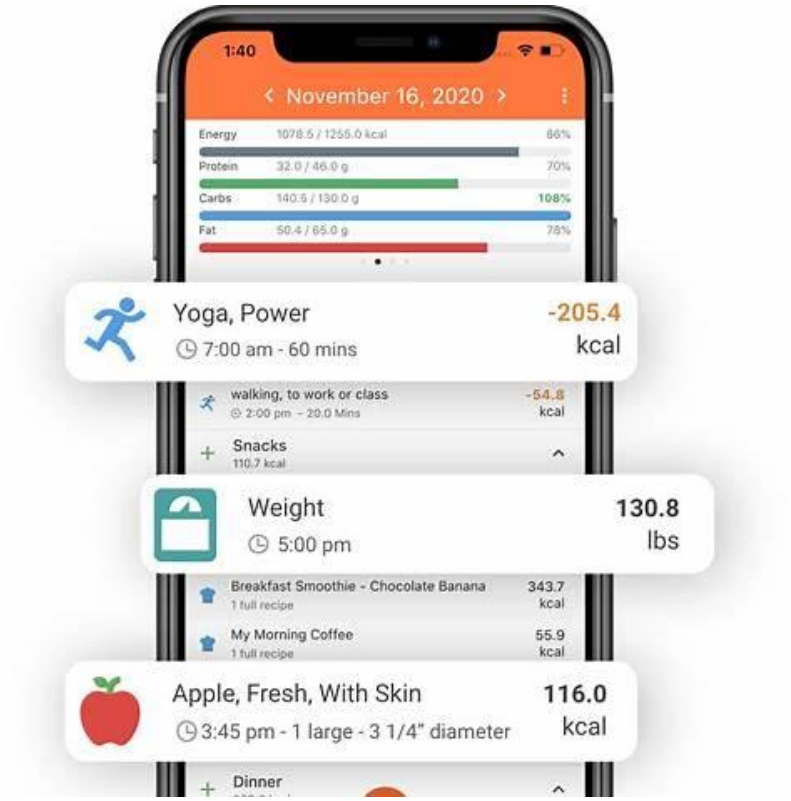

Рисунок 1.5 – Інтерфейс застосунку FatSecret

На рисунку 1.6 подано приклад інтерфейсу Cronometer — системи, що робить акцент на точному аналізі харчової цінності та деталізованому обліку нутрієнтів. Cronometer дозволяє відстежувати калорії, макронутрієнти, мікронутрієнти, воду, фізичну активність і біометричні показники. Перевагою системи є висока деталізація харчового аналізу та підтримка великої кількості нутрієнтів. Недоліком є складніший інтерфейс для звичайного користувача, якому потрібен швидкий щоденний облік калорій без надмірної деталізації.

Рисунок 1.6 – Інтерфейс застосунку Cronometer

Порівняльний аналіз існуючих рішень наведено в таблиці 1.3.

Таблиця 1.3 – Порівняння існуючих систем моніторингу калорій та харчування

Система	Тип рішення	Облік калорій і БЖВ	Планування раціону	Аналітика	Основне обмеження
MyFitnessPal	Мобільний / вебсервіс	Так	Частково	Так	Частина функцій доступна за підпискою
YAZIO	Мобільний застосунок	Так	Так	Так	Розширені рекомендації переважно платні
Lifesum	Мобільний застосунок	Так	Так	Так	Орієнтація на готові плани, менше гнучкості
FatSecret	Мобільний / вебсервіс	Так	Частково	Так	Менш сучасна логіка персоналізації
Cronometer	Мобільний / вебсервіс	Так	Частково	Розширена	Надмірна деталізація для базового користувача
Розроблювана система	Програмне забезпечення / клієнт–сервер	Так	Так	Так	Орієнтація на простий персональний

					облік і навчальний прототип

Результати аналізу показують, що наявні рішення забезпечують широкий набір функцій для контролю харчування, однак часто є перевантаженими, залежать від хмарних сервісів або обмежують частину можливостей платною підпискою. Для навчального програмного продукту доцільно реалізувати компактну систему, яка зосереджується на ключових функціях: веденні профілю користувача, базі продуктів, додаванні прийомів їжі, автоматичному розрахунку калорій і БЖВ, перегляді добового підсумку, статистики та рекомендацій.

Розроблюване програмне забезпечення для моніторингу калорій та харчування має бути простішим за комерційні аналоги, але водночас функціонально достатнім для щоденного використання. Основний акцент робиться на зрозумілому інтерфейсі, локальному або централізованому збереженні даних, швидкому додаванні продуктів, контролі добових норм і формуванні аналітичної інформації без зайвого ускладнення користувацького сценарію.

1.3 Моделювання програмної системи

Моделювання предметної області виконано з метою формалізації основних функцій програмного забезпечення для моніторингу калорій та харчування, визначення ролей користувачів, послідовності виконання операцій і логіки оброблення даних. Для опису системи використано UML-діаграми прецедентів, послідовності та активності, які дозволяють подати роботу програмного забезпечення з різних точок зору.

У межах предметної області система розглядається як програмний засіб, що забезпечує реєстрацію користувача, налаштування персонального профілю, встановлення цілей харчування, ведення журналу прийомів їжі, пошук

продуктів, розрахунок калорійності та БЖВ, а також формування щоденних і періодичних аналітичних звітів. Окремо передбачено адміністративні функції, пов’язані з керуванням базою продуктів і модерацією користувачів.

На рисунку 1.7 подано діаграму прецедентів системи моніторингу калорій та харчування.

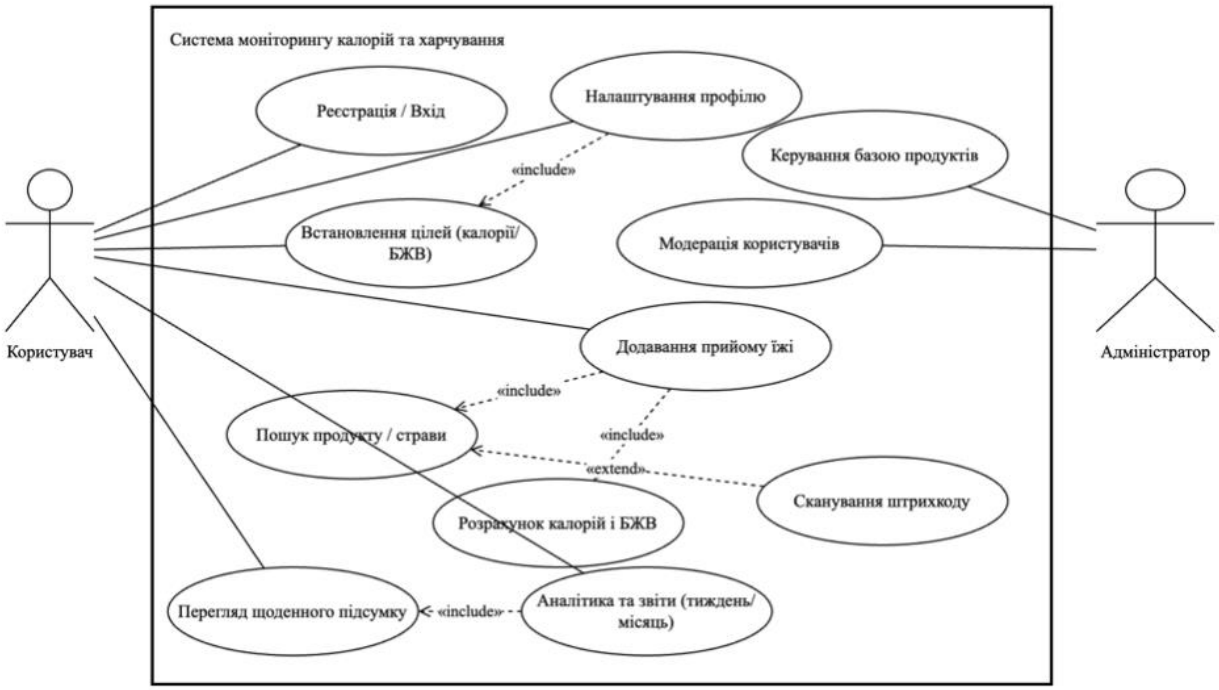

Рисунок 1.7 - Діаграма прецедентів системи моніторингу калорій та харчування

Діаграма прецедентів відображає дві основні ролі: користувача та адміністратора. Користувач працює з основним функціоналом системи: проходить реєстрацію або вхід, налаштовує профіль, встановлює цілі за калоріями та БЖВ, додає прийоми їжі, виконує пошук продуктів або страв, переглядає щоденний підсумок і аналітичні звіти. Адміністратор відповідає за супровід системи, зокрема керує базою продуктів і виконує модерацію користувачів.

Основні прецеденти системи наведено в таблиці 1.4.

Таблиця 1.4 - Основні прецеденти програмного забезпечення

Актор	Прецедент	Зміст операції	Результат
Користувач	Реєстрація / вхід	Введення облікових даних і перевірка доступу	Користувач отримує доступ до системи
Користувач	Налаштування профілю	Внесення віку, статі, маси, зросту, рівня активності	Сформовано персональний профіль
Користувач	Встановлення цілей	Визначення добової норми калорій, білків, жирів і вуглеводів	Збережено цільові показники
Користувач	Пошук продукту / страви	Пошук позиції у базі продуктів або введення назви страви	Отримано дані про харчову цінність
Користувач	Сканування штрихкоду	Швидке визначення продукту за кодом	Продукт знайдено або запропоновано додати вручну
Користувач	Додавання прийому їжі	Вибір продукту, маси порції, часу та типу прийому	Створено запис у журналі харчування
Користувач	Розрахунок калорій і БЖВ	Обчислення харчової цінності прийому їжі	Оновлено добовий підсумок
Користувач	Перегляд аналітики	Перегляд статистики за день, тиждень або місяць	Сформовано звіт про харчування
Адміністратор	Керування базою продуктів	Додавання, редагування та видалення продуктів	База продуктів підтримується в актуальному стані
Адміністратор	Модерація користувачів	Перегляд і керування обліковими записами	Забезпечено контроль користувачів системи

Зв’язки між прецедентами уточнюють логіку роботи системи. Наприклад, встановлення цілей пов’язане з налаштуванням профілю, оскільки добова норма залежить від персональних параметрів користувача. Додавання прийому їжі включає пошук продукту та розрахунок калорійності. Сканування штрихкоду розглядається як додатковий варіант пошуку продукту, який спрощує введення даних.

На рисунку 1.8 подано діаграму послідовності, що описує типовий сценарій роботи користувача із системою.

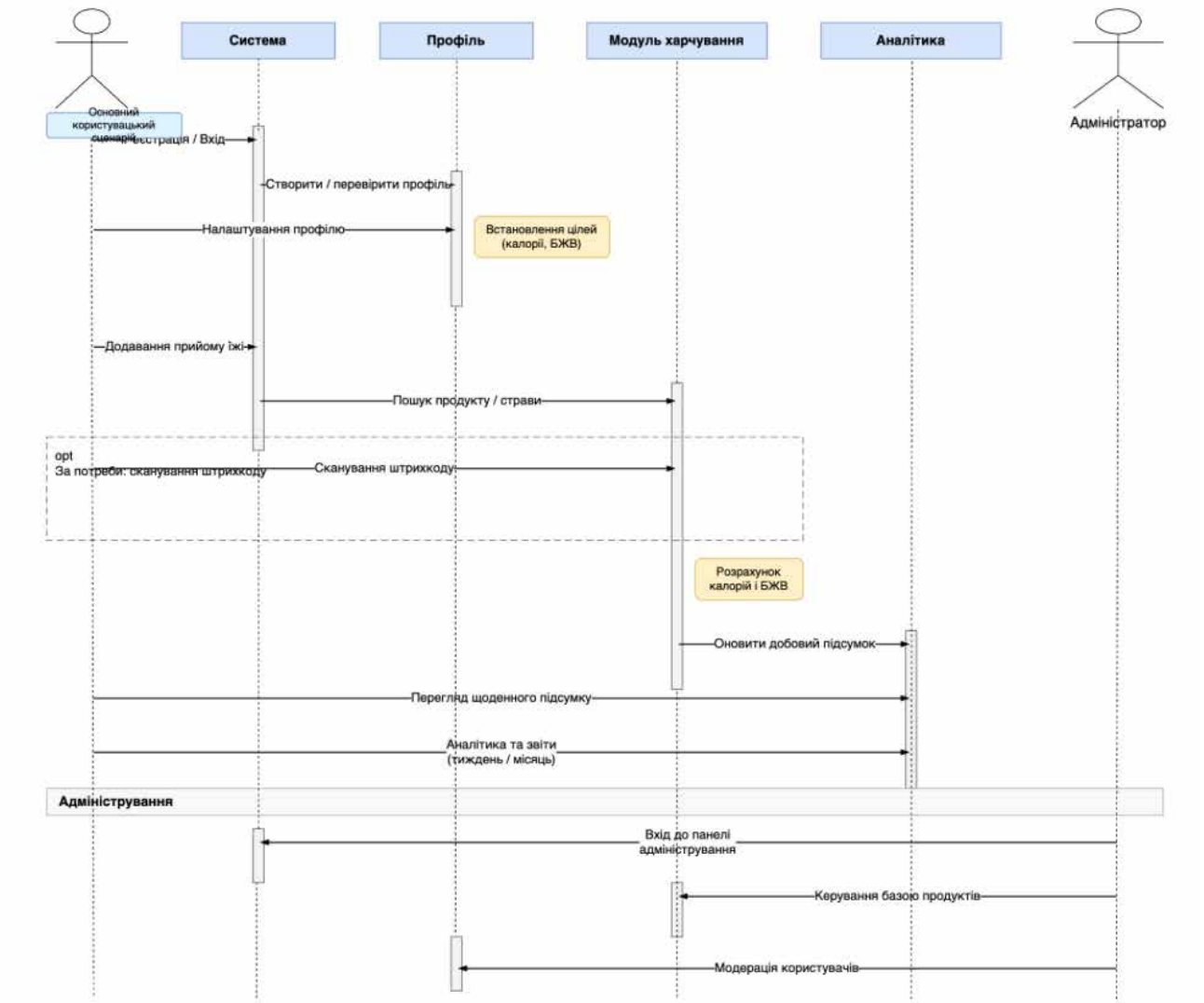

Рисунок 1.8 - Діаграма послідовності роботи системи моніторингу калорій та харчування

Діаграма послідовності демонструє взаємодію між користувачем, системою, профілем, модулем харчування та модулем аналітики. Спочатку користувач виконує реєстрацію або вхід, після чого система створює або перевіряє профіль. Далі користувач налаштовує персональні параметри та встановлює цілі за калоріями і БЖВ.

Після цього користувач додає прийом їжі. Система передає запит до модуля харчування, де виконується пошук продукту або страви. За потреби може використовуватися сканування штрихкоду. Після вибору продукту система розраховує калорійність і БЖВ, оновлює добовий підсумок і передає

дані до модуля аналітики. Користувач може переглядати щоденний підсумок, а також звіти за тиждень або місяць.

Адміністратор взаємодіє із системою через панель адміністрування. Його основні дії пов’язані з керуванням базою продуктів і модерацією користувачів. Це дозволяє підтримувати коректність довідкової інформації та забезпечувати стабільну роботу системи.

Основні етапи сценарію взаємодії наведено в таблиці 1.5.

Таблиця 1.5 - Послідовність оброблення основного сценарію

№	Етап	Учасник	Дія	Результат
1	Авторизація	Користувач, система	Введення логіна та пароля	Перевірено доступ
2	Формування профілю	Система, профіль	Створення або перевірка профілю	Профіль готовий до роботи
3	Налаштування цілей	Користувач, профіль	Встановлення добових норм	Збережено цілі харчування
4	Додавання їжі	Користувач, система	Внесення нового прийому їжі	Передано запит до модуля харчування
5	Пошук продукту	Модуль харчування	Пошук у базі продуктів або сканування штрихкоду	Отримано дані про продукт
6	Розрахунок	Модуль харчування	Обчислення калорій, білків, жирів і вуглеводів	Сформовано харчові показники
7	Оновлення підсумку	Модуль харчування, аналітика	Передача результатів до аналітики	Оновлено добову статистику
8	Перегляд результатів	Користувач, аналітика	Відображення підсумку та звітів	Користувач отримує аналітичну інформацію
9	Адміністрування	Адміністратор, система	Оновлення продуктів і модерація користувачів	Актуалізовано системні дані

На рисунку 1.9 наведено діаграму активності, яка деталізує загальну логіку виконання процесів у системі.

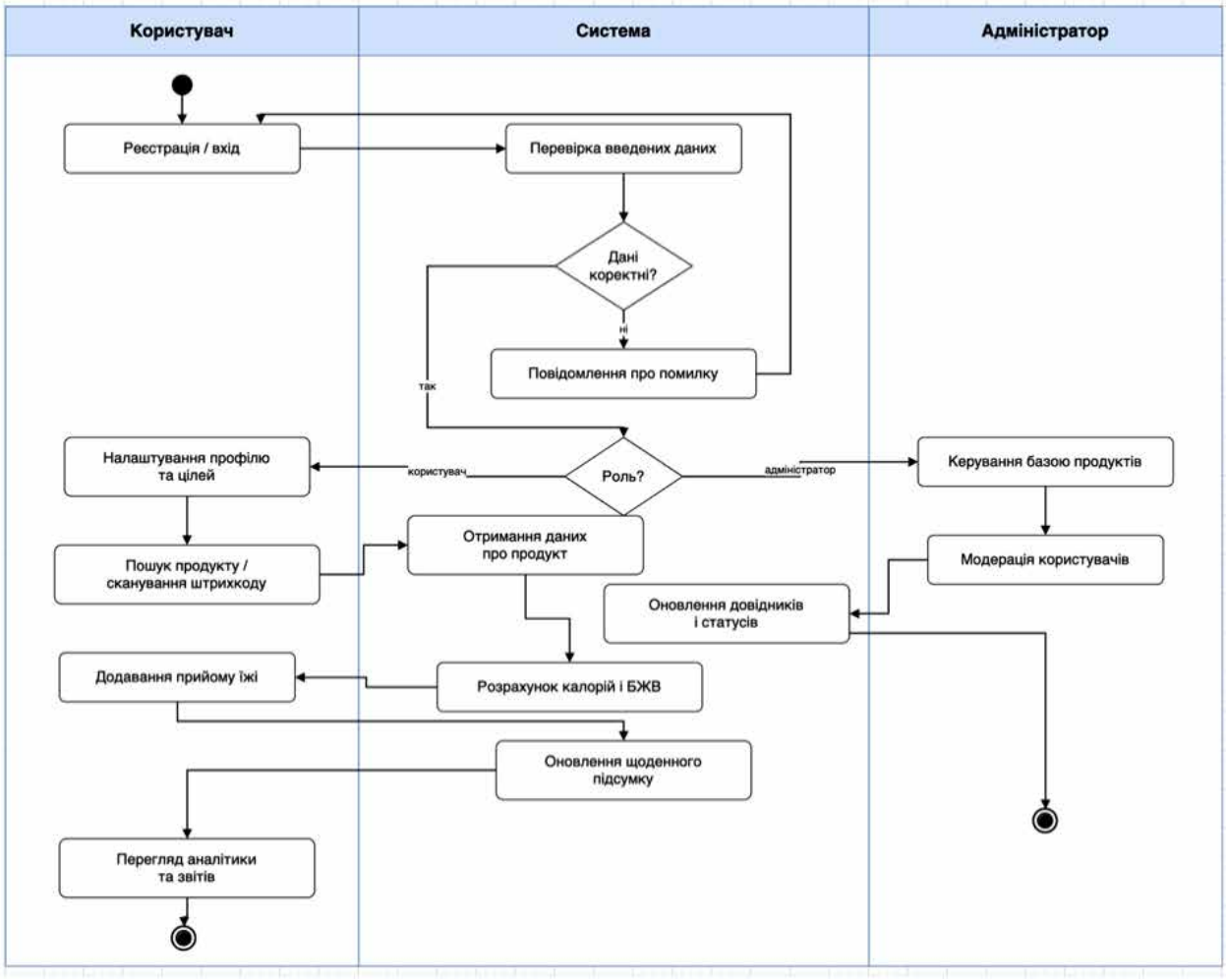

Рисунок 1.9 - Діаграма активності процесу моніторингу калорій та харчування

Діаграма активності поділена на три зони відповідальності: користувач, система та адміністратор. Процес починається з реєстрації або входу. Після введення даних система виконує перевірку. Якщо дані некоректні, користувач отримує повідомлення про помилку. Якщо перевірка успішна, система визначає роль користувача.

Для звичайного користувача основний сценарій передбачає налаштування профілю та цілей, пошук продукту або сканування штрихкоду, додавання прийому їжі, розрахунок калорій і БЖВ, оновлення щоденного підсумку та перегляд аналітики. Для адміністратора передбачено керування базою продуктів, модерацію користувачів і оновлення довідників або статусів.

Особливістю моделі є те, що обчислення калорійності та БЖВ виконується системою автоматично після отримання даних про продукт і масу

порції. Це зменшує кількість ручних дій користувача та підвищує точність обліку харчування.

Основні інформаційні об’єкти, які використовуються в моделі предметної області, наведено в таблиці 1.6.

Таблиця 1.6 - Інформаційні об’єкти предметної області

Об’єкт	Основні дані	Призначення
Користувач	Ідентифікатор, ім’я, email, пароль, роль	Забезпечує персоналізований доступ до системи
Профіль	Вік, стать, маса, зріст, рівень активності	Використовується для розрахунку індивідуальних норм
Ціль харчування	Калорії, білки, жири, вуглеводи, тип цілі	Визначає добові орієнтири користувача
Продукт	Назва, калорійність, білки, жири, вуглеводи	Використовується при формуванні прийомів їжі
Прийом їжі	Дата, час, тип прийому, список продуктів	Фіксує факт споживання їжі
Запис продукту в прийомі	Продукт, маса порції, розраховані показники	Зберігає деталізацію прийому їжі
Щоденний підсумок	Калорії, БЖВ, залишок норми, перевищення	Відображає стан харчування за день
Звіт	Період, середні значення, динаміка	Використовується для аналізу харчування
Сповіщення	Тип, текст, дата, статус	Інформує користувача про важливі події

Виконане моделювання предметної області дозволило визначити основні ролі, функції, інформаційні потоки та порядок роботи програмного забезпечення. Побудовані UML-діаграми підтверджують, що система повинна забезпечувати повний цикл обліку харчування: від створення профілю та встановлення цілей до додавання прийомів їжі, автоматичного розрахунку калорійності, формування щоденного підсумку та перегляду аналітичних звітів. Отримані моделі є основою для подальшого проєктування структури даних, архітектури програмного забезпечення та реалізації функціональних модулів системи.

1.4 Аналіз вимог програмної системи

Аналіз вимог є необхідним етапом проєктування програмного забезпечення для моніторингу калорій та харчування, оскільки саме на цьому етапі визначаються основні функції системи, обмеження її роботи, вимоги до збереження даних, безпеки та зручності використання. Розроблювана система повинна забезпечувати користувачу можливість вести персональний харчовий щоденник, контролювати добову норму калорій і БЖВ, переглядати статистику та отримувати узагальнену інформацію про раціон.

Основними користувачами системи є звичайний користувач і адміністратор. Користувач працює з профілем, продуктами, прийомами їжі, добовими підсумками та аналітикою. Адміністратор відповідає за актуальність бази продуктів, контроль облікових записів і підтримання коректності довідкових даних.

Функціональні вимоги до програмної системи наведено в таблиці 1.7.

Таблиця 1.7 - Функціональні вимоги до програмної системи

№	Вимога	Опис	Користувач
1	Реєстрація та вхід	Система повинна забезпечувати створення облікового запису та авторизацію користувача	Користувач, адміністратор
2	Налаштування профілю	Користувач повинен мати змогу вносити вік, стать, масу, зріст і рівень активності	Користувач
3	Встановлення цілей харчування	Система повинна дозволяти задавати добову норму калорій, білків, жирів і вуглеводів	Користувач
4	Ведення бази продуктів	Система повинна зберігати продукти з калорійністю та показниками БЖВ	Адміністратор
5	Пошук продуктів	Користувач повинен мати змогу швидко знаходити продукт або страву за назвою	Користувач
6	Додавання прийому їжі	Система повинна дозволяти додавати продукти до сніданку, обіду, вечері або перекусу	Користувач
7	Розрахунок калорій і БЖВ	Після вибору продукту та порції система повинна автоматично обчислювати харчову цінність	Система

Продовження таблиці 1.7

8	Формування добового підсумку	Система повинна показувати кількість спожитих калорій, БЖВ і залишок добової норми	Користувач
9	Перегляд аналітики	Користувач повинен мати змогу переглядати статистику за день, тиждень або місяць	Користувач
10	Формування звітів	Система повинна формувати узагальнені звіти про раціон і динаміку харчування	Користувач
11	Сповіщення	Система повинна нагадувати про прийом їжі або повідомляти про перевищення добової норми	Користувач
12	Модерація користувачів	Адміністратор повинен мати змогу переглядати та керувати обліковими записами	Адміністратор
13	Адміністрування продуктів	Адміністратор повинен додавати, редагувати та видаляти продукти в базі	Адміністратор

Нефункціональні вимоги визначають якість роботи програмної системи, її продуктивність, надійність, зручність і можливість подальшого розвитку. Для системи моніторингу калорій та харчування важливо забезпечити швидку обробку запитів, стабільне збереження даних і простий інтерфейс, оскільки користувач взаємодіє із системою щодня.

Таблиця 1.8 - Нефункціональні вимоги до програмної системи

№	Вимога	Характеристика
1	Продуктивність	Основні операції пошуку продукту, додавання прийому їжі та оновлення підсумку мають виконуватися без помітної затримки
2	Надійність	Дані користувача, записи прийомів їжі та цілі харчування повинні зберігатися без втрати після завершення роботи системи
3	Зручність інтерфейсу	Інтерфейс має бути зрозумілим, мінімалістичним і придатним для щоденного використання
4	Масштабованість	Структура системи повинна дозволяти додавання нових модулів, наприклад планування раціону або рекомендацій
5	Сумісність	Система повинна працювати на стандартному робочому середовищі користувача без складного налаштування
6	Підтримуваність	Програмний код має бути поділений на логічні модулі: профіль, харчування, аналітика, база даних, адміністрування
7	Точність розрахунків	Розрахунок калорій і БЖВ повинен виконуватися на основі маси порції та довідкових значень продукту

Продовження таблиці 1.8

8	Актуальність даних	База продуктів повинна підтримувати редагування та оновлення харчової цінності
9	Доступність	Користувач повинен мати доступ до основних функцій без складної послідовності дій
10	Відновлення після помилок	У разі некоректного введення даних система повинна показувати повідомлення про помилку без завершення роботи

Оскільки система обробляє персональні дані користувача, параметри тіла, цілі харчування та історію раціону, важливим є дотримання вимог до інформаційної безпеки. Дані мають бути доступні лише авторизованим користувачам, а адміністративні функції повинні бути відокремлені від звичайного режиму роботи.

Таблиця 1.9 - Вимоги до інформаційної безпеки

№	Вимога	Опис
1	Автентифікація	Доступ до системи повинен надаватися лише після перевірки облікових даних
2	Розмежування ролей	Користувач і адміністратор повинні мати різні рівні доступу
3	Захист паролів	Паролі не повинні зберігатися у відкритому вигляді
4	Захист персональних даних	Параметри профілю та історія харчування повинні бути доступні лише власнику облікового запису
5	Контроль введення даних	Система повинна перевіряти коректність числових значень, дат, порцій і параметрів профілю
6	Захист адміністративних функцій	Операції керування продуктами та користувачами повинні бути доступні лише адміністратору
7	Журналювання дій	Доцільно передбачити фіксацію важливих подій, зокрема входу, зміни продуктів і редагування профілю
8	Цілісність даних	Система повинна запобігати створенню неповних або некоректних записів прийомів їжі

З точки зору користувача система повинна бути простою та швидкою у використанні. Основний сценарій має зводитися до вибору продукту, введення маси порції та перегляду оновленого добового підсумку. Надмірна кількість проміжних дій ускладнює роботу із системою та знижує її практичну цінність.

Таблиця 1.10 - Вимоги з точки зору користувача

№	Вимога користувача	Очікуваний результат

1	Швидко додати прийом їжі	Запис з’являється у щоденному журналі
2	Побачити залишок калорій	Система показує спожиту кількість і залишок добової норми
3	Контролювати БЖВ	Відображається кількість білків, жирів і вуглеводів
4	Переглянути історію харчування	Користувач бачить записи за вибраний день або період
5	Отримати статистику	Система формує підсумки за день, тиждень або місяць
6	Оновити профіль	Персональні параметри змінюються та враховуються у подальших розрахунках
7	Отримувати нагадування	Система повідомляє про прийом їжі або перевищення норми

З точки зору програмної системи необхідно забезпечити логічну узгодженість усіх модулів. Дані профілю використовуються для визначення цілей, база продуктів є джерелом розрахунків, журнал харчування накопичує записи, а аналітичний модуль формує підсумкову інформацію.

Таблиця 1.11 - Вимоги з точки зору програмної системи

№	Системна вимога	Призначення
1	Зберігати користувачів і ролі	Забезпечення персоналізованого доступу
2	Зберігати профілі та цілі	Використання персональних параметрів у розрахунках
3	Підтримувати базу продуктів	Надання довідкових даних для обчислення калорій і БЖВ
4	Обробляти записи прийомів їжі	Формування журналу харчування
5	Виконувати автоматичні розрахунки	Оновлення добових підсумків після кожного запису
6	Формувати аналітику	Надання користувачу статистики та звітів
7	Перевіряти введені дані	Запобігання помилкам у профілі, продуктах і порціях
8	Розмежовувати доступ	Захист адміністративних функцій від звичайних користувачів

Отже, розроблювана програмна система повинна забезпечувати повний цикл моніторингу харчування: від створення профілю та встановлення добових цілей до додавання прийомів їжі, автоматичного розрахунку калорійності, перегляду щоденного підсумку та аналізу статистики. Сформовані вимоги є

основою для подальшого проєктування бази даних, архітектури програмного забезпечення, користувацького інтерфейсу та модулів оброблення інформації.

1.5 Постановка завдання

На основі проведеного аналізу предметної області, сформульованих функціональних і нефункціональних вимог, а також результатів UML-моделювання було сформовано постановку завдання розроблення програмного забезпечення для моніторингу калорій та харчування.

Завдання полягає у проєктуванні та розробленні програмної системи, призначеної для автоматизації процесів ведення харчового щоденника, обліку прийомів їжі, розрахунку калорійності та показників БЖВ, контролю добових норм і формування аналітичної інформації щодо раціону користувача.

Проєктована система розглядається як програмний комплекс із клієнт-серверною архітектурою, що забезпечує персоналізовану роботу користувача з власними даними харчування. Система повинна підтримувати повний цикл моніторингу раціону: від створення профілю та встановлення індивідуальних цілей до додавання прийомів їжі, автоматичного розрахунку харчової цінності, перегляду щоденного підсумку та аналізу статистики за вибраний період.

Для досягнення поставленої мети необхідно спроєктувати архітектуру системи, що складається з взаємопов’язаних функціональних підсистем:

– підсистема користувацької взаємодії, яка забезпечує реєстрацію, вхід, налаштування профілю, введення прийомів їжі та перегляд результатів через зручний інтерфейс;

– підсистема керування профілем, що відповідає за зберігання персональних параметрів користувача, рівня активності, харчових цілей і добових норм;

– підсистема обліку харчування, яка забезпечує додавання продуктів до прийомів їжі, визначення маси порції, фіксацію дати й типу прийому;

– підсистема бази продуктів, що містить довідкову інформацію про калорійність, білки, жири, вуглеводи та одиниці вимірювання продуктів;

– підсистема розрахунків, яка автоматично визначає калорійність і БЖВ окремого продукту, прийому їжі та добового раціону;

– підсистема аналітики та звітності, що формує щоденні підсумки, статистику за тиждень або місяць, показники прогресу та відхилення від добових норм;

– підсистема адміністрування, яка забезпечує керування користувачами, оновлення бази продуктів і контроль коректності довідкових даних;

– підсистема сповіщень, що інформує користувача про необхідність внесення прийому їжі, перевищення добової норми або досягнення встановленої цілі.

Основною метою постановки завдання є створення програмного забезпечення, яке забезпечує простий, точний і зручний моніторинг харчування користувача. Система повинна зменшити кількість ручних розрахунків, підвищити точність обліку калорій і БЖВ, забезпечити збереження історії харчування та надати користувачу інструменти для аналізу власного раціону.

Результатом виконання поставленого завдання має стати програмний продукт, який забезпечує централізоване збереження даних про користувачів, продукти, прийоми їжі, добові норми та аналітичні показники. Розроблена система повинна підтримувати основні сценарії щоденного використання: створення профілю, встановлення цілей, пошук продуктів, додавання прийомів їжі, автоматичний розрахунок калорійності, перегляд добового підсумку та формування звітів.

Отже, розроблення програмного забезпечення для моніторингу калорій та харчування спрямоване на автоматизацію персонального контролю раціону, підвищення зручності ведення харчового щоденника та створення

інформаційної основи для прийняття обґрунтованих рішень щодо коригування харчування.

2 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Логічна модель даних у вигляді ER-діаграми

Логічна модель даних програмного забезпечення для моніторингу калорій та харчування побудована з урахуванням основних інформаційних об’єктів предметної області: користувачів, профілів харчування, добових цілей, продуктів, прийомів їжі, позицій у прийомах їжі та рекомендацій. Модель відображає структуру даних, необхідну для збереження персональних параметрів користувача, ведення журналу харчування, розрахунку калорійності та формування аналітичних висновків.

На рисунку 2.1 подано ER-діаграму логічної моделі даних системи.

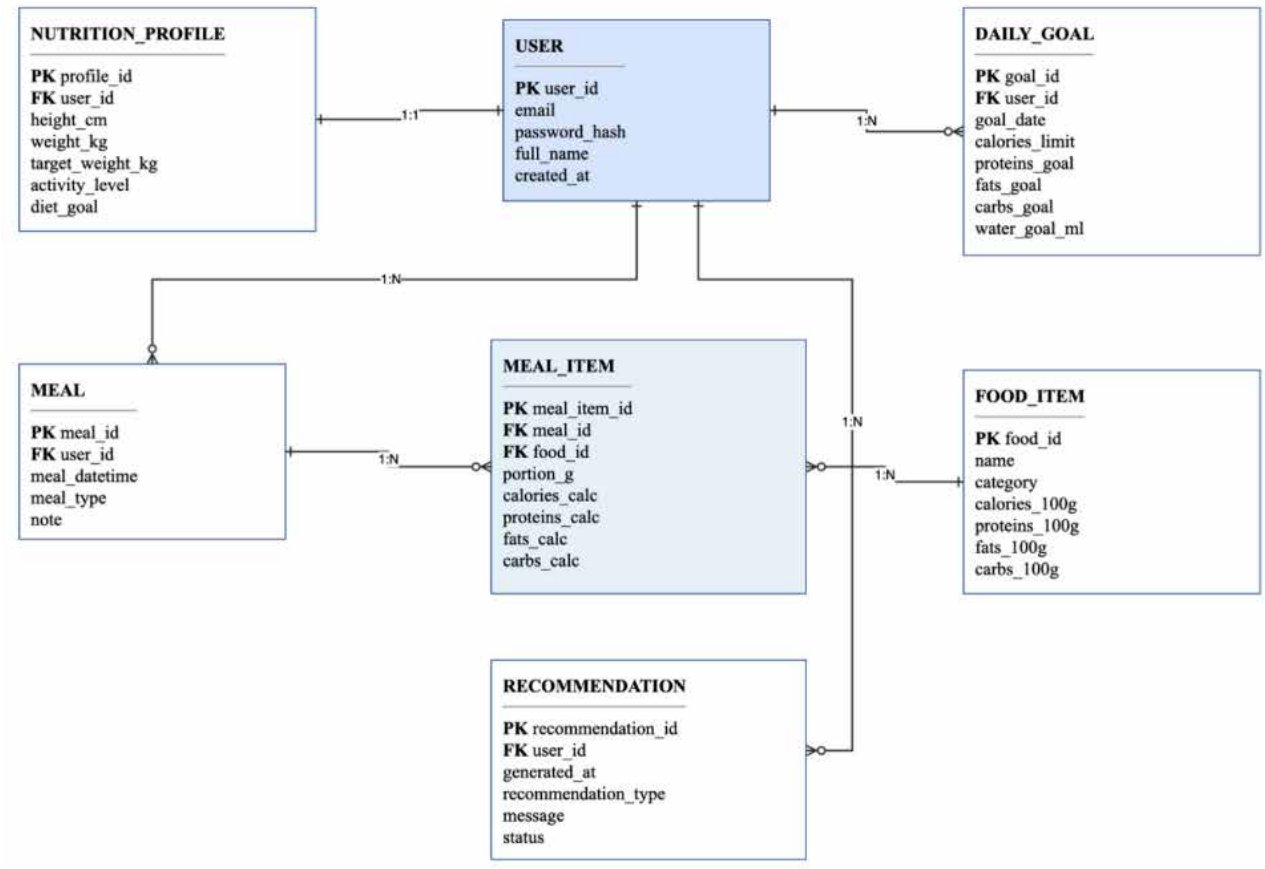

Рисунок 2.1 - Логічна модель даних програмного забезпечення для моніторингу калорій та харчування

Центральною сутністю моделі є USER, яка зберігає облікові дані користувача. З нею пов’язані сутності NUTRITION_PROFILE, DAILY_GOAL, MEAL та RECOMMENDATION. Такий підхід дозволяє відокремити облікову інформацію від параметрів харчування, добових цілей, історії прийомів їжі та сформованих рекомендацій.

Таблиця 2.1 - Основні сутності логічної моделі даних

Сутність	Призначення
USER	Зберігає облікові дані користувача: email, пароль, ім’я, дату створення
NUTRITION_PROFILE	Містить фізичні параметри користувача та ціль харчування
DAILY_GOAL	Зберігає добові норми калорій, БЖВ і води на конкретну дату
FOOD_ITEM	Містить довідкову інформацію про продукти та їх харчову цінність
MEAL	Фіксує прийом їжі користувача із зазначенням дати, часу та типу
MEAL_ITEM	Зберігає конкретні продукти в межах прийому їжі та розраховані показники
RECOMMENDATION	Містить сформовані системою рекомендації для користувача

Окрема сутність MEAL_ITEM використовується для реалізації зв’язку між прийомами їжі та продуктами. Це необхідно, оскільки один прийом їжі може містити багато продуктів, а один і той самий продукт може входити до різних прийомів їжі. Крім цього, у MEAL_ITEM зберігається маса порції та розраховані значення калорій, білків, жирів і вуглеводів.

Логічна модель приведена до третьої нормальної форми. У першій нормальній формі всі атрибути мають атомарні значення, тобто в одному полі не зберігаються списки продуктів, кілька дат або декілька показників

одночасно. У другій нормальній формі всі неключові атрибути залежать від первинного ключа відповідної сутності. Наприклад, назва продукту, категорія та харчова цінність залежать від food_id, а не від користувача або прийому їжі. У третій нормальній формі усунено транзитивні залежності: параметри профілю винесено в NUTRITION_PROFILE, добові цілі - в DAILY_GOAL, а дані про продукти - в FOOD_ITEM.

Нормалізація дозволила уникнути дублювання даних. Наприклад, інформація про харчову цінність продукту зберігається один раз у FOOD_ITEM, а в MEAL_ITEM фіксується лише використання цього продукту в конкретному прийомі їжі. Завдяки цьому спрощується оновлення довідника продуктів, зменшується обсяг зайвих даних і забезпечується цілісність записів.

Отже, запропонована логічна модель даних забезпечує структуроване збереження інформації про користувачів, продукти, прийоми їжі, добові цілі та рекомендації. Вона є основою для подальшого створення фізичної моделі бази даних і реалізації програмних модулів системи.

2.2 Діаграма класів і кооперації

Діаграма класів використовується для опису статичної структури програмного забезпечення для моніторингу калорій та харчування. Вона визначає основні класи системи, їхні атрибути, операції та зв’язки між об’єктами. Побудована модель узгоджується з логічною моделлю даних і відображає основні функціональні блоки: керування користувачами, профілем, цілями харчування, прийомами їжі, продуктами, розрахунками та звітністю.

На рисунку 2.2 подано діаграму класів системи.

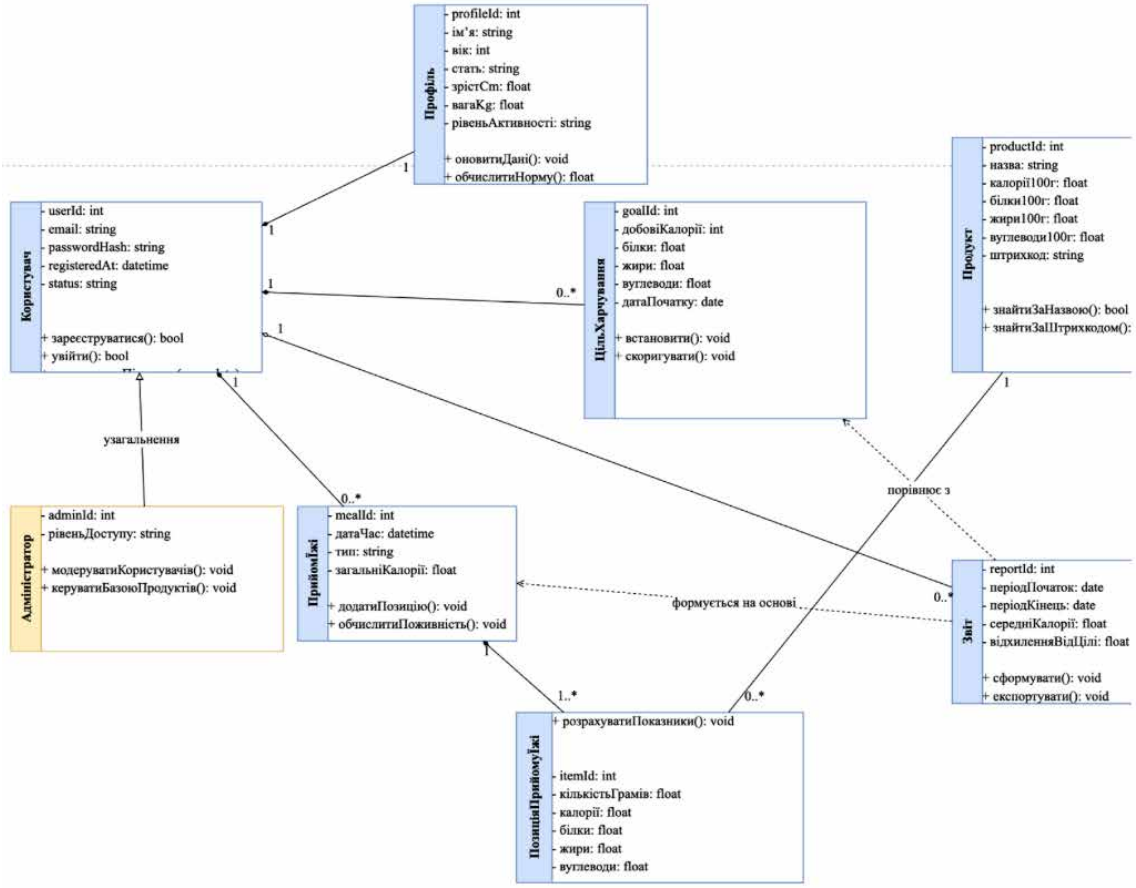

Рисунок 2.2 - Діаграма класів програмного забезпечення для моніторингу калорій та харчування

Основним класом є Користувач, який містить облікові дані, статус і методи реєстрації та входу до системи. З ним пов’язані класи Профіль, ЦільХарчування, ПрийомЇжі та Звіт. Клас Адміністратор є спеціалізованим типом користувача та реалізує операції модерації користувачів і керування базою продуктів.

Клас Профіль зберігає персональні параметри користувача: ім’я, вік, стать, зріст, вагу та рівень активності. Ці дані використовуються для обчислення індивідуальної норми калорій і БЖВ. Клас ЦільХарчування містить добові цільові показники: калорії, білки, жири, вуглеводи та дату початку дії цілі.

Клас ПрийомЇжі відповідає за фіксацію конкретного прийому їжі, його дату, тип і загальну калорійність. Для деталізації складу прийому використовується клас ПозиціяПрийомуЇжі, який пов’язує прийом їжі з

конкретним продуктом і містить кількість грамів, калорії, білки, жири та вуглеводи. Клас Продукт зберігає довідкові дані про харчову цінність продукту на 100 г і підтримує пошук за назвою або штрихкодом. Клас Звіт використовується для формування аналітики за вибраний період.

Таблиця 2.2 - Основні класи системи

Клас	Призначення
Користувач	Зберігає облікові дані та забезпечує доступ до системи
Адміністратор	Виконує модерацію користувачів і керування базою продуктів
Профіль	Містить персональні параметри користувача
ЦільХарчування	Зберігає добові норми калорій і БЖВ
ПрийомЇжі	Фіксує дату, тип і загальні показники прийому їжі
ПозиціяПрийомуЇжі	Зберігає продукт, порцію та розраховані харчові показники
Продукт	Містить назву, категорію, калорійність і БЖВ продукту
Звіт	Формує підсумкову аналітику за період

Між класами встановлено такі основні зв’язки: один користувач має один профіль, може мати декілька цілей харчування, багато прийомів їжі та багато звітів. Один прийом їжі складається з однієї або кількох позицій, а кожна позиція пов’язана з конкретним продуктом. Така структура дозволяє уникнути дублювання інформації про продукти та забезпечує коректний розрахунок харчової цінності кожного прийому їжі.

Для уточнення поведінки системи побудовано діаграми кооперації, які показують взаємодію об’єктів під час виконання основних сценаріїв роботи.

На рисунку 2.3 наведено кооперацію «Запис прийому їжі».

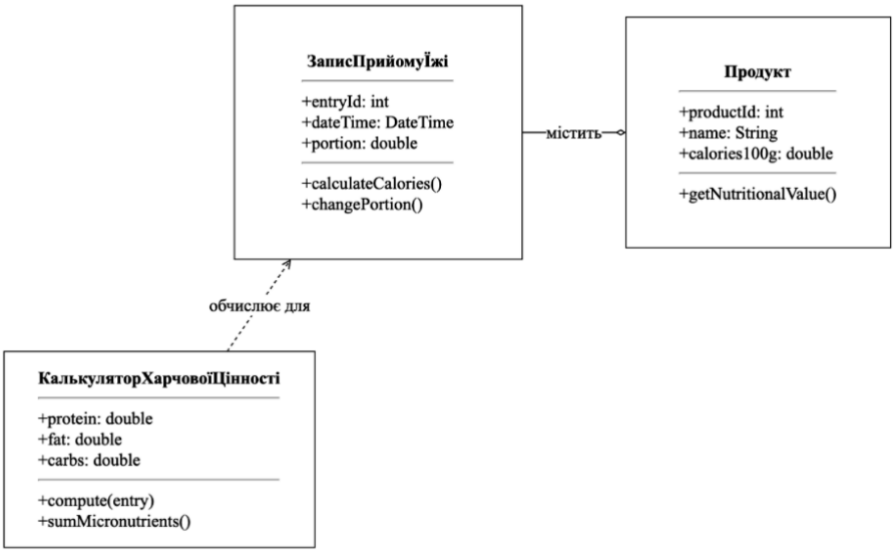

Рисунок 2.3 - Кооперація «Запис прийому їжі»

У цій кооперації об’єкт ЗаписПрийомуЇжі взаємодіє з об’єктом Продукт і КалькуляторХарчовоїЦінності. Користувач додає продукт до прийому їжі, задає розмір порції, після чого калькулятор виконує обчислення калорійності, білків, жирів і вуглеводів. Результат зберігається в записі прийому їжі та використовується для оновлення добового підсумку.

На рисунку 2.4 подано кооперацію «Планування харчування».

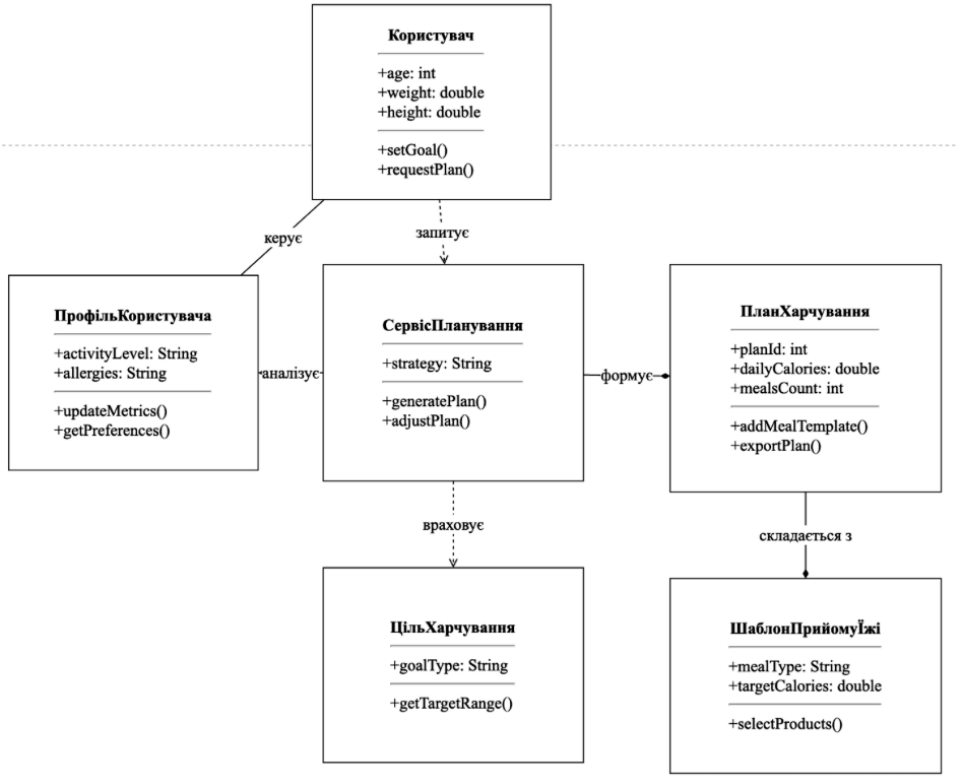

Рисунок 2.4 - Кооперація «Планування харчування»

Кооперація описує взаємодію користувача, профілю, сервісу планування, цілі харчування, плану харчування та шаблонів прийомів їжі. Користувач задає параметри та ціль, сервіс планування аналізує профіль і формує план харчування. План складається з шаблонів прийомів їжі, у яких можуть бути визначені тип прийому, цільова калорійність і перелік рекомендованих продуктів.

На рисунку 2.5 наведено кооперацію «Рекомендаційна система».

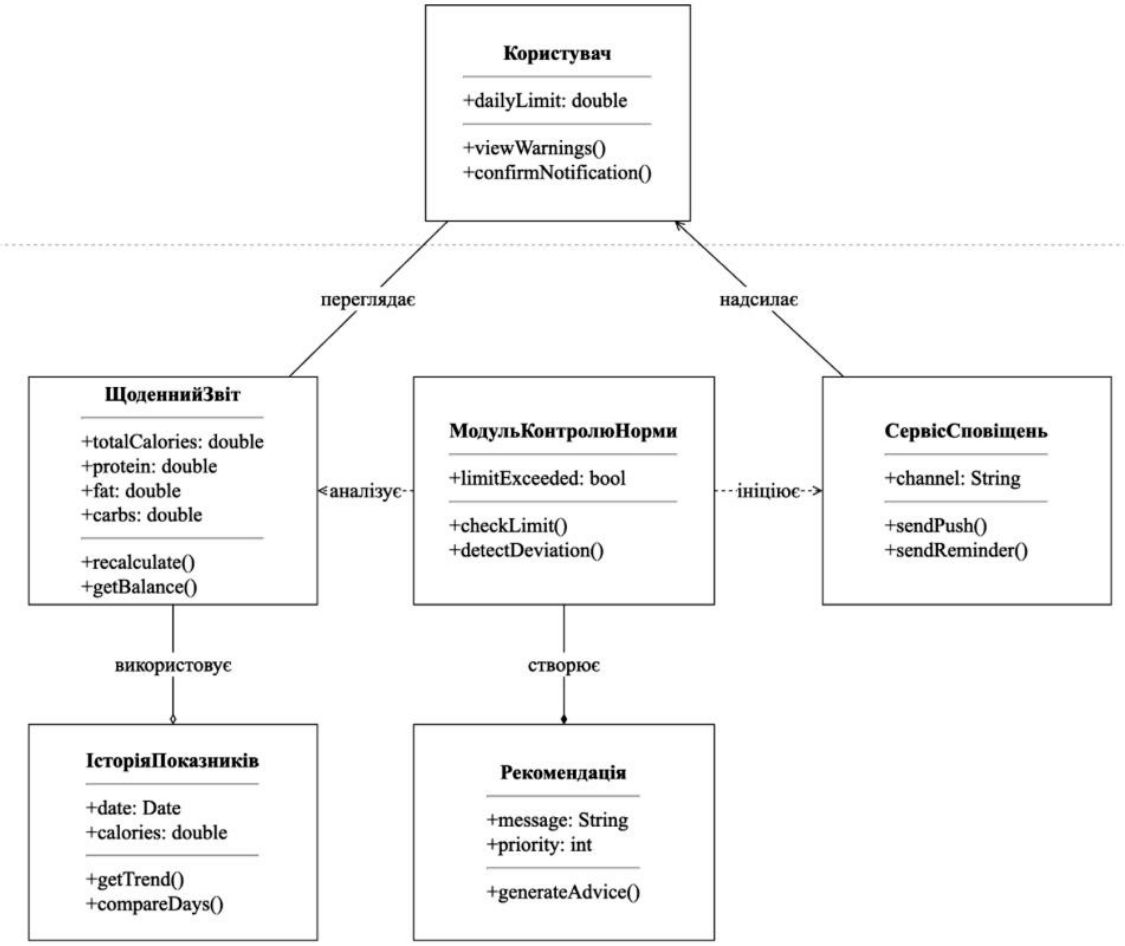

Рисунок 2.5 - Кооперація «Рекомендаційна система»

У цій кооперації користувач переглядає щоденний звіт, а модуль контролю норми аналізує фактичні показники харчування та порівнює їх із добовим лімітом. Якщо система виявляє перевищення або відхилення, створюється рекомендація та ініціюється сервіс сповіщень. Це дозволяє повідомити користувача про необхідність коригування раціону або нагадати про регулярне внесення даних.

Таблиця 2.3 - Призначення діаграм кооперації

Кооперація	Основні об’єкти	Призначення
Запис прийому їжі	ЗаписПрийомуЇжі, Продукт, КалькуляторХарчовоїЦінності	Розрахунок харчової цінності прийому їжі
Планування харчування	Користувач, ПрофільКористувача, СервісПланування, ПланХарчування	Формування плану відповідно до параметрів і цілі
Рекомендаційна система	ЩоденнийЗвіт, МодульКонтролюНорми, Рекомендація, СервісСповіщень	Виявлення відхилень і формування повідомлень

Отже, діаграма класів визначає основну структуру програмної системи, а діаграми кооперації деталізують взаємодію об’єктів під час виконання ключових сценаріїв. Побудовані моделі підтверджують, що система має модульну структуру, у якій окремо виділено облік користувачів, профіль, харчові цілі, продукти, прийоми їжі, розрахунки, звіти та рекомендації. Це спрощує подальшу реалізацію програмного забезпечення та забезпечує можливість його розширення.

2.3 Діаграма компонентів розроблювальної системи

Діаграма компонентів використовується для відображення програмної архітектури системи моніторингу калорій та харчування. Вона показує основні програмні модулі, їхні інтерфейси взаємодії, бази даних і зовнішні сервіси, які забезпечують роботу системи.

На рисунку 2.6 наведено діаграму компонентів програмного забезпечення для моніторингу калорій та харчування.

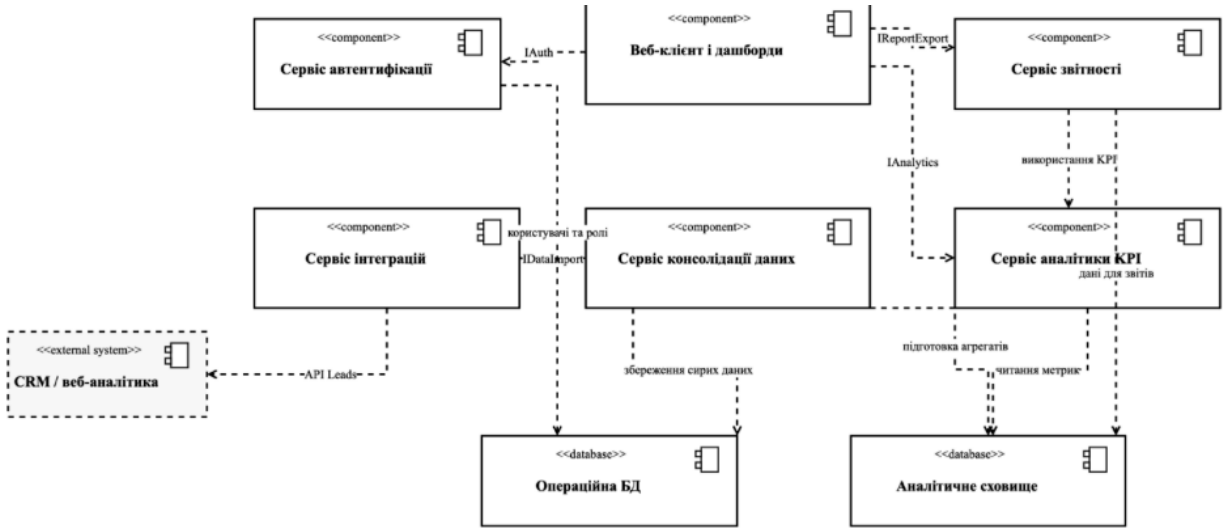

Рисунок 2.6 - Діаграма компонентів програмного забезпечення для моніторингу калорій та харчування

Центральним компонентом системи є веб-клієнт і дашборди, через які користувач взаємодіє з основними функціями програмного забезпечення. Через інтерфейс користувач може виконати вхід до системи, налаштувати профіль, додати прийом їжі, переглянути добовий підсумок, звіти та аналітичні показники.

Сервіс автентифікації відповідає за реєстрацію, вхід до системи, перевірку облікових даних і розмежування ролей користувача та адміністратора. Він передає дані про користувачів і ролі до операційної бази даних, де зберігаються облікові записи, профілі та параметри доступу.

Сервіс інтеграції забезпечує взаємодію із зовнішніми джерелами даних, зокрема базами харчової цінності або сервісами сканування штрихкодів. Через цей компонент система може отримувати дані про продукти, їх калорійність, білки, жири та вуглеводи.

Сервіс консолідації даних виконує первинне оброблення отриманої інформації. Він узгоджує дані про користувача, продукти, прийоми їжі та передає їх до операційної бази даних. У цій базі зберігаються профілі користувачів, цілі харчування, продукти, записи прийомів їжі та розраховані показники.

Окремим компонентом є сервіс аналітики, який обробляє записи харчування та формує добові й періодичні підсумки. Він розраховує спожиті калорії, білки, жири, вуглеводи, визначає залишок або перевищення добової норми та готує агреговані показники для звітів. Результати аналітичної обробки можуть зберігатися в аналітичному сховищі для подальшого перегляду статистики за тиждень або місяць.

Сервіс звітності використовує підготовлені аналітичні дані та формує звіти для користувача. Через цей компонент користувач отримує інформацію про динаміку харчування, виконання добових цілей, середні показники за період і рекомендації щодо коригування раціону.

У структурі системи також передбачено сервіс сповіщень, який може надсилати користувачу повідомлення про необхідність внесення прийому їжі, перевищення добової норми або досягнення встановленої цілі. Це підвищує практичну цінність системи та підтримує регулярне ведення харчового щоденника.

Компонентна структура є модульною, тому кожен блок виконує окрему функцію та може доопрацьовуватися незалежно від інших. Такий підхід спрощує супровід програмного забезпечення, дозволяє розширювати функціональність системи та забезпечує логічний поділ між інтерфейсом, бізнес-логікою, обробленням даних, аналітикою і збереженням інформації.

Отже, діаграма компонентів показує, що програмне забезпечення для моніторингу калорій та харчування будується як набір взаємопов’язаних модулів, які забезпечують автентифікацію користувачів, облік продуктів і прийомів їжі, розрахунок калорійності, формування аналітики, звітності та сповіщень. Така архітектура є достатньою для реалізації основного функціоналу системи та подальшого її розвитку.

2.4 Діаграма пакетів

Діаграма пакетів відображає логічну організацію програмного забезпечення для моніторингу калорій та харчування. Вона показує, з яких основних модулів складається система та як між ними передаються запити й дані.

На рисунку 2.7 подано діаграму пакетів системи.

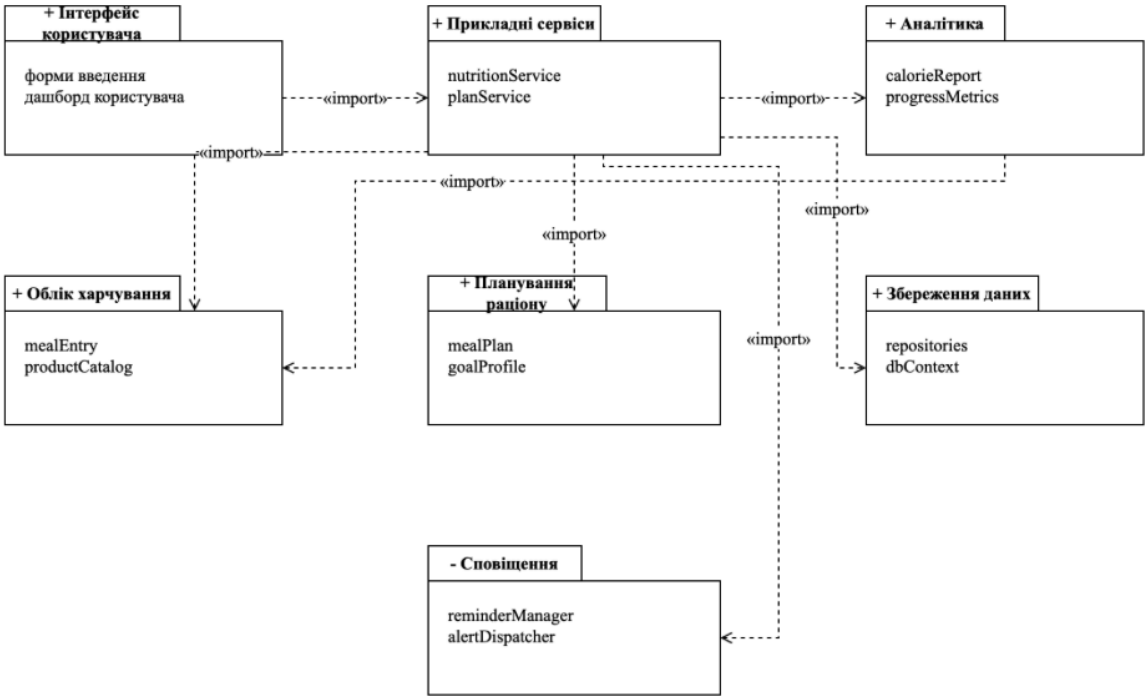

Рисунок 2.7 - Діаграма пакетів програмного забезпечення для моніторингу калорій та харчування

У системі виділено сім основних пакетів: Інтерфейс користувача, Прикладні сервіси, Облік харчування, Планування раціону, Аналітика, Збереження даних і Сповіщення. Центральним пакетом є Прикладні сервіси, оскільки саме він координує роботу інших частин системи.

Таблиця 2.4 - Основні пакети програмної системи

Пакет	Складові	Призначення
Інтерфейс користувача	форми введення, дашборд користувача	Забезпечує взаємодію користувача із системою
Прикладні сервіси	nutritionService, planService	Обробляє основні запити та координує роботу модулів

Продовження таблиці 2.4

Облік харчування	mealEntry, productCatalog	Відповідає за прийоми їжі та базу продуктів
Планування раціону	mealPlan, goalProfile	Зберігає цілі харчування та добові норми
Аналітика	calorieReport, progressMetrics	Формує звіти, статистику та показники прогресу
Збереження даних	repositories, dbContext	Забезпечує доступ до бази даних
Сповіщення	reminderManager, alertDispatcher	Формує нагадування та повідомлення користувачу

Отже, пакетна структура забезпечує поділ системи на окремі логічні модулі. Такий підхід спрощує розроблення, тестування та подальше розширення програмного забезпечення.

2.5 Висновки до другого розділу

У другому розділі було виконано проєктування інформаційного та програмного забезпечення системи моніторингу калорій та харчування. На основі результатів аналізу предметної області визначено основні інформаційні об’єкти, програмні модулі та взаємозв’язки між ними.

Розроблено логічну модель даних у вигляді ER-діаграми, яка включає сутності користувача, профілю харчування, добових цілей, продуктів, прийомів їжі, позицій прийому та рекомендацій. Модель приведено до третьої нормальної форми, що дозволяє уникнути дублювання даних, забезпечити цілісність інформації та створити основу для подальшої реалізації фізичної бази даних.

Побудовано діаграму класів, яка відображає статичну структуру програмної системи. У ній виділено класи користувача, адміністратора, профілю, цілі харчування, продукту, прийому їжі, позиції прийому та звіту. Також розроблено кооперації для основних сценаріїв роботи системи: запису прийому їжі, планування харчування та формування рекомендацій. Це дозволило уточнити взаємодію об’єктів під час виконання ключових функцій.

Розроблено діаграму компонентів, яка показує архітектуру програмного забезпечення. У структурі системи виділено веб-клієнт, сервіс автентифікації, сервіс інтеграції, сервіс консолідації даних, сервіс аналітики, сервіс звітності, операційну базу даних і аналітичне сховище. Такий поділ забезпечує модульність системи та спрощує подальший розвиток програмного продукту.

Також побудовано діаграму пакетів, яка відображає логічну організацію програмних модулів. Система поділена на пакети інтерфейсу користувача, прикладних сервісів, обліку харчування, планування раціону, аналітики, збереження даних і сповіщень. Така структура дозволяє відокремити бізнес-логіку від інтерфейсу та роботи з базою даних.

Отже, у другому розділі сформовано проєктну основу програмного забезпечення для моніторингу калорій та харчування. Розроблені моделі визначають структуру даних, логіку взаємодії класів, компонентну архітектуру та пакетну організацію системи, що є необхідною базою для подальшої реалізації програмного продукту.

3 ПРОЄКТУВАННЯ АРХІТЕКТУРИ ПРОГРАМНОЇ СИСТЕМИ ТА РОЗРОБЛЕННЯ АЛГОРИТМІЧНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Обґрунтування вибору технологій для розроблення програмної системи

Для реалізації програмного забезпечення для моніторингу калорій та харчування доцільно використати стек технологій, який забезпечує швидке розроблення, зручну роботу з даними, можливість створення зрозумілого інтерфейсу та подальше розширення системи. Оскільки система передбачає роботу з користувачами, профілями, базою продуктів, прийомами їжі, аналітикою та звітами, обрані технології повинні підтримувати клієнт-серверну архітектуру, оброблення запитів і збереження структурованих даних.

У межах проєкту програмну систему доцільно реалізувати як веб-орієнтований застосунок. Такий підхід дозволяє користувачу працювати із системою через браузер без встановлення складного додаткового програмного забезпечення. Веб-інтерфейс забезпечує доступ до основних функцій: реєстрації, налаштування профілю, додавання прийомів їжі, перегляду добового підсумку, статистики та рекомендацій.

Для клієнтської частини обрано HTML, CSS і JavaScript, оскільки ці технології є базовими для створення веб-інтерфейсів. HTML відповідає за структуру сторінок, CSS використовується для оформлення елементів інтерфейсу, а JavaScript забезпечує інтерактивність, оброблення дій користувача та динамічне оновлення даних на сторінці.

Серверну частину доцільно реалізувати з використанням Python та фреймворку Flask або FastAPI. Python має простий синтаксис, велику кількість бібліотек і добре підходить для створення навчальних програмних систем. Flask є зручним для невеликих і середніх веб-застосунків, а FastAPI краще підходить для REST API та структурованої клієнт-серверної взаємодії. У межах роботи

доцільно використати FastAPI, оскільки система передбачає окремі API-запити для роботи з користувачами, продуктами, прийомами їжі, цілями та звітами.

Для збереження даних обрано SQLite як просту локальну реляційну базу даних. Вона не потребує складного адміністрування, зберігає дані в одному файлі та добре підходить для прототипу системи. У разі подальшого масштабування систему можна перенести на PostgreSQL без суттєвої зміни логіки роботи з даними.

Основні технології реалізації системи наведено в таблиці 3.1.

Таблиця 3.1 - Обраний стек технологій програмної системи

Технологія	Призначення	Обґрунтування вибору
HTML	Структура веб-сторінок	Забезпечує побудову форм, таблиць, дашбордів і сторінок системи
CSS	Оформлення інтерфейсу	Дозволяє створити зручний і візуально зрозумілий дизайн
JavaScript	Інтерактивність клієнтської частини	Забезпечує оброблення дій користувача та динамічне оновлення даних
Python	Основна мова серверної частини	Має простий синтаксис і достатній набір бібліотек для веб-розробки
FastAPI	Серверний фреймворк	Підходить для створення REST API та взаємодії між клієнтом і сервером
SQLite	База даних	Забезпечує просте збереження користувачів, продуктів, прийомів їжі та звітів
SQLAlchemy	Робота з БД	Дозволяє зручно описувати таблиці та виконувати запити до бази даних
JSON	Формат обміну даними	Використовується для передавання даних між клієнтом і сервером
Git	Контроль версій	Дозволяє зберігати історію змін програмного коду

Архітектура програмної системи передбачає поділ на клієнтську частину, серверну логіку та базу даних. Клієнтська частина відповідає за відображення інтерфейсу та прийом дій користувача. Серверна частина обробляє запити, виконує перевірку даних, розрахунок калорій і БЖВ, формує статистику та передає результати назад у клієнтську частину. База даних забезпечує збереження облікових записів, профілів, продуктів, прийомів їжі, добових цілей і рекомендацій.

Для реалізації основних функцій системи передбачається створення таких програмних модулів:

- модуль автентифікації користувачів;
- модуль керування профілем;
- модуль бази продуктів;
- модуль додавання прийомів їжі;
- модуль розрахунку калорійності та БЖВ;
- модуль аналітики та звітності;
- модуль адміністрування;
- модуль сповіщень і рекомендацій.

Вибраний стек технологій відповідає вимогам до програмної системи, оскільки забезпечує модульність, простоту реалізації, зручне збереження даних і можливість подальшого розширення. За потреби систему можна доповнити авторизацією через JWT, підключенням зовнішньої бази харчової цінності, скануванням штрихкодів, мобільною версією інтерфейсу або переходом на повноцінну серверну СУБД.

Отже, для реалізації програмного забезпечення для моніторингу калорій та харчування обрано веб-орієнтований стек на основі HTML, CSS, JavaScript, Python, FastAPI та SQLite. Такий набір технологій є достатнім для створення функціонального прототипу системи, забезпечує зрозумілу архітектуру та дозволяє реалізувати основні можливості: облік користувачів, ведення харчового щоденника, розрахунок калорійності, контроль добових норм і формування аналітики.

3.2 Реалізація фізичної моделі бази даних програмної системи

Для збереження даних програмного забезпечення для моніторингу калорій та харчування спроєктовано фізичну модель бази даних. Вона побудована на основі логічної ER-моделі та враховує основні процеси системи: реєстрацію користувачів, ведення профілю, встановлення добових цілей,

роботу з базою продуктів, додавання прийомів їжі, розрахунок харчових показників і формування рекомендацій.

На рисунку 3.1 подано фізичну модель бази даних системи.

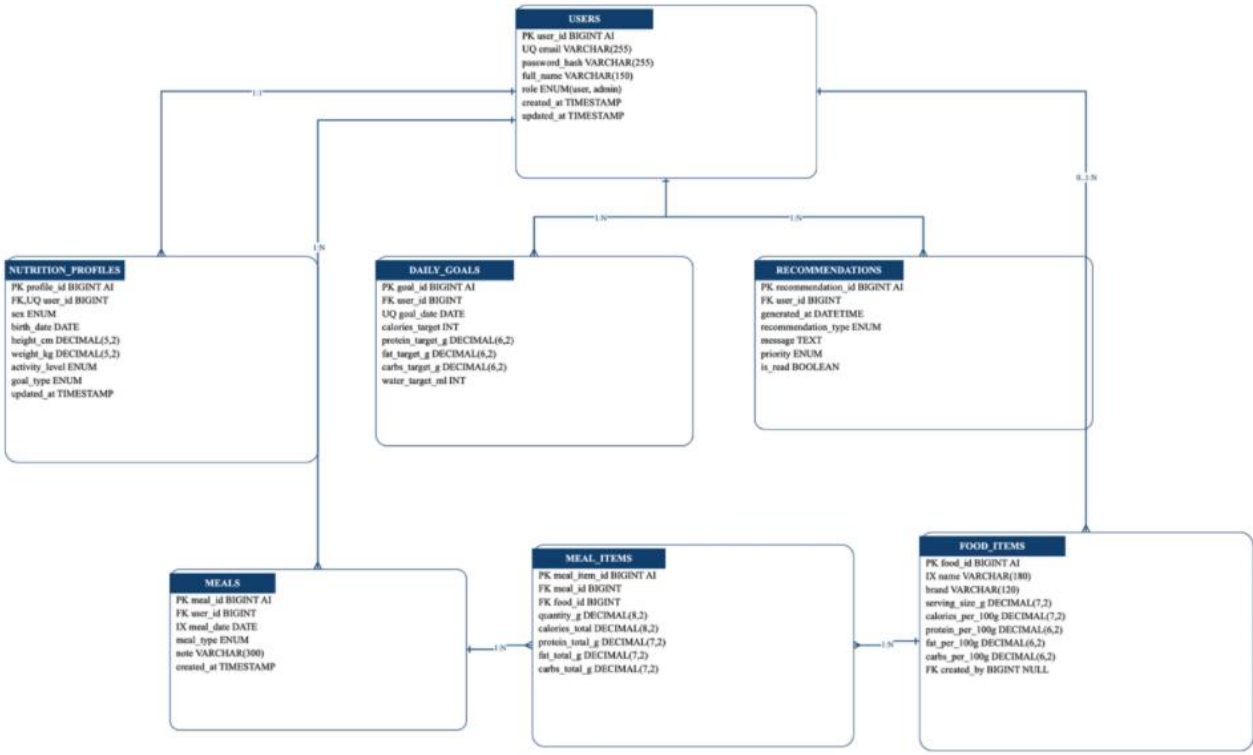

Рисунок 3.1 - Фізична модель бази даних програмного забезпечення для моніторингу калорій та харчування

Фізична модель містить сім основних таблиць: USERS, NUTRITION_PROFILES, DAILY_GOALS, FOOD_ITEMS, MEALS, MEAL_ITEMS та RECOMMENDATIONS. Центральною таблицею є USERS, оскільки всі персональні дані, прийоми їжі, цілі та рекомендації прив’язуються до конкретного користувача.

Таблиця 3.2 - Основні таблиці фізичної моделі бази даних

Таблиця	Призначення
USERS	Зберігає облікові записи користувачів і ролі доступу
NUTRITION_PROFILES	Містить персональні параметри користувача для розрахунку норм
DAILY_GOALS	Зберігає добові цілі за калоріями, БЖВ і водою
FOOD_ITEMS	Містить довідник продуктів та їх харчову цінність
MEALS	Фіксує прийоми їжі користувача за датою і типом
MEAL_ITEMS	Зберігає продукти, додані до конкретного прийому їжі
RECOMMENDATIONS	Містить рекомендації, сформовані системою

У таблиці USERS зберігаються email, хеш пароля, ім’я користувача, роль і службові дати створення та оновлення запису. Для поля email передбачено унікальне обмеження, що унеможливлює створення кількох акаунтів з однаковою поштою.

Таблиця NUTRITION_PROFILES реалізує зв’язок один до одного з користувачем. У ній зберігаються стать, дата народження, зріст, вага, рівень активності та тип харчової цілі. Таблиця DAILY_GOALS має зв’язок один до багатьох із USERS, оскільки користувач може мати окремі добові цілі на різні дати.

Таблиці MEALS і MEAL_ITEMS реалізують журнал харчування. Один прийом їжі може містити багато продуктів, тому для деталізації використано таблицю MEAL_ITEMS. У ній зберігається кількість продукту в грамах і розраховані значення калорій, білків, жирів і вуглеводів для конкретної порції.

Таблиця FOOD_ITEMS є довідником продуктів. Вона містить назву, бренд, розмір порції, калорійність і показники БЖВ на 100 г. Поле barcode може використовуватися для пошуку продукту за штрихкодом. Таблиця RECOMMENDATIONS призначена для збереження повідомлень і рекомендацій, які формуються на основі аналізу харчування користувача.

Для реалізації бази даних може бути використаний такий SQL-код:

```
CREATE TABLE users (
    user_id BIGINT PRIMARY KEY AUTO_INCREMENT,
    email VARCHAR(255) NOT NULL UNIQUE,
    password_hash VARCHAR(255) NOT NULL,
    full_name VARCHAR(150) NOT NULL,
    role ENUM('user', 'admin') NOT NULL DEFAULT 'user',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
        ON UPDATE CURRENT_TIMESTAMP
);

CREATE TABLE nutrition_profiles (
    profile_id BIGINT PRIMARY KEY AUTO_INCREMENT,
    user_id BIGINT NOT NULL UNIQUE,
    sex ENUM('male', 'female') NOT NULL,
    birth_date DATE NOT NULL,
    height_cm DECIMAL(5,2) NOT NULL,
    weight_kg DECIMAL(5,2) NOT NULL,
    activity_level ENUM('low', 'medium', 'high') NOT NULL,
    goal_type ENUM('lose_weight', 'maintain_weight', 'gain_weight') NOT NULL,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
        ON UPDATE CURRENT_TIMESTAMP,
```

```
    FOREIGN KEY (user_id) REFERENCES users(user_id)
        ON DELETE CASCADE
);

CREATE TABLE daily_goals (
    goal_id BIGINT PRIMARY KEY AUTO_INCREMENT,
    user_id BIGINT NOT NULL,
    goal_date DATE NOT NULL,
    calories_target INT NOT NULL,
    protein_target_g DECIMAL(6,2) NOT NULL,
    fat_target_g DECIMAL(6,2) NOT NULL,
    carbs_target_g DECIMAL(6,2) NOT NULL,
    water_target_ml INT NOT NULL,
    UNIQUE (user_id, goal_date),
    FOREIGN KEY (user_id) REFERENCES users(user_id)
        ON DELETE CASCADE
);

CREATE TABLE food_items (
    food_id BIGINT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(180) NOT NULL,
    brand VARCHAR(120),
    serving_size_g DECIMAL(7,2) DEFAULT 100.00,
    calories_per_100g DECIMAL(7,2) NOT NULL,
    protein_per_100g DECIMAL(6,2) NOT NULL,
    fat_per_100g DECIMAL(6,2) NOT NULL,
    carbs_per_100g DECIMAL(6,2) NOT NULL,
    barcode VARCHAR(64) UNIQUE,
    created_by BIGINT,
    FOREIGN KEY (created_by) REFERENCES users(user_id)
        ON DELETE SET NULL
);

CREATE TABLE meals (
    meal_id BIGINT PRIMARY KEY AUTO_INCREMENT,
    user_id BIGINT NOT NULL,
    meal_datetime DATETIME NOT NULL,
    meal_type ENUM('breakfast', 'lunch', 'dinner', 'snack') NOT NULL,
    note VARCHAR(300),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(user_id)
        ON DELETE CASCADE
);

CREATE TABLE meal_items (
    meal_item_id BIGINT PRIMARY KEY AUTO_INCREMENT,
    meal_id BIGINT NOT NULL,
    food_id BIGINT NOT NULL,
    quantity_g DECIMAL(8,2) NOT NULL,
    calories_total DECIMAL(8,2) NOT NULL,
    protein_total_g DECIMAL(7,2) NOT NULL,
    fat_total_g DECIMAL(7,2) NOT NULL,
    carbs_total_g DECIMAL(7,2) NOT NULL,
    FOREIGN KEY (meal_id) REFERENCES meals(meal_id)
        ON DELETE CASCADE,
    FOREIGN KEY (food_id) REFERENCES food_items(food_id)
        ON DELETE RESTRICT
);

CREATE TABLE recommendations (
    recommendation_id BIGINT PRIMARY KEY AUTO_INCREMENT,
    user_id BIGINT NOT NULL,
    generated_at DATETIME NOT NULL,
```

```
    recommendation_type ENUM('calories', 'protein', 'fat', 'carbs', 'water',
'general') NOT NULL,
    message TEXT NOT NULL,
    priority ENUM('low', 'medium', 'high') NOT NULL DEFAULT 'medium',
    is_read BOOLEAN NOT NULL DEFAULT FALSE,
    FOREIGN KEY (user_id) REFERENCES users(user_id)
        ON DELETE CASCADE
);
```

Для прискорення пошуку доцільно створити індекси за полями, які найчастіше використовуються у запитах: ідентифікатор користувача, дата прийому їжі, назва продукту та дата добової цілі.

```
CREATE INDEX idx_meals_user_datetime
ON meals(user_id, meal_datetime);

CREATE INDEX idx_daily_goals_user_date
ON daily_goals(user_id, goal_date);

CREATE INDEX idx_food_items_name
ON food_items(name);

CREATE INDEX idx_recommendations_user_status
ON recommendations(user_id, is_read);
```

Запропонована структура бази даних відповідає третій нормальній формі. Дані про користувача, профіль, продукти, прийоми їжі та рекомендації зберігаються в окремих таблицях, що усуває дублювання та спрощує оновлення інформації. Наприклад, харчова цінність продукту зберігається тільки в таблиці `FOOD_ITEMS`, а в таблиці `MEAL_ITEMS` фіксуються лише порція та розраховані значення для конкретного прийому їжі.

Отже, фізична модель бази даних забезпечує надійне збереження інформації, підтримує зв’язки між основними сутностями системи та створює основу для реалізації функцій обліку харчування, розрахунку калорійності, формування добових підсумків і рекомендацій користувачу.

3.3 Алгоритмізація програмних модулів системи

Алгоритмізація програмних модулів виконана для визначення послідовності роботи основних функцій програмного забезпечення для моніторингу калорій та харчування. Розроблені блок-схеми описують логіку реєстрації користувача, налаштування профілю, додавання прийому їжі, планування раціону та формування аналітичного звіту.

На рисунку 3.2 подано блок-схему алгоритму реєстрації, входу та налаштування профілю користувача.

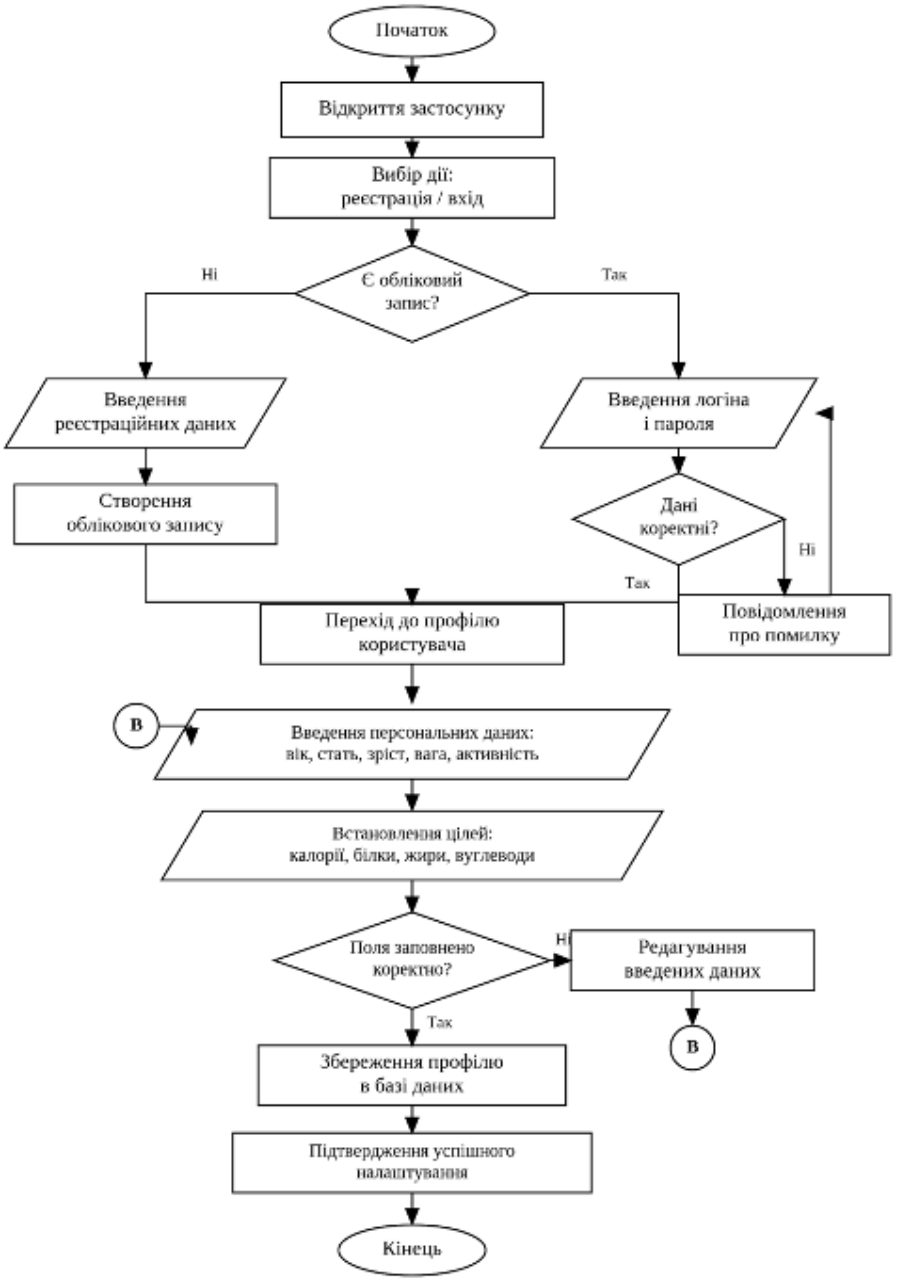

Рисунок 3.2 - Блок-схема алгоритму реєстрації та налаштування профілю користувача

Алгоритм починається з відкриття застосунку та вибору дії: реєстрація нового користувача або вхід до наявного облікового запису. Якщо користувач проходить реєстрацію, система приймає введені дані, створює обліковий запис і переводить користувача до профілю. Якщо користувач обирає вхід, система

перевіряє логін і пароль. У разі помилки виводиться повідомлення, після чого користувач повторно вводить дані.

Після успішної авторизації користувач переходить до налаштування профілю. У цьому модулі вводяться персональні параметри: вік, стать, зріст, маса тіла та рівень активності. Далі користувач встановлює цілі харчування, зокрема добову норму калорій, білків, жирів і вуглеводів. Перед збереженням система перевіряє правильність заповнення полів. Якщо дані введено некоректно, користувач повертається до редагування. Якщо всі поля заповнено правильно, профіль зберігається в базі даних.

На рисунку 3.3 наведено блок-схему алгоритму додавання прийому їжі.

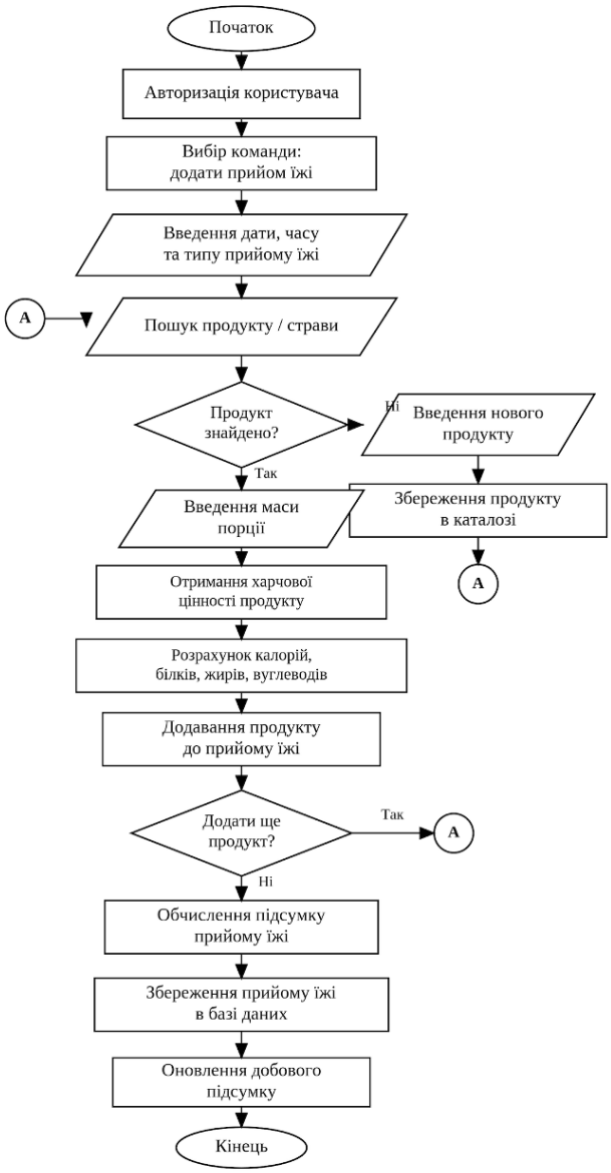

Рисунок 3.3 - Блок-схема алгоритму додавання прийому їжі

Алгоритм додавання прийому їжі реалізує основний сценарій ведення харчового щоденника. Після авторизації користувач обирає команду додавання прийому їжі, вказує дату, час і тип прийому: сніданок, обід, вечеря або перекус. Після цього виконується пошук продукту або страви в каталозі.

Якщо продукт знайдено, користувач вводить масу порції. Система отримує харчову цінність продукту з бази даних і автоматично розраховує калорії, білки, жири та вуглеводи для заданої порції. Якщо продукт відсутній у каталозі, користувач може додати новий продукт, вказавши його назву та харчові показники. Після збереження нового продукту система повертається до етапу пошуку.

Після розрахунку продукт додається до поточного прийому їжі. Якщо користувач хоче додати ще один продукт, алгоритм повторюється. Після завершення введення система обчислює підсумок прийому їжі, зберігає його в базі даних і оновлює добовий підсумок користувача.

На рисунку 3.4 подано блок-схему алгоритму планування раціону.

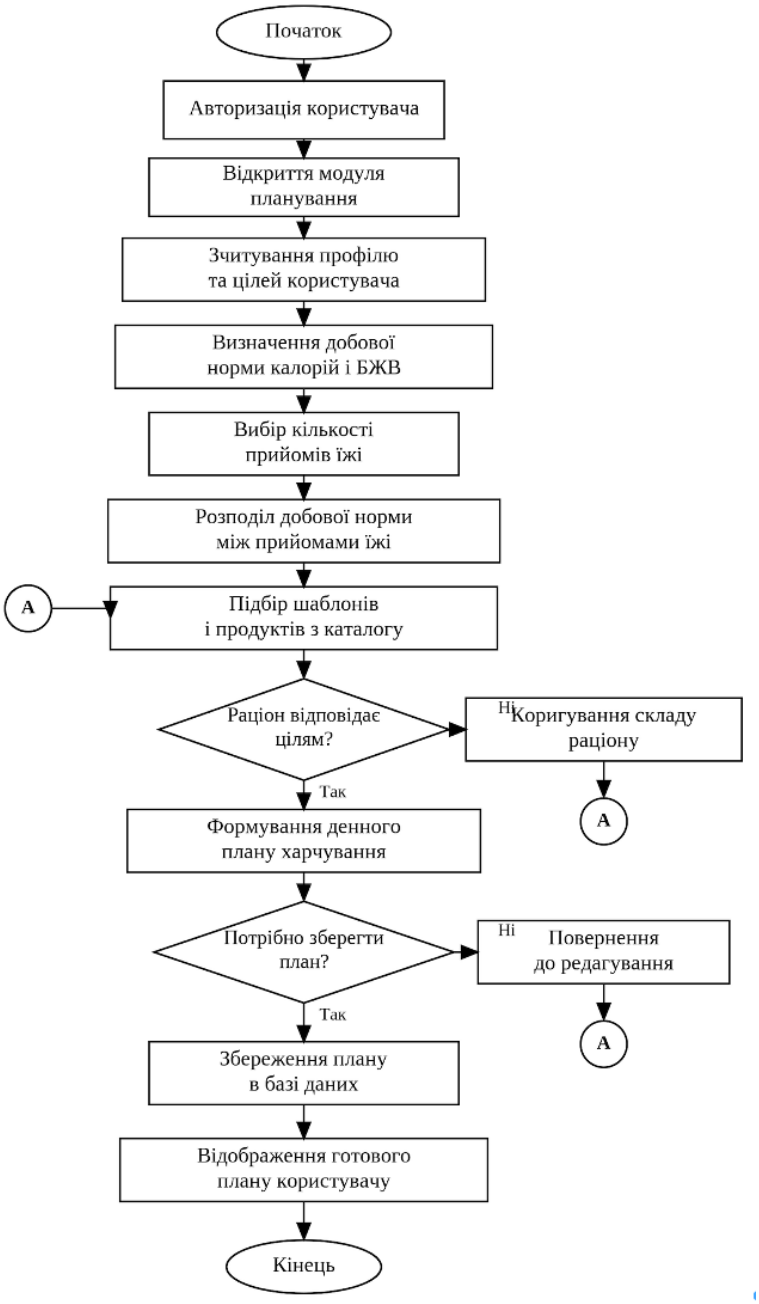

Рисунок 3.4 - Блок-схема алгоритму планування раціону користувача

Алгоритм планування раціону починається з авторизації користувача та відкриття відповідного модуля. Система зчитує профіль користувача і встановлені цілі харчування. На основі цих даних визначається добова норма калорій і БЖВ, після чого користувач обирає кількість прийомів їжі протягом дня.

Далі система розподіляє добову норму між прийомами їжі та підбирає шаблони раціону з урахуванням заданої мети. Користувач може переглянути запропонований план і за потреби змінити склад продуктів. Якщо раціон не

відповідає встановленим цілям, він коригується. Якщо план є прийнятним, система формує готовий денний план харчування.

Після формування плану користувач обирає, чи потрібно зберегти його в базі даних. Якщо план зберігається, він стає доступним для подальшого перегляду та використання. Якщо користувач не підтверджує збереження, система повертає його до етапу редагування.

На рисунку 3.5 наведено блок-схему алгоритму формування аналітичного звіту.

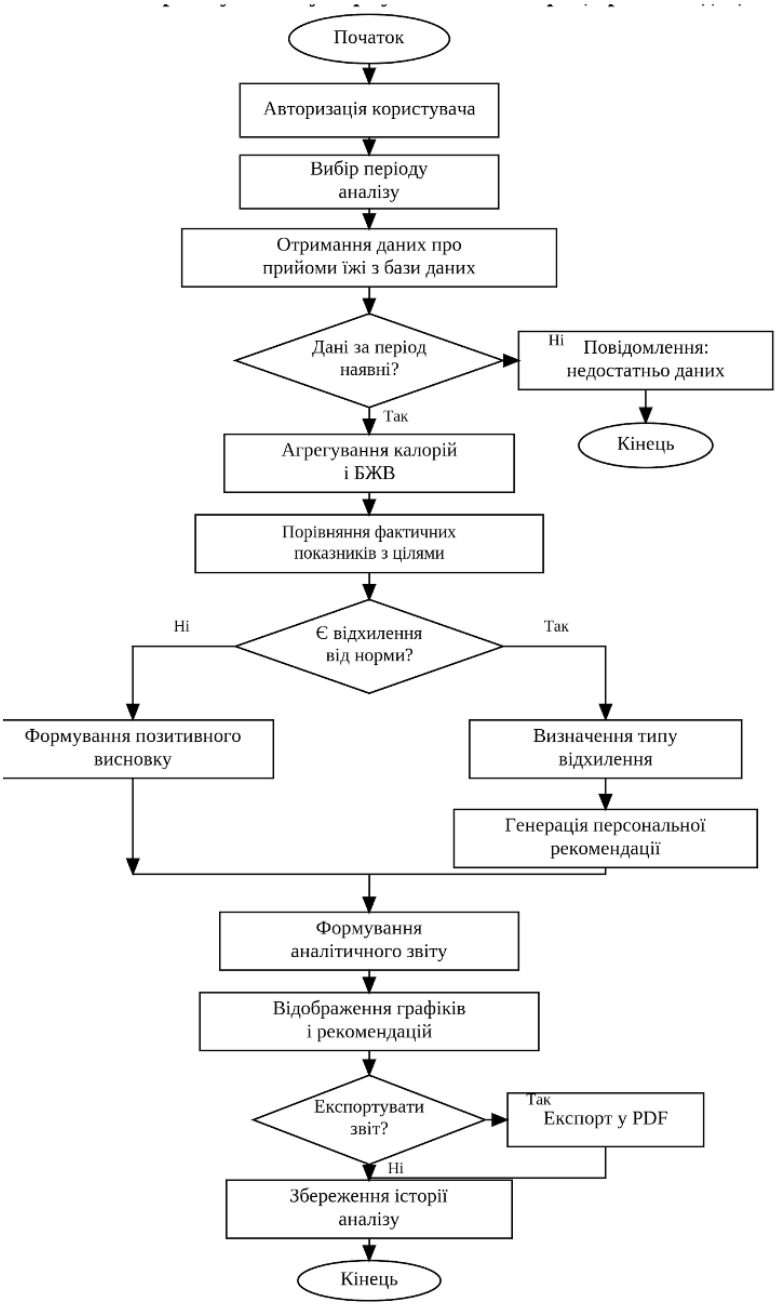

Рисунок 3.5 - Блок-схема алгоритму формування аналітичного звіту

Алгоритм формування аналітичного звіту призначений для аналізу харчування користувача за вибраний період. Після авторизації користувач обирає період аналізу. Система отримує з бази даних записи про прийоми їжі та перевіряє, чи достатньо даних для формування звіту.

Якщо дані за вибраний період відсутні або їх недостатньо, система виводить відповідне повідомлення. Якщо дані наявні, виконується узагальнення інформації про спожиті калорії, білки, жири та вуглеводи. Після цього система порівнює фактичні показники з установленими цілями користувача.

Якщо виявлено відхилення від добової норми, система визначає його тип і формує персональну рекомендацію. Якщо показники відповідають заданим цілям, формується позитивний висновок. Після цього система створює аналітичний звіт, відображає графіки, рекомендації та короткий підсумок. За потреби користувач може експортувати звіт у PDF. Після завершення аналізу результати зберігаються в історії.

Таблиця 3.3 - Алгоритми програмних модулів системи

Алгоритм	Вхідні дані	Основні дії	Результат
Реєстрація та налаштування профілю	Облікові дані, персональні параметри, цілі харчування	Перевірка даних, створення акаунта, заповнення профілю, збереження цілей	Створений профіль користувача
Додавання прийому їжі	Дата, тип прийому, продукт, маса порції	Пошук продукту, введення порції, розрахунок показників, збереження запису	Оновлений журнал харчування
Планування раціону	Профіль, добові цілі, кількість прийомів їжі	Визначення норми, розподіл між прийомами, підбір продуктів, коригування плану	Сформований денний план харчування
Формування аналітичного звіту	Період аналізу, записи харчування, добові цілі	Отримання даних, узагальнення показників, порівняння з цілями, формування рекомендацій	Аналітичний звіт користувача

Отже, алгоритмізація програмних модулів визначає порядок виконання основних функцій системи. Розроблені алгоритми забезпечують реєстрацію користувача, налаштування профілю, додавання прийомів їжі, планування раціону, оновлення добових підсумків і формування аналітичних звітів. Така структура дозволяє реалізувати програмну систему послідовно, без дублювання функцій і з чітким поділом відповідальності між модулями.

3.4 Висновки до третього розділу

У третьому розділі було розглянуто практичні аспекти реалізації програмного забезпечення для моніторингу калорій та харчування. На основі раніше спроєктованих моделей визначено технологічну основу системи, структуру бази даних і алгоритми роботи основних програмних модулів.

Обґрунтовано вибір стеку технологій для розроблення системи. Для реалізації програмного продукту обрано веб-орієнтований підхід із використанням клієнт-серверної архітектури, що забезпечує зручну взаємодію користувача з інтерфейсом, централізоване оброблення даних і можливість подальшого розширення функціональності. Вибрані технології дають змогу реалізувати реєстрацію користувачів, ведення профілю, облік прийомів їжі, роботу з базою продуктів, розрахунок калорійності та формування аналітичних результатів.

Розроблено фізичну модель бази даних, яка включає таблиці користувачів, профілів харчування, добових цілей, продуктів, прийомів їжі, позицій прийому та рекомендацій. Така структура забезпечує збереження основних даних системи, підтримує зв’язки між сутностями та дозволяє коректно обробляти інформацію про раціон користувача. Передбачене використання первинних і зовнішніх ключів забезпечує цілісність даних, а окреме збереження продуктів, прийомів їжі та розрахованих показників зменшує дублювання інформації.

Також було виконано алгоритмізацію програмних модулів системи. Розроблені блок-схеми описують послідовність дій під час реєстрації та налаштування профілю, додавання прийому їжі, планування раціону й формування аналітичного звіту. Алгоритми враховують перевірку введених даних, пошук продуктів, додавання нових позицій до каталогу, оновлення добового підсумку, порівняння фактичних показників із цілями та формування рекомендацій.

Отже, у третьому розділі сформовано основу для програмної реалізації системи моніторингу калорій та харчування. Обраний стек технологій, фізична модель бази даних і розроблені алгоритми забезпечують логічну послідовність реалізації програмного продукту та створюють підґрунтя для подальшого тестування його функціональних можливостей.

4 ТЕСТУВАННЯ ТА ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ СИСТЕМИ

4.1 Тестування програмних модулів системи моніторингу калорій та харчування

Тестування програмних модулів системи моніторингу калорій та харчування спрямоване на перевірку коректності роботи основних функцій програмного забезпечення, правильності оброблення даних користувача, точності розрахунку калорійності та БЖВ, стабільності роботи бази даних і надійності механізмів авторизації. Методика тестування охоплює модульне, функціональне, інтеграційне та навантажувальне тестування, що дозволяє перевірити систему на різних рівнях її роботи.

Загальний план тестування програмних модулів наведено у таблиці 4.1.

Таблиця 4.1 - План тестування основних програмних модулів системи

№	Модуль / компонент	Тип тестування	Тестові дії	Очікуваний результат
1	Модуль автентифікації	Модульне	Реєстрація, вхід, перевірка ролей, оброблення помилкових даних	Коректний вхід до системи та розмежування доступу
2	Модуль профілю користувача	Функціональне	Введення віку, статі, зросту, ваги, рівня активності	Дані профілю збережено та використано в розрахунках
3	Модуль цілей харчування	Функціональне	Встановлення добової норми калорій, білків, жирів і вуглеводів	Добові цілі коректно збережено
4	Модуль бази продуктів	Модульне	Додавання, редагування, пошук і видалення продуктів	Каталог продуктів працює без помилок
5	Модуль додавання прийому їжі	Функціональне	Вибір продукту, введення порції, додавання до сніданку, обіду, вечері або перекусу	Прийом їжі створено та збережено в журналі
6	Модуль розрахунку калорій і БЖВ	Алгоритмічне	Перевірка розрахунків для різних мас порцій	Калорії, білки, жири та вуглеводи обчислено правильно

Продовження таблиці 4.1

7	Модуль добового підсумку	Інтеграційне	Додавання кількох прийомів їжі протягом дня	Добовий підсумок оновлюється автоматично
8	Модуль аналітики	Функціональне	Формування статистики за день, тиждень і місяць	Звіти містять повні та коректні показники
9	Модуль рекомендацій	Функціональне	Перевірка реакції на перевищення або недобір норми	Система формує відповідні рекомендації
10	Модуль сповіщень	Модульне	Перевірка нагадувань і повідомлень про відхилення	Користувач отримує коректні повідомлення
11	База даних	Інтеграційне	Збереження користувачів, продуктів, прийомів їжі та звітів	Дані зберігаються без втрати й дублювання
12	Продуктивність системи	Навантажувальне	Робота з великою кількістю продуктів і записів харчування	Система працює стабільно без помітних затримок

Методика оцінювання результатів тестування базується на перевірці правильності роботи бізнес-логіки системи, коректності взаємодії між модулями та стабільності збереження даних у базі SQLite. Для кожного функціонального модуля перевіряється відповідність фактичного результату очікуваному: створення користувача, збереження профілю, додавання продуктів, формування прийомів їжі, оновлення добового підсумку та побудова звітів.

Особлива увага приділяється модулю розрахунку калорій і БЖВ, оскільки він є основою роботи всієї системи. Під час тестування перевіряється, чи правильно система перераховує харчову цінність продукту відповідно до введеної маси порції, чи коректно додає ці значення до прийому їжі та чи правильно оновлює добовий підсумок користувача.

Для перевірки модуля профілю користувача аналізується коректність введення персональних параметрів, збереження цілей харчування та подальше використання цих даних у розрахунках. Система повинна не допускати

збереження порожніх, від’ємних або некоректних значень, а в разі помилки виводити зрозуміле повідомлення користувачу.

Інтеграційне тестування спрямоване на перевірку взаємодії між модулями профілю, продуктів, прийомів їжі, аналітики та бази даних. Під час такого тестування контролюється правильність виконання SQL-запитів, збереження даних у взаємопов’язаних таблицях, робота зовнішніх ключів і відсутність втрати інформації під час додавання або редагування записів.

Окремо перевіряється модуль аналітики. Для цього створюються тестові записи харчування за різні дні, після чого система формує підсумки за день, тиждень і місяць. Очікуваним результатом є правильне відображення кількості спожитих калорій, білків, жирів, вуглеводів, а також визначення перевищення або залишку добової норми.

Навантажувальне тестування дозволяє оцінити стабільність роботи системи при збільшенні кількості продуктів, користувачів і записів прийомів їжі. Під час тестування перевіряється швидкість пошуку продуктів, час додавання нового прийому їжі, швидкість формування звітів і стабільність роботи бази даних при послідовному виконанні операцій запису та читання.

У процесі тестування також перевіряється робота повідомлень і рекомендацій. Система повинна своєчасно інформувати користувача про перевищення добової норми, недостатнє споживання окремих нутрієнтів або необхідність оновлення харчового щоденника. Це забезпечує не лише технічну коректність роботи системи, а й практичну користь для користувача.

Отже, тестування програмних модулів дає змогу перевірити працездатність системи моніторингу калорій та харчування, виявити можливі помилки в обробленні даних, підтвердити правильність розрахунків і забезпечити стабільну роботу основних функцій програмного забезпечення.

4.2 Тестування інтерфейсних модулів інформаційної системи моніторингу калорій та харчування

Тестування інтерфейсних модулів інформаційної системи моніторингу калорій та харчування проводилося з метою перевірки коректності взаємодії користувача з основними екранами програмного продукту, правильності передавання введених даних до серверної частини та узгодженості результатів, що відображаються у вебінтерфейсі. Основну увагу приділено модулям реєстрації, авторизації, дашборду, журналу прийомів їжі, каталогу продуктів, планування раціону, аналітики та профілю користувача.

Головна сторінка системи виконує навігаційну та презентаційну функції. Під час її тестування перевірено відображення назви системи, доступність переходів до реєстрації та входу, коректне завантаження стилів, адаптивне розміщення інформаційних блоків і відсутність зайвих елементів, які могли б ускладнювати роботу користувача. Приклад головної сторінки системи наведено на рис. 4.1.

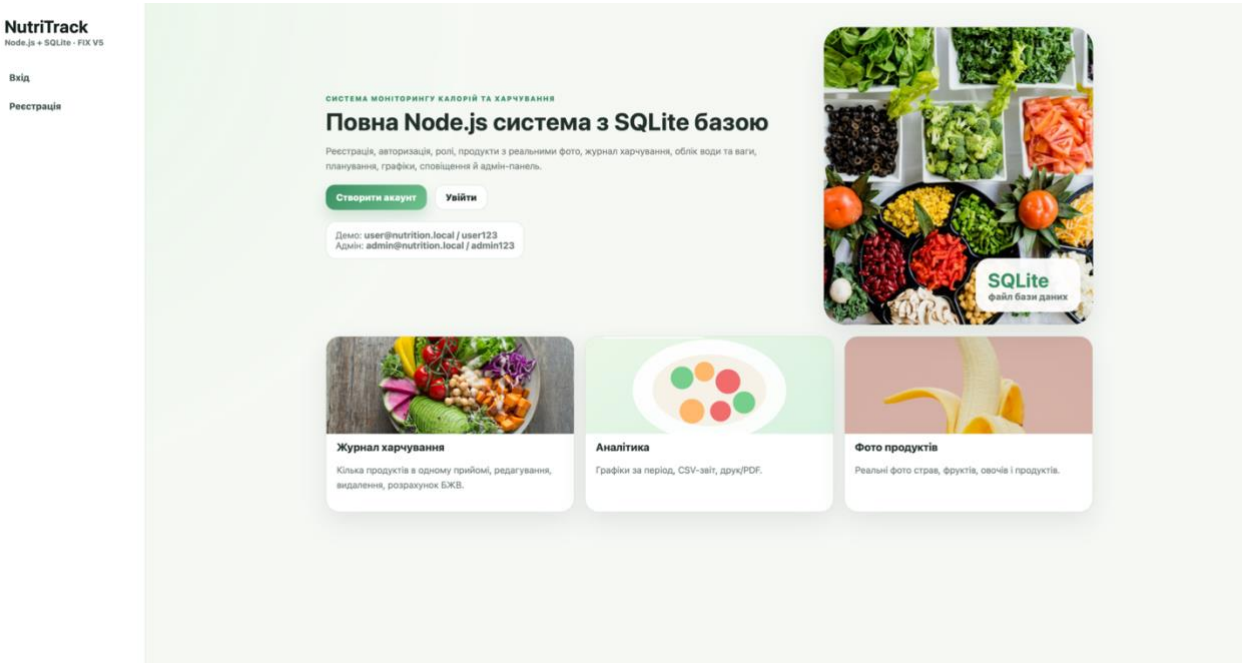

Рисунок 4.1 – Головна сторінка інформаційної системи NutriTrack

Окремо протестовано модуль реєстрації користувача. Перевірка включала введення ПІБ, електронної пошти та пароля, оброблення порожніх полів,

створення нового облікового запису і подальший перехід користувача до системи. Інтерфейс форми реєстрації показано на рис. 4.2. У результаті тестування встановлено, що форма має зрозумілу структуру, поля введення коректно відображаються, а після успішної реєстрації користувач може перейти до подальшого налаштування профілю.

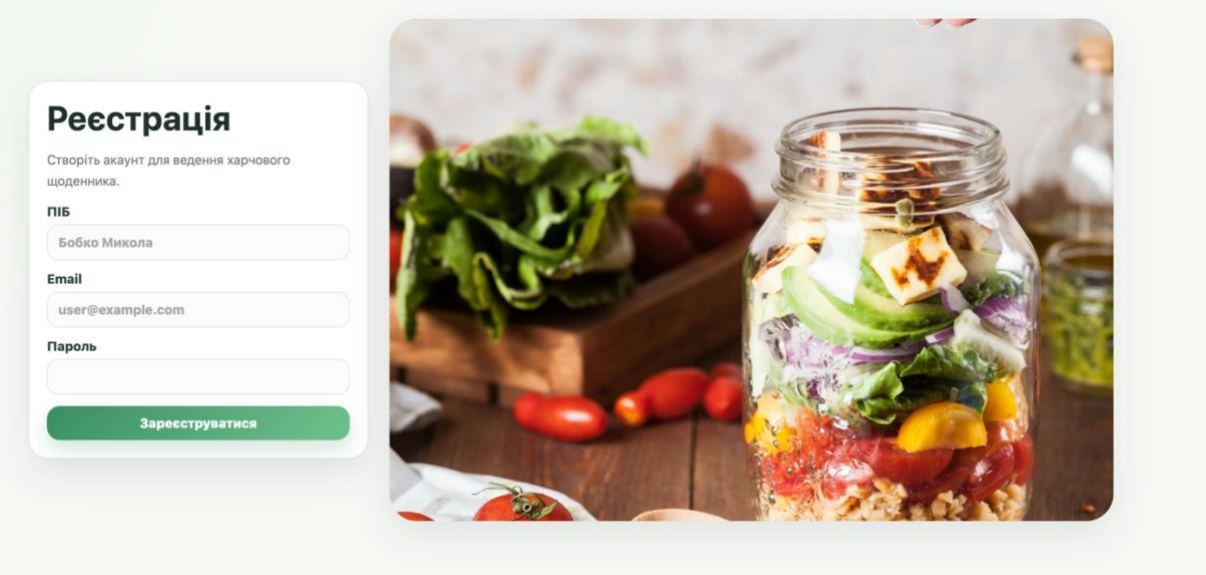

Рисунок 4.2 – Інтерфейс реєстрації користувача

Після входу користувача перевірявся дашборд системи, який є основним екраном контролю добового стану харчування. На цьому екрані відображаються добові значення калорій, білків, жирів, вуглеводів, облік води, поточна вага, короткий графік активності та перелік прийомів їжі за день. Тестування дашборду показало, що агреговані показники автоматично оновлюються після додавання прийомів їжі та відповідають даним, збереженим у базі SQLite. Приклад роботи дашборду наведено на рис. 4.3.

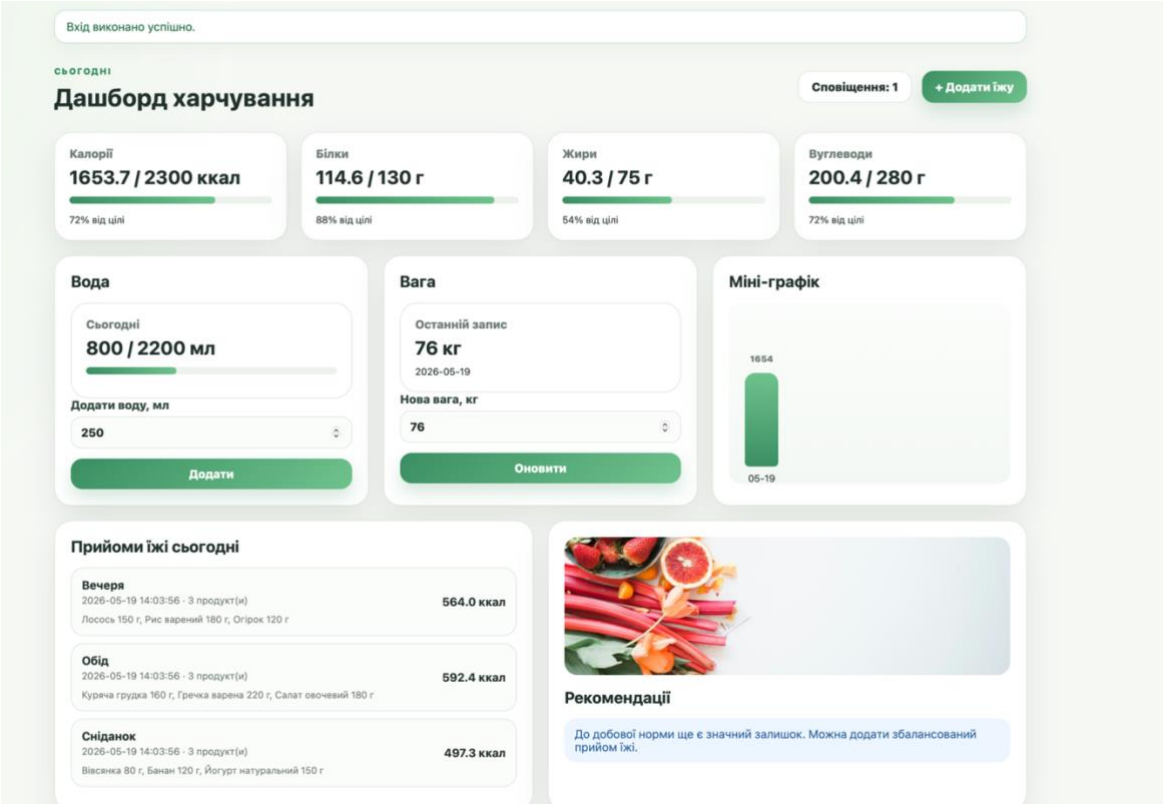

Рисунок 4.3 – Дашборд користувача з добовими показниками харчування

Особливу увагу приділено тестуванню журналу прийомів їжі, оскільки цей модуль безпосередньо впливає на правильність розрахунків. Перевірено додавання прийому їжі з одним і декількома продуктами, вибір типу прийому, введення маси порції, збереження запису у базі даних, редагування та видалення. Також перевірено, що система не створює порожній запис, якщо продукт або маса порції не передані до серверної частини. Інтерфейс журналу прийомів їжі наведено на рис. 4.4.

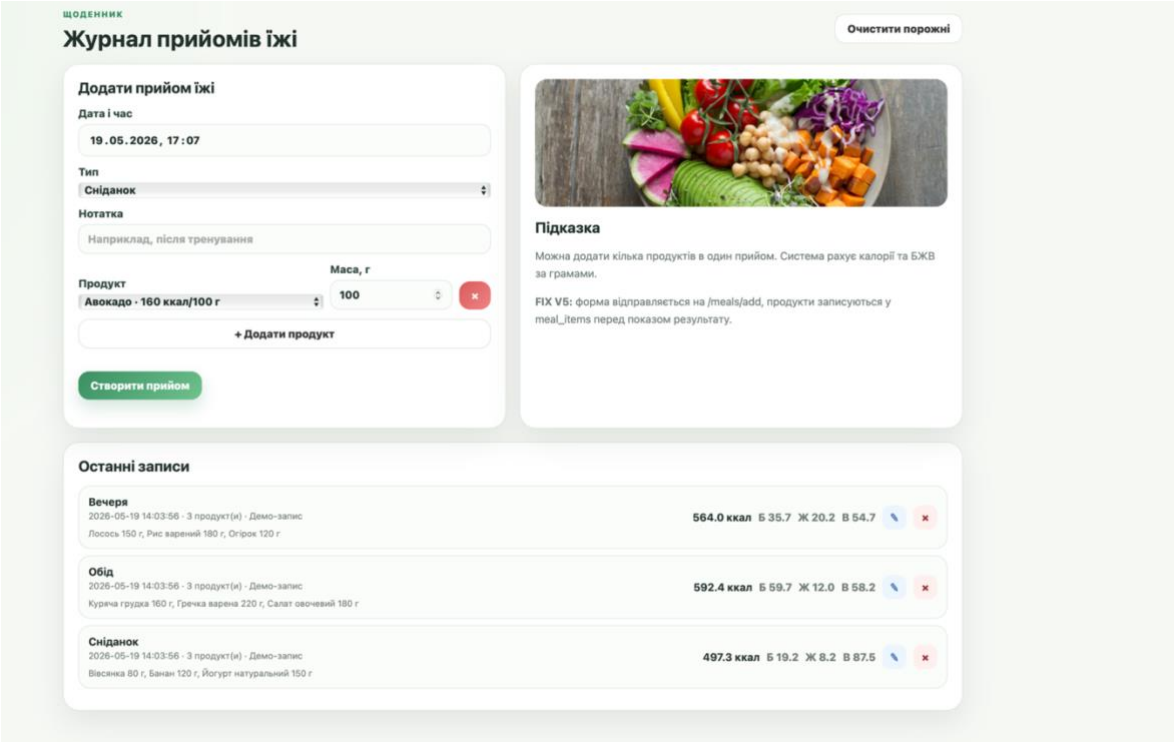

Рисунок 4.4 – Інтерфейс журналу прийомів їжі

Під час тестування підтверджено, що для кожного продукту система отримує харчову цінність на 100 г із таблиці продуктів, після чого виконує перерахунок за фактичною масою порції. Результат зберігається у таблиці позицій прийому їжі та надалі використовується для формування добового підсумку, аналітики й рекомендацій. Це дозволяє уникнути ручного дублювання розрахунків у клієнтській частині та забезпечує єдину логіку оброблення даних на сервері.

Каталог продуктів тестувався за сценаріями перегляду списку, пошуку продуктів, додавання нового продукту, редагування харчової цінності та видалення запису. У каталозі відображається назва продукту, категорія, калорійність, БЖВ та фото. Приклад інтерфейсу модуля керування продуктами наведено на рис. 4.5.

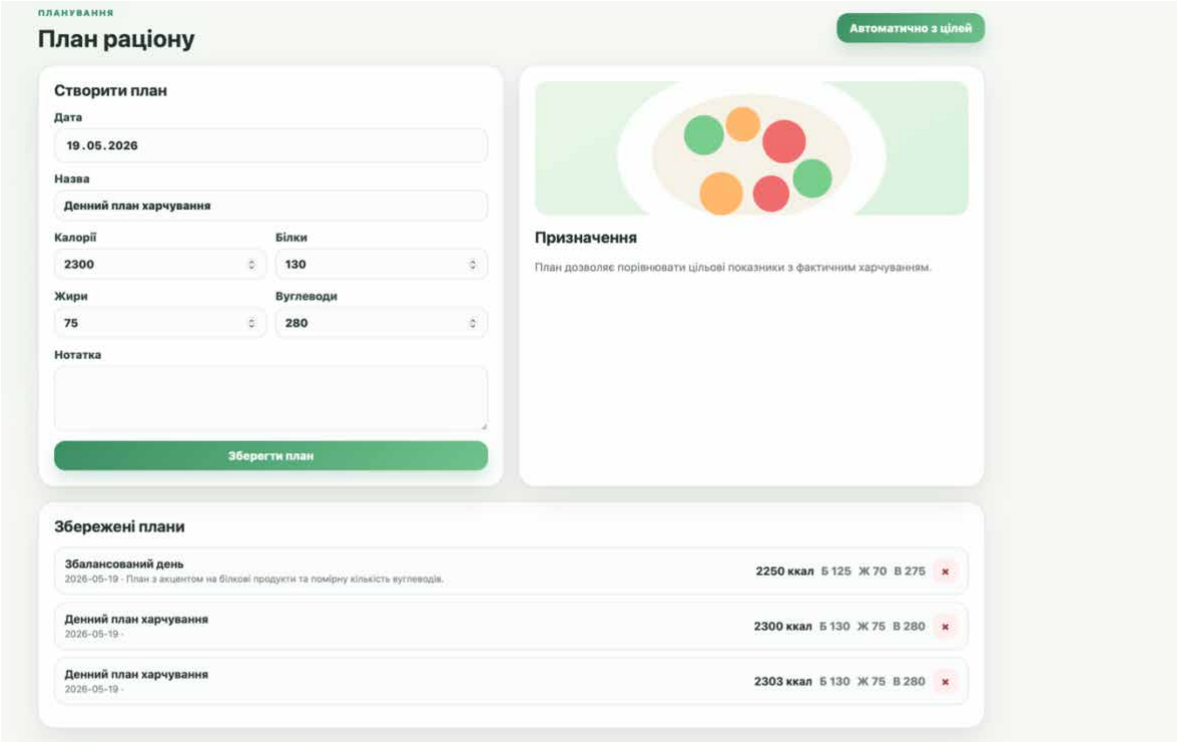

Рисунок 4.5 – Інтерфейс каталогу продуктів

Модуль планування раціону перевірявся шляхом створення денного плану харчування з цільовими значеннями калорій, білків, жирів і вуглеводів. Додатково протестовано автоматичне формування плану на основі добових цілей користувача. Результати тестування підтвердили, що створені плани зберігаються у базі даних і відображаються у відповідному списку. Інтерфейс планування раціону подано на рис. 4.6.

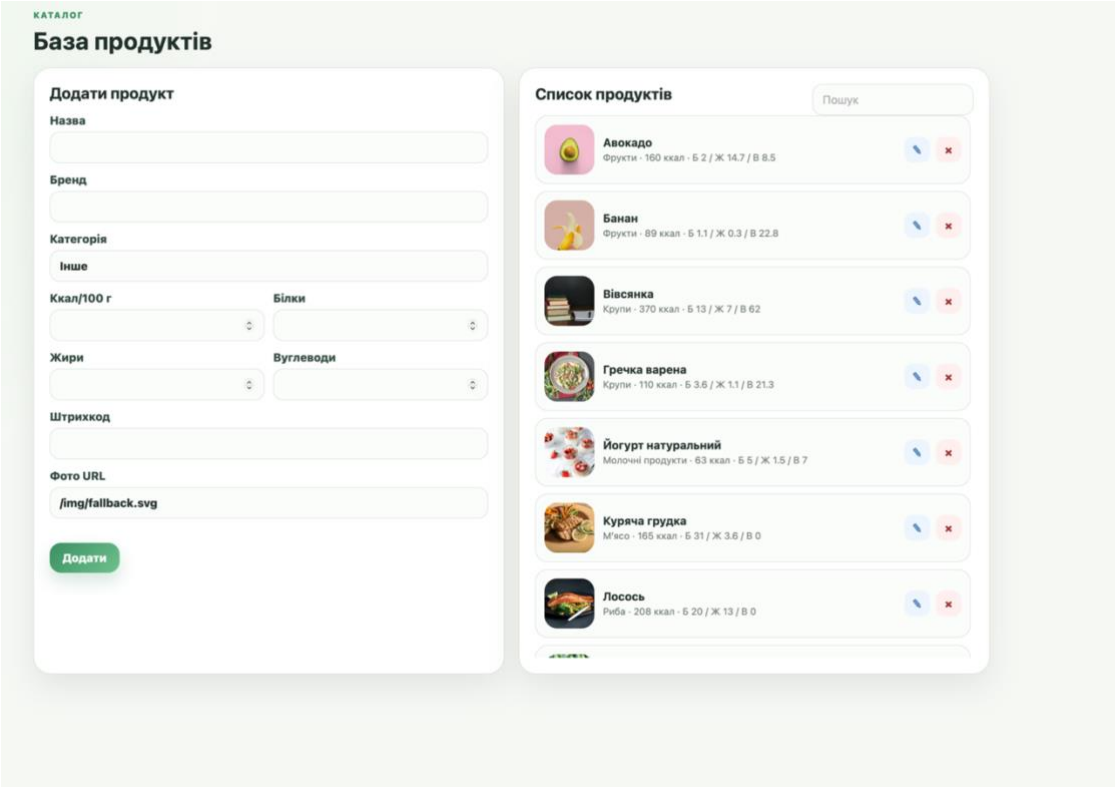

Рисунок 4.6 – Інтерфейс модуля планування раціону

Аналітичний модуль тестувався за періодом, заданим користувачем. Перевірено формування сумарних показників калорійності та БЖВ, побудову графіка динаміки калорійності, діаграми розподілу БЖВ, графіка споживання води та табличної деталізації за днями. Також протестовано експорт звіту у CSV та можливість друку або збереження звіту у PDF засобами браузера. Приклад аналітичної панелі наведено на рис. 4.7.

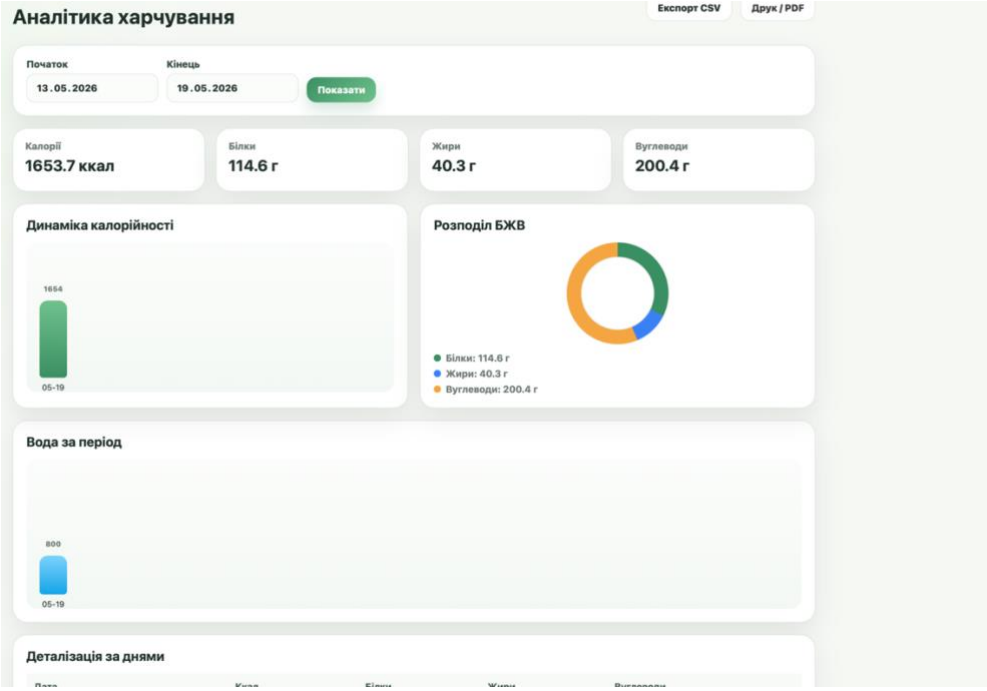

Рисунок 4.7 – Аналітична панель системи моніторингу харчування

Модуль профілю користувача перевірявся за сценарієм зміни персональних параметрів і добових цілей. До тестових дій входило оновлення статі, віку, зросту, ваги, рівня активності, цільової ваги, калорійності, білків, жирів, вуглеводів і норми води. Після збереження система повинна використовувати оновлені значення для дашборду, аналітики та рекомендацій. Інтерфейс профілю користувача показано на рис. 4.8.

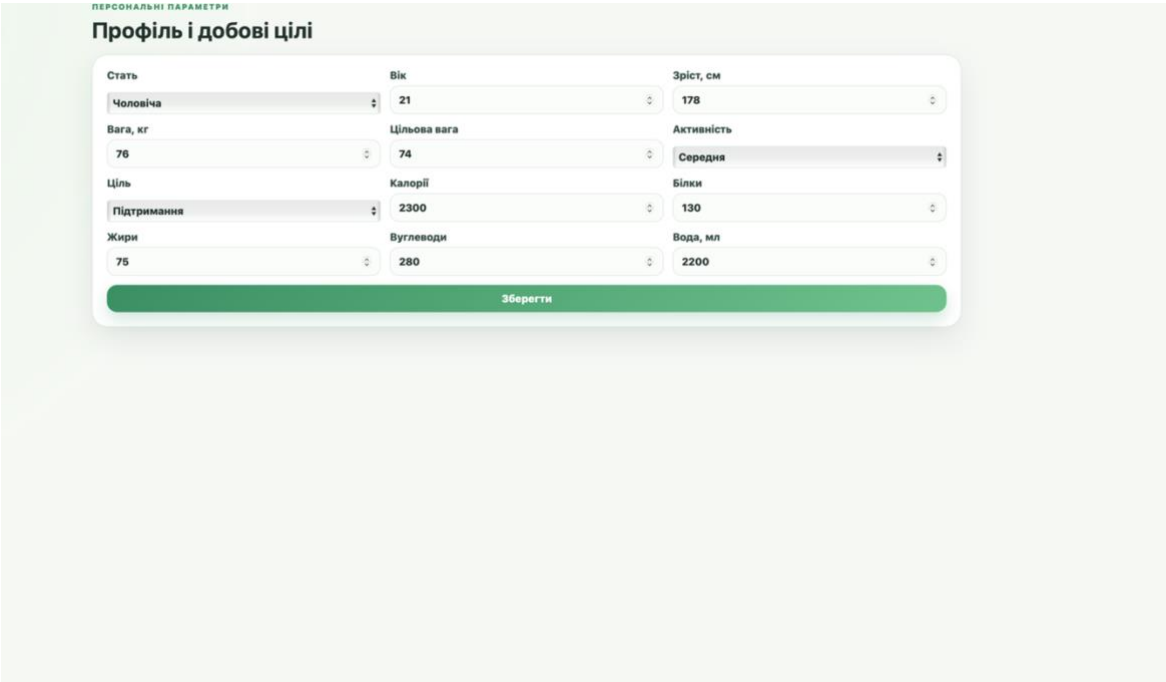

Рисунок 4.8 – Налаштування профілю та добових цілей користувача

Результати тестування інтерфейсних модулів узагальнено у табл. 4.2.

Таблиця 4.2 – Результати тестування інтерфейсних модулів системи

№	Інтерфейсний модуль	Тестові дії	Очікуваний результат	Результат
1	Головна сторінка	Перехід до входу та реєстрації	Кнопки навігації працюють, сторінка відображається коректно	Виконано
2	Реєстрація	Створення нового користувача	Дані зберігаються, користувач переходить до системи	Виконано
3	Авторизація	Вхід під тестовим акаунтом	Система відкриває дашборд користувача	Виконано
4	Дашборд	Перегляд добових показників	Калорії, БЖВ, вода та вага відображаються коректно	Виконано
5	Журнал їжі	Додавання прийому з кількома продуктами	Продукти записуються у базу, показники перераховуються	Виконано
6	Журнал їжі	Спроба створення запису без продукту	Порожній запис не створюється	Виконано
7	Каталог продуктів	Додавання, редагування, видалення продукту	Дані продукту змінюються у базі SQLite	Виконано
8	План раціону	Створення ручного та автоматичного плану	План зберігається і відображається у списку	Виконано
9	Аналітика	Вибір періоду, перегляд графіків	Дані агрегуються, графіки формуються коректно	Виконано
10	Профіль	Зміна добових цілей	Нові цілі застосовуються в дашборді та аналітиці	Виконано

Проведене тестування показало, що інтерфейсні модулі системи забезпечують повний цикл роботи користувача: від реєстрації та налаштування профілю до ведення журналу харчування, перегляду аналітики й формування плану раціону. Найважливішим результатом є підтвердження коректної взаємодії між формами інтерфейсу, серверною логікою Node.js та базою даних SQLite. Система правильно обробляє введені користувачем дані, не допускає створення некоректних порожніх записів і забезпечує автоматичний розрахунок калорійності та БЖВ на основі фактичної маси продуктів.

4.3 Результати тестування та аналіз ефективності системи, склад інсталяційного пакету

Результати проведеного тестування інформаційної системи моніторингу калорій та харчування підтвердили коректність реалізації основних функціональних модулів, стабільність роботи вебінтерфейсу, правильність збереження даних у базі SQLite та узгодженість розрахунків калорійності і БЖВ. У процесі перевірки оцінювалися сценарії реєстрації, авторизації, налаштування профілю, ведення журналу харчування, керування продуктами, формування плану раціону, побудови аналітики та роботи сповіщень.

На рисунку 4.9 подано діаграму розгортання інформаційної системи моніторингу калорій та харчування.

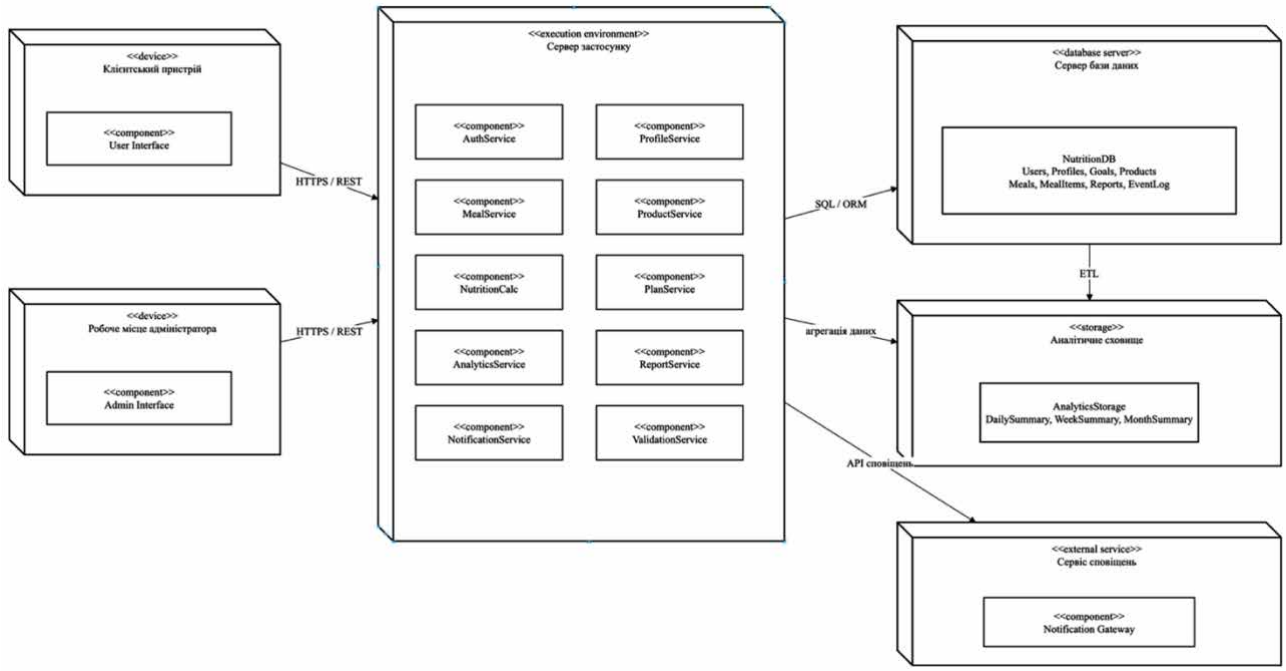

Рисунок 4.9 – Діаграма розгортання інформаційної системи моніторингу калорій та харчування

Подана діаграма відображає логічне розміщення основних програмних компонентів системи. Клієнтський пристрій користувача та робоче місце адміністратора взаємодіють із сервером застосунку через HTTPS/REST. На сервері застосунку розміщено прикладні сервіси авторизації, профілю, прийомів їжі, продуктів, розрахунку харчової цінності, планування, аналітики,

звітності, валідації та сповіщень. Зберігання основних даних виконується у базі NutritionDB, а агреговані показники можуть передаватися до аналітичного сховища для формування денних, тижневих і місячних підсумків.

У межах реалізованого програмного прототипу система розгортається як вебзастосунок на основі Node.js та SQLite. Клієнтська частина працює у браузері, а серверна частина забезпечує маршрутизацію запитів, оброблення форм, виконання розрахунків, роботу з базою даних і формування сторінок інтерфейсу. Такий підхід спрощує запуск системи, оскільки для роботи достатньо встановленого середовища Node.js і локального файлу бази даних SQLite.

Під час аналізу ефективності перевірено швидкість виконання типових операцій: вхід у систему, відкриття дашборду, додавання прийому їжі, перерахунок калорій і БЖВ, оновлення профілю, побудова аналітичних графіків та експорт звіту. Результати тестування показали, що система стабільно працює з локальною базою SQLite, коректно обробляє пов’язані записи таблиць meals, meal_items, foods, goals, water_logs і weight_logs, а також не допускає створення некоректних порожніх прийомів їжі.

Таблиця 4.3 – Результати оцінювання ефективності системи

№	Показник	Результат
1	Час авторизації користувача	до 1 с
2	Відкриття дашборду	до 1 с
3	Додавання прийому їжі з кількома продуктами	до 2 с
4	Перерахунок калорій та БЖВ	виконується автоматично
5	Оновлення добових цілей користувача	до 1 с
6	Формування аналітики за період	до 2 с
7	Експорт CSV-звіту	до 1 с
8	Коректність запису в SQLite	підтверджено
9	Створення порожніх прийомів їжі	не допускається
10	Стабільність роботи інтерфейсу	стабільна

Отримані результати підтверджують ефективність обраної архітектури та доцільність використання Node.js разом із SQLite для реалізації навчального програмного прототипу. Серверна частина централізує бізнес-логіку

розрахунків, тому клієнтський інтерфейс не дублює критичні обчислення, а лише передає введені користувачем дані до серверних маршрутів. Це підвищує узгодженість роботи системи та зменшує ризик помилок під час оброблення харчових записів.

Склад інсталяційного пакету системи охоплює програмні файли серверної частини, статичні ресурси інтерфейсу, локальну базу даних та службові файли запуску. До складу пакету входять:

– package.json - файл опису проєкту, залежностей і команд запуску;

– src/server.js - основний серверний файл Node.js;

– src/reset-db.js - службовий файл для скидання локальної бази даних;

– public/css/style.css - файл стилізації вебінтерфейсу;

– public/js/app.js - клієнтський JavaScript для роботи динамічних елементів форми;

– public/img/ - каталог локальних зображень і резервних ілюстрацій;

– data/nutrition.sqlite - файл бази даних SQLite;

– README.md - інструкція із запуску та опис тестових акаунтів;

– run_macos_linux.sh та run_windows.bat - допоміжні файли запуску для різних операційних систем.

Таблиця 4.4 – Склад інсталяційного пакету програмної системи

№	Елемент пакету	Призначення
1	package.json	Опис залежностей і команд запуску
2	src/server.js	Серверна логіка, маршрути, авторизація, розрахунки
3	src/reset-db.js	Очищення та повторне створення бази даних
4	public/css/style.css	Оформлення інтерфейсу користувача
5	public/js/app.js	Динамічне додавання продуктів у прийом їжі
6	data/nutrition.sqlite	Локальне зберігання користувачів, продуктів, прийомів і звітів
7	public/img/	Зображення продуктів і резервні графічні ресурси
8	README.md	Інструкція з розгортання і запуску
9	run_macos_linux.sh, run_windows.bat	Спрощений запуск застосунку

Для запуску системи користувач переходить у каталог проєкту, встановлює залежності командою npm install, за потреби очищує базу командою npm run reset-db та запускає сервер командою npm start. Після цього вебінтерфейс доступний у браузері за локальною адресою http://127.0.0.1:3000.

Проведене тестування та аналіз ефективності підтвердили, що розроблена інформаційна система забезпечує коректне ведення харчового щоденника, автоматичний розрахунок калорійності та БЖВ, підтримку персональних цілей, формування аналітики й роботу з локальною базою SQLite. Отже, програмний продукт є придатним для використання як навчальний прототип інформаційної системи моніторингу калорій та харчування.

4.4 Висновки до четвертого розділу

У четвертому розділі виконано тестування та оцінювання ефективності інформаційної системи моніторингу калорій та харчування. Проведено перевірку основних програмних модулів, інтерфейсних форм, механізмів авторизації, роботи з базою даних SQLite, розрахунку калорійності та БЖВ, формування аналітичних показників і роботи сповіщень.

У процесі тестування підтверджено коректність функціонування модулів реєстрації та авторизації користувачів, налаштування профілю, ведення журналу прийомів їжі, керування базою продуктів, створення плану раціону, перегляду аналітики та експорту звітів. Особливу увагу приділено перевірці журналу харчування, оскільки саме цей модуль забезпечує збереження даних про продукти, масу порцій та автоматичний розрахунок харчової цінності.

Тестування інтерфейсних модулів показало, що вебінтерфейс системи є зручним для користувача, має логічну структуру та забезпечує швидкий доступ до основних функцій. Сторінки дашборду, журналу їжі, каталогу продуктів, планування, аналітики та профілю коректно відображають дані, отримані із серверної частини та бази SQLite. Також підтверджено правильність роботи

форм введення, кнопок керування, навігації та механізмів оновлення інформації.

У результаті перевірки серверної логіки встановлено, що система правильно обробляє введені користувачем дані, не допускає створення некоректних порожніх записів прийомів їжі та забезпечує коректний перерахунок калорій, білків, жирів і вуглеводів відповідно до маси продукту. Дані зберігаються у взаємопов’язаних таблицях бази SQLite, що забезпечує цілісність інформації та можливість подальшого формування аналітичних звітів.

Проведений аналіз ефективності підтвердив достатню швидкодію системи при виконанні типових операцій: вході користувача, відкритті дашборду, додаванні прийому їжі, оновленні профілю, формуванні графіків та експорті звітів. Обрана архітектура на основі Node.js і SQLite є доцільною для реалізації навчального програмного прототипу, оскільки забезпечує простий запуск, мінімальні вимоги до середовища та стабільну роботу без складної серверної інфраструктури.

Також визначено склад інсталяційного пакету програмної системи, до якого входять серверні файли Node.js, статичні ресурси інтерфейсу, локальна база даних SQLite, службові файли запуску та інструкція користувача. Така структура спрощує розгортання системи на локальному комп’ютері та дає змогу швидко підготувати програмний продукт до демонстрації або подальшого доопрацювання.

Отже, результати четвертого розділу підтверджують працездатність, стабільність і практичну придатність розробленої інформаційної системи моніторингу калорій та харчування. Реалізоване програмне забезпечення забезпечує повний цикл роботи користувача з харчовими даними: від реєстрації та налаштування цілей до ведення журналу, розрахунку показників, перегляду аналітики та формування рекомендацій.

ВИСНОВКИ

У кваліфікаційній роботі було виконано проєктування та розроблення інформаційної системи моніторингу калорій та харчування, призначеної для автоматизованого ведення харчового щоденника, обліку прийомів їжі, розрахунку калорійності та БЖВ, контролю добових цілей і формування аналітичних показників. Розроблена система орієнтована на користувачів, які потребують зручного інструменту для контролю раціону, а також на адміністратора, який забезпечує керування базою продуктів і користувачами.

У першому розділі було досліджено предметну область систем моніторингу харчування, визначено основні процеси обліку продуктів, прийомів їжі, добових норм, аналітики та рекомендацій. Проведено аналіз існуючих програмних рішень, що використовуються для контролю калорійності та харчової цінності раціону. Встановлено, що більшість аналогів забезпечує базове ведення харчового щоденника, однак не завжди поєднує простоту використання, локальне зберігання даних, прозорі розрахунки та можливість адміністрування продуктів у межах одного навчального програмного прототипу. На основі цього було сформовано функціональні, нефункціональні та безпекові вимоги до системи.

У процесі моделювання предметної області побудовано DFD-діаграму, UML-діаграму прецедентів, діаграму послідовності та діаграму діяльності. Ці моделі дали змогу формалізувати основні сценарії роботи системи: реєстрацію й авторизацію користувача, налаштування профілю, додавання прийомів їжі, пошук продуктів, розрахунок калорій і БЖВ, перегляд щоденного підсумку, формування аналітики та адміністрування каталогу продуктів. У результаті було уточнено структуру майбутньої системи та взаємодію між її основними учасниками.

У другому розділі виконано проєктування інформаційного та програмного забезпечення системи. Розроблено логічну модель даних у вигляді

ER-діаграми, у якій визначено ключові сутності: користувач, профіль харчування, добова ціль, продукт, прийом їжі, позиція прийому їжі та рекомендація. Структуру даних нормалізовано, що дозволило уникнути дублювання інформації про продукти та забезпечити коректне зберігання пов’язаних записів. Також розроблено діаграму класів, кооперації, діаграму компонентів і діаграму пакетів, які відображають програмну структуру системи та розподіл функцій між модулями інтерфейсу, прикладної логіки, аналітики, планування, сповіщень і збереження даних.

У третьому розділі обґрунтовано вибір технологічного стеку та реалізовано програмний прототип системи. Для серверної частини використано Node.js, для зберігання даних — SQLite, для побудови вебінтерфейсу — HTML, CSS і JavaScript. Такий стек було обрано через простоту запуску, мінімальні вимоги до середовища, достатню швидкодію та зручність використання у навчальному проєкті. Реалізовано базу даних із таблицями користувачів, профілів, цілей, продуктів, прийомів їжі, позицій прийому їжі, планів, води, ваги, рекомендацій та журналу подій.

У програмній реалізації створено модулі реєстрації, авторизації, керування профілем, ведення журналу харчування, обліку продуктів, планування раціону, аналітики, рекомендацій, сповіщень та адміністрування. Особливу увагу приділено коректності розрахунків: система бере харчову цінність продукту на 100 г, перераховує її відповідно до фактичної маси порції та зберігає результат у базі даних. Це забезпечує правильне формування добових підсумків, графіків і рекомендацій.

У четвертому розділі проведено тестування програмних та інтерфейсних модулів системи. Перевірено роботу форм реєстрації, авторизації, профілю, дашборду, журналу прийомів їжі, каталогу продуктів, планування раціону, аналітики та сповіщень. Тестування підтвердило, що система коректно обробляє введені користувачем дані, не допускає створення порожніх прийомів їжі, правильно зберігає пов’язані записи в SQLite та автоматично оновлює добові показники після додавання або редагування прийомів їжі.

Результати тестування показали, що розроблена система стабільно виконує основні функції, має зрозумілий інтерфейс, підтримує роботу з реальними фото продуктів, забезпечує графічне відображення аналітики та дозволяє експортувати звітні дані. Визначено склад інсталяційного пакету, до якого входять серверні файли Node.js, статичні ресурси інтерфейсу, файл бази даних SQLite, службові скрипти запуску та інструкція користувача.

Отже, поставлену мету кваліфікаційної роботи досягнуто. У результаті виконання роботи розроблено працездатну інформаційну систему моніторингу калорій та харчування, яка забезпечує реєстрацію користувачів, ведення харчового щоденника, керування продуктами, розрахунок калорійності та БЖВ, облік води й ваги, планування раціону, формування аналітики та рекомендацій. Розроблений програмний продукт може бути використаний як навчальний прототип системи персонального контролю харчування та надалі розширений шляхом додавання мобільної версії, інтеграції зі сканером штрихкодів, зовнішніми базами продуктів і персоналізованими алгоритмами рекомендацій.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. World Health Organization. Healthy diet [Електронний ресурс]. URL: https://www.who.int/news-room/fact-sheets/detail/healthy-diet (дата звернення: 19.05.2026).

2. World Health Organization. Obesity and overweight [Електронний ресурс]. URL: https://www.who.int/news-room/fact-sheets/detail/obesity-and-overweight (дата звернення: 19.05.2026).

3. World Health Organization. Physical activity [Електронний ресурс]. URL: https://www.who.int/news-room/fact-sheets/detail/physical-activity (дата звернення: 19.05.2026).

4. U.S. Department of Agriculture, U.S. Department of Health and Human Services. Dietary Guidelines for Americans, 2020–2025 [Електронний ресурс]. URL: https://www.dietaryguidelines.gov/resources/2020-2025-dietary-guidelines-online-materials (дата звернення: 19.05.2026).

5. USDA. FoodData Central [Електронний ресурс]. URL: https://fdc.nal.usda.gov/ (дата звернення: 19.05.2026).

6. USDA. FoodData Central API Guide [Електронний ресурс]. URL: https://fdc.nal.usda.gov/api-guide.html (дата звернення: 19.05.2026).

7. Open Food Facts. Open Food Facts API Documentation [Електронний ресурс]. URL: https://openfoodfacts.github.io/openfoodfacts-server/api/ (дата звернення: 19.05.2026).

8. Edamam. Food Database API Documentation [Електронний ресурс]. URL: https://developer.edamam.com/food-database-api-docs (дата звернення: 19.05.2026).

9. FatSecret Platform API [Електронний ресурс]. URL: https://platform.fatsecret.com/api/ (дата звернення: 19.05.2026).

10. MyFitnessPal. Calorie Counter App [Електронний ресурс]. URL: https://www.myfitnesspal.com/ (дата звернення: 19.05.2026).

11. Cronometer. Nutrition Tracking App [Електронний ресурс]. URL: https://cronometer.com/ (дата звернення: 19.05.2026).

12. MyNetDiary. Calorie Counter and Diet Assistant [Електронний ресурс]. URL: https://www.mynetdiary.com/ (дата звернення: 19.05.2026).

13. Lose It! Calorie Counter [Електронний ресурс]. URL: https://www.loseit.com/ (дата звернення: 19.05.2026).

14. Node.js. Documentation [Електронний ресурс]. URL: https://nodejs.org/docs/latest/api/ (дата звернення: 19.05.2026).

15. Node.js. About Node.js [Електронний ресурс]. URL: https://nodejs.org/en/about (дата звернення: 19.05.2026).

16. npm Docs. About npm [Електронний ресурс]. URL: https://docs.npmjs.com/about-npm (дата звернення: 19.05.2026).

17. Express.js. Node.js web application framework [Електронний ресурс]. URL: https://expressjs.com/ (дата звернення: 19.05.2026).

18. Express.js. Routing [Електронний ресурс]. URL: https://expressjs.com/en/guide/routing.html (дата звернення: 19.05.2026).

19. express-session. Simple session middleware for Express [Електронний ресурс]. URL: https://www.npmjs.com/package/express-session (дата звернення: 19.05.2026).

20. bcrypt.js. Optimized bcrypt in JavaScript [Електронний ресурс]. URL: https://www.npmjs.com/package/bcryptjs(дата звернення: 19.05.2026).

21. sql.js. SQLite compiled to JavaScript [Електронний ресурс]. URL: https://github.com/sql-js/sql.js (дата звернення: 19.05.2026).

22. Multer. Node.js middleware for handling multipart/form-data [Електронний ресурс]. URL: https://github.com/expressjs/multer (дата звернення: 19.05.2026).

23. SQLite. SQLite Documentation [Електронний ресурс]. URL: https://sqlite.org/docs.html (дата звернення: 19.05.2026).

24. SQLite. SQL Language: CREATE TABLE [Електронний ресурс]. URL: https://sqlite.org/lang_createtable.html(дата звернення: 19.05.2026).

25. SQLite. Foreign Key Support [Електронний ресурс]. URL: https://sqlite.org/foreignkeys.html (дата звернення: 19.05.2026).

26. MDN Web Docs. HTML: HyperText Markup Language [Електронний ресурс]. URL: https://developer.mozilla.org/en-US/docs/Web/HTML (дата звернення: 19.05.2026).

27. MDN Web Docs. CSS: Cascading Style Sheets [Електронний ресурс]. URL: https://developer.mozilla.org/en-US/docs/Web/CSS (дата звернення: 19.05.2026).

28. MDN Web Docs. JavaScript [Електронний ресурс]. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript (дата звернення: 19.05.2026).

29. MDN Web Docs. Working with JSON [Електронний ресурс]. URL: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON (дата звернення: 19.05.2026).

30. OWASP Foundation. Authentication Cheat Sheet [Електронний ресурс]. URL: https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html (дата звернення: 19.05.2026).

31. OWASP Foundation. Password Storage Cheat Sheet [Електронний ресурс]. URL: https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html (дата звернення: 19.05.2026).

32. OWASP Foundation. Session Management Cheat Sheet [Електронний ресурс]. URL: https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html (дата звернення: 19.05.2026).

33. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures [Електронний ресурс]: doctoral dissertation. Irvine: University of California, 2000.

URL: https://ics.uci.edu/~fielding/pubs/dissertation/top.htm (дата звернення: 19.05.2026).

34. Fielding R., Nottingham M., Reschke J. RFC 9110: HTTP Semantics [Електронний ресурс]. RFC Editor, 2022. URL: https://www.rfc-editor.org/rfc/rfc9110.html (дата звернення: 19.05.2026).

35. Object Management Group. OMG Unified Modeling Language Version 2.5.1 [Електронний ресурс]. URL: https://www.omg.org/spec/UML/2.5.1/PDF (дата звернення: 19.05.2026).

36. ISO/IEC 25010:2011. Systems and software engineering. Systems and software Quality Requirements and Evaluation (SQuaRE). System and software quality models [Електронний ресурс]. URL: https://www.iso.org/standard/35733.html (дата звернення: 19.05.2026).

ДОДАТОК А

Фрагмент програмного коду серверної частини інформаційної системи NutriTrack

```
// app.js
const express = require("express");
const session = require("express-session");
const bcrypt = require("bcrypt");
const path = require("path");
const { initDatabase, db } = require("./database");

const app = express();
const PORT = 3000;

app.use(express.urlencoded({ extended: true }));
app.use(express.json());
app.use(express.static(path.join(__dirname, "public")));

app.use(session({
    secret: "nutritrack-session-secret",
    resave: false,
    saveUninitialized: false,
    cookie: {
        maxAge: 1000 * 60 * 60 * 24
    }
}));

initDatabase();

function requireAuth(req, res, next) {
    if (!req.session.user) {
        return res.status(401).json({ error: "Користувач не авторизований" });
    }
    next();
}

function requireAdmin(req, res, next) {
    if (!req.session.user || req.session.user.role !== "admin") {
        return res.status(403).json({ error: "Недостатньо прав доступу" });
    }
    next();
}

app.post("/api/register", async (req, res) => {
    try {
        const { fullName, email, password } = req.body;

        if (!fullName || !email || !password) {
            return res.status(400).json({ error: "Заповніть усі обов’язкові
поля" });
        }

        const existingUser = db.prepare(`
            SELECT id FROM users WHERE email = ?
        `).get(email);

        if (existingUser) {
            return res.status(409).json({ error: "Користувач із таким email уже
існує" });
        }
```

```
        const passwordHash = await bcrypt.hash(password, 10);

        const result = db.prepare(`
            INSERT INTO users (full_name, email, password_hash, role,
created_at)
            VALUES (?, ?, ?, 'user', datetime('now'))
        `).run(fullName, email, passwordHash);

        db.prepare(`
            INSERT INTO user_profiles
            (user_id, age, gender, height_cm, weight_kg, activity_level)
            VALUES (?, 18, 'не вказано', 170, 70, 'середній')
        `).run(result.lastInsertRowid);

        db.prepare(`
            INSERT INTO nutrition_goals
            (user_id, goal_type, calories_goal, proteins_goal, fats_goal,
carbs_goal)
            VALUES (?, 'підтримання ваги', 2200, 120, 70, 260)
        `).run(result.lastInsertRowid);

        res.status(201).json({ message: "Користувача зареєстровано" });
    } catch (error) {
        res.status(500).json({ error: "Помилка реєстрації користувача" });
    }
});

app.post("/api/login", async (req, res) => {
    try {
        const { email, password } = req.body;

        const user = db.prepare(`
            SELECT id, full_name, email, password_hash, role
            FROM users
            WHERE email = ?
        `).get(email);

        if (!user) {
            return res.status(401).json({ error: "Невірний email або пароль" });
        }

        const isValidPassword = await bcrypt.compare(password,
user.password_hash);

        if (!isValidPassword) {
            return res.status(401).json({ error: "Невірний email або пароль" });
        }

        req.session.user = {
            id: user.id,
            fullName: user.full_name,
            email: user.email,
            role: user.role
        };

        res.json({
            message: "Авторизація успішна",
            user: req.session.user
        });
    } catch (error) {
        res.status(500).json({ error: "Помилка авторизації" });
    }
```

```
});

app.post("/api/logout", requireAuth, (req, res) => {
    req.session.destroy(() => {
        res.json({ message: "Вихід із системи виконано" });
    });
});

app.get("/api/profile", requireAuth, (req, res) => {
    const profile = db.prepare(`
        SELECT u.full_name, u.email, p.age, p.gender, p.height_cm,
               p.weight_kg, p.activity_level, g.goal_type,
               g.calories_goal, g.proteins_goal, g.fats_goal, g.carbs_goal
        FROM users u
        JOIN user_profiles p ON p.user_id = u.id
        JOIN nutrition_goals g ON g.user_id = u.id
        WHERE u.id = ?
    `).get(req.session.user.id);

    res.json(profile);
});

app.put("/api/profile", requireAuth, (req, res) => {
    const {
        age,
        gender,
        heightCm,
        weightKg,
        activityLevel,
        goalType,
        caloriesGoal,
        proteinsGoal,
        fatsGoal,
        carbsGoal
    } = req.body;

    if (age <= 0 || heightCm <= 0 || weightKg <= 0) {
        return res.status(400).json({ error: "Параметри профілю мають бути
коректними" });
    }

    db.prepare(`
        UPDATE user_profiles
        SET age = ?, gender = ?, height_cm = ?, weight_kg = ?, activity_level =
?
        WHERE user_id = ?
    `).run(age, gender, heightCm, weightKg, activityLevel, req.session.user.id);

    db.prepare(`
        UPDATE nutrition_goals
        SET goal_type = ?, calories_goal = ?, proteins_goal = ?,
            fats_goal = ?, carbs_goal = ?
        WHERE user_id = ?
    `).run(
        goalType,
        caloriesGoal,
        proteinsGoal,
        fatsGoal,
        carbsGoal,
        req.session.user.id
    );

    res.json({ message: "Профіль і цілі харчування оновлено" });
```

```
});

app.get("/api/products", requireAuth, (req, res) => {
    const search = req.query.search || "";

    const products = db.prepare(`
        SELECT id, name, calories_per_100g, proteins_per_100g,
               fats_per_100g, carbs_per_100g
        FROM products
        WHERE name LIKE ?
        ORDER BY name
        LIMIT 50
    `).all(`%${search}%`);

    res.json(products);
});

app.post("/api/products", requireAdmin, (req, res) => {
    const { name, calories, proteins, fats, carbs } = req.body;

    if (!name || calories < 0 || proteins < 0 || fats < 0 || carbs < 0) {
        return res.status(400).json({ error: "Некоректні дані продукту" });
    }

    db.prepare(`
        INSERT INTO products
        (name, calories_per_100g, proteins_per_100g, fats_per_100g,
carbs_per_100g)
        VALUES (?, ?, ?, ?, ?)
    `).run(name, calories, proteins, fats, carbs);

    res.status(201).json({ message: "Продукт додано до бази" });
});

app.listen(PORT, () => {
    console.log(`NutriTrack запущено: http://localhost:${PORT}`);
});
```

ДОДАТОК Б

Фрагмент програмного коду створення бази даних NutriTrack

```
// database.js
const Database = require("better-sqlite3");
const path = require("path");
const fs = require("fs");

const dataDir = path.join(__dirname, "data");

if (!fs.existsSync(dataDir)) {
    fs.mkdirSync(dataDir);
}

const db = new Database(path.join(dataDir, "nutritrack.sqlite"));

function initDatabase() {
    db.pragma("foreign_keys = ON");

    db.exec(`
        CREATE TABLE IF NOT EXISTS users (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            full_name TEXT NOT NULL,
            email TEXT NOT NULL UNIQUE,
            password_hash TEXT NOT NULL,
            role TEXT NOT NULL DEFAULT 'user',
            created_at TEXT NOT NULL
        );

        CREATE TABLE IF NOT EXISTS user_profiles (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            user_id INTEGER NOT NULL UNIQUE,
            age INTEGER NOT NULL,
            gender TEXT NOT NULL,
            height_cm REAL NOT NULL,
            weight_kg REAL NOT NULL,
            activity_level TEXT NOT NULL,
            FOREIGN KEY (user_id) REFERENCES users(id)
                ON UPDATE CASCADE
                ON DELETE CASCADE
        );

        CREATE TABLE IF NOT EXISTS nutrition_goals (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            user_id INTEGER NOT NULL UNIQUE,
            goal_type TEXT NOT NULL,
            calories_goal REAL NOT NULL,
            proteins_goal REAL NOT NULL,
            fats_goal REAL NOT NULL,
            carbs_goal REAL NOT NULL,
            FOREIGN KEY (user_id) REFERENCES users(id)
                ON UPDATE CASCADE
                ON DELETE CASCADE
        );

        CREATE TABLE IF NOT EXISTS products (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            name TEXT NOT NULL UNIQUE,
            calories_per_100g REAL NOT NULL,
```

```
    proteins_per_100g REAL NOT NULL,
    fats_per_100g REAL NOT NULL,
    carbs_per_100g REAL NOT NULL,
    created_at TEXT NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE IF NOT EXISTS meal_entries (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    user_id INTEGER NOT NULL,
    meal_type TEXT NOT NULL,
    meal_date TEXT NOT NULL,
    meal_time TEXT NOT NULL,
    note TEXT,
    created_at TEXT NOT NULL DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(id)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);

CREATE TABLE IF NOT EXISTS meal_products (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    meal_id INTEGER NOT NULL,
    product_id INTEGER NOT NULL,
    portion_grams REAL NOT NULL,
    calculated_calories REAL NOT NULL,
    calculated_proteins REAL NOT NULL,
    calculated_fats REAL NOT NULL,
    calculated_carbs REAL NOT NULL,
    FOREIGN KEY (meal_id) REFERENCES meal_entries(id)
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    FOREIGN KEY (product_id) REFERENCES products(id)
        ON UPDATE CASCADE
        ON DELETE RESTRICT
);

CREATE TABLE IF NOT EXISTS daily_summaries (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    user_id INTEGER NOT NULL,
    summary_date TEXT NOT NULL,
    total_calories REAL NOT NULL DEFAULT 0,
    total_proteins REAL NOT NULL DEFAULT 0,
    total_fats REAL NOT NULL DEFAULT 0,
    total_carbs REAL NOT NULL DEFAULT 0,
    calories_difference REAL NOT NULL DEFAULT 0,
    updated_at TEXT NOT NULL DEFAULT CURRENT_TIMESTAMP,
    UNIQUE(user_id, summary_date),
    FOREIGN KEY (user_id) REFERENCES users(id)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);

CREATE TABLE IF NOT EXISTS notifications (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    user_id INTEGER NOT NULL,
    title TEXT NOT NULL,
    message TEXT NOT NULL,
    status TEXT NOT NULL DEFAULT 'new',
    created_at TEXT NOT NULL DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(id)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);
```

```
        CREATE TABLE IF NOT EXISTS audit_log (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            user_id INTEGER,
            action TEXT NOT NULL,
            entity_name TEXT NOT NULL,
            entity_id INTEGER,
            created_at TEXT NOT NULL DEFAULT CURRENT_TIMESTAMP,
            FOREIGN KEY (user_id) REFERENCES users(id)
                ON UPDATE CASCADE
                ON DELETE SET NULL
        );

        CREATE INDEX IF NOT EXISTS idx_products_name
            ON products(name);

        CREATE INDEX IF NOT EXISTS idx_meal_entries_user_date
            ON meal_entries(user_id, meal_date);

        CREATE INDEX IF NOT EXISTS idx_daily_summaries_user_date
            ON daily_summaries(user_id, summary_date);
    `);

    seedProducts();
}

function seedProducts() {
    const count = db.prepare(`
        SELECT COUNT(*) AS total FROM products
    `).get().total;

    if (count > 0) {
        return;
    }

    const insertProduct = db.prepare(`
        INSERT INTO products
        (name, calories_per_100g, proteins_per_100g, fats_per_100g,
carbs_per_100g)
        VALUES (?, ?, ?, ?, ?)
    `);

    const products = [
        ["Куряче філе", 165, 31, 3.6, 0],
        ["Яйце куряче", 157, 12.7, 10.9, 0.7],
        ["Рис відварений", 130, 2.7, 0.3, 28.2],
        ["Гречка відварена", 110, 3.6, 1.1, 21.3],
        ["Вівсянка", 389, 16.9, 6.9, 66.3],
        ["Сир кисломолочний 5%", 121, 17, 5, 1.8],
        ["Яблуко", 52, 0.3, 0.2, 13.8],
        ["Банан", 89, 1.1, 0.3, 22.8],
        ["Огірок", 15, 0.7, 0.1, 3.6],
        ["Помідор", 18, 0.9, 0.2, 3.9],
        ["Оливкова олія", 884, 0, 100, 0],
        ["Хліб цільнозерновий", 247, 13, 4.2, 41]
    ];

    const transaction = db.transaction(() => {
        for (const product of products) {
            insertProduct.run(product);
        }
    });
```

```
    transaction();
}

function recalculateDailySummary(userId, date) {
    const totals = db.prepare(`
        SELECT
            COALESCE(SUM(mp.calculated_calories), 0) AS calories,
            COALESCE(SUM(mp.calculated_proteins), 0) AS proteins,
            COALESCE(SUM(mp.calculated_fats), 0) AS fats,
            COALESCE(SUM(mp.calculated_carbs), 0) AS carbs
        FROM meal_entries me
        JOIN meal_products mp ON mp.meal_id = me.id
        WHERE me.user_id = ? AND me.meal_date = ?
    `).get(userId, date);

    const goal = db.prepare(`
        SELECT calories_goal
        FROM nutrition_goals
        WHERE user_id = ?
    `).get(userId);

    const caloriesDifference = goal
        ? goal.calories_goal - totals.calories
        : 0 - totals.calories;

    db.prepare(`
        INSERT INTO daily_summaries
        (user_id, summary_date, total_calories, total_proteins,
         total_fats, total_carbs, calories_difference, updated_at)
        VALUES (?, ?, ?, ?, ?, ?, ?, datetime('now'))
        ON CONFLICT(user_id, summary_date)
        DO UPDATE SET
            total_calories = excluded.total_calories,
            total_proteins = excluded.total_proteins,
            total_fats = excluded.total_fats,
            total_carbs = excluded.total_carbs,
            calories_difference = excluded.calories_difference,
            updated_at = datetime('now')
    `).run(
        userId,
        date,
        totals.calories,
        totals.proteins,
        totals.fats,
        totals.carbs,
        caloriesDifference
    );
}

module.exports = {
    db,
    initDatabase,
    recalculateDailySummary
};
```

ДОДАТОК В

Фрагмент програмного коду модуля додавання прийому їжі та формування аналітики

```
// meal-routes.js
const express = require("express");
const router = express.Router();
const { db, recalculateDailySummary } = require("./database");

function requireAuth(req, res, next) {
    if (!req.session.user) {
        return res.status(401).json({ error: "Користувач не авторизований" });
    }
    next();
}

function calculateNutrition(product, portionGrams) {
    const coefficient = portionGrams / 100;

    return {
        calories: Number((product.calories_per_100g * coefficient).toFixed(2)),
        proteins: Number((product.proteins_per_100g * coefficient).toFixed(2)),
        fats: Number((product.fats_per_100g * coefficient).toFixed(2)),
        carbs: Number((product.carbs_per_100g * coefficient).toFixed(2))
    };
}

router.post("/api/meals", requireAuth, (req, res) => {
    try {
        const {
            mealType,
            mealDate,
            mealTime,
            products,
            note
        } = req.body;

        if (!mealType || !mealDate || !mealTime) {
            return res.status(400).json({ error: "Не заповнено дані прийому їжі"
});
        }

        if (!Array.isArray(products) || products.length === 0) {
            return res.status(400).json({ error: "До прийому їжі потрібно додати
хоча б один продукт" });
        }

        const createMealTransaction = db.transaction(() => {
            const mealResult = db.prepare(`
                INSERT INTO meal_entries
                (user_id, meal_type, meal_date, meal_time, note)
                VALUES (?, ?, ?, ?, ?)
            `).run(
                req.session.user.id,
                mealType,
                mealDate,
                mealTime,
                note || null
            );
```

```
            const mealId = mealResult.lastInsertRowid;

            const insertMealProduct = db.prepare(`
                INSERT INTO meal_products
                (meal_id, product_id, portion_grams, calculated_calories,
                 calculated_proteins, calculated_fats, calculated_carbs)
                VALUES (?, ?, ?, ?, ?, ?, ?)
            `);

            for (const item of products) {
                if (!item.productId || item.portionGrams <= 0) {
                    throw new Error("Некоректна порція продукту");
                }

                const product = db.prepare(`
                    SELECT id, calories_per_100g, proteins_per_100g,
                           fats_per_100g, carbs_per_100g
                    FROM products
                    WHERE id = ?
                `).get(item.productId);

                if (!product) {
                    throw new Error("Продукт не знайдено в базі даних");
                }

                const nutrition = calculateNutrition(product,
item.portionGrams);

                insertMealProduct.run(
                    mealId,
                    product.id,
                    item.portionGrams,
                    nutrition.calories,
                    nutrition.proteins,
                    nutrition.fats,
                    nutrition.carbs
                );
            }

            db.prepare(`
                INSERT INTO audit_log (user_id, action, entity_name, entity_id)
                VALUES (?, 'Додано прийом їжі', 'meal_entries', ?)
            `).run(req.session.user.id, mealId);

            recalculateDailySummary(req.session.user.id, mealDate);

            return mealId;
        });

        const mealId = createMealTransaction();

        res.status(201).json({
            message: "Прийом їжі додано",
            mealId
        });
    } catch (error) {
        res.status(500).json({
            error: error.message || "Помилка додавання прийому їжі"
        });
    }
});

router.get("/api/meals/:date", requireAuth, (req, res) => {
```

```
    const date = req.params.date;

    const meals = db.prepare(`
        SELECT id, meal_type, meal_date, meal_time, note
        FROM meal_entries
        WHERE user_id = ? AND meal_date = ?
        ORDER BY meal_time
    `).all(req.session.user.id, date);

    const getProducts = db.prepare(`
        SELECT p.name, mp.portion_grams, mp.calculated_calories,
               mp.calculated_proteins, mp.calculated_fats, mp.calculated_carbs
        FROM meal_products mp
        JOIN products p ON p.id = mp.product_id
        WHERE mp.meal_id = ?
        ORDER BY p.name
    `);

    const result = meals.map(meal => ({
        ...meal,
        products: getProducts.all(meal.id)
    }));

    res.json(result);
});

router.get("/api/dashboard/:date", requireAuth, (req, res) => {
    const date = req.params.date;

    recalculateDailySummary(req.session.user.id, date);

    const summary = db.prepare(`
        SELECT total_calories, total_proteins, total_fats,
               total_carbs, calories_difference
        FROM daily_summaries
        WHERE user_id = ? AND summary_date = ?
    `).get(req.session.user.id, date);

    const goal = db.prepare(`
        SELECT calories_goal, proteins_goal, fats_goal, carbs_goal
        FROM nutrition_goals
        WHERE user_id = ?
    `).get(req.session.user.id);

    const mealCount = db.prepare(`
        SELECT COUNT(*) AS total
        FROM meal_entries
        WHERE user_id = ? AND meal_date = ?
    `).get(req.session.user.id, date).total;

    res.json({
        date,
        mealCount,
        summary: summary || {
            total_calories: 0,
            total_proteins: 0,
            total_fats: 0,
            total_carbs: 0,
            calories_difference: goal ? goal.calories_goal : 0
        },
        goal
    });
});
```

```
router.get("/api/analytics/week", requireAuth, (req, res) => {
    const rows = db.prepare(`
        SELECT summary_date, total_calories, total_proteins,
               total_fats, total_carbs, calories_difference
        FROM daily_summaries
        WHERE user_id = ?
        ORDER BY summary_date DESC
        LIMIT 7
    `).all(req.session.user.id);

    const average = {
        calories: 0,
        proteins: 0,
        fats: 0,
        carbs: 0
    };

    if (rows.length > 0) {
        average.calories = Number((
            rows.reduce((sum, item) => sum + item.total_calories, 0) /
rows.length
        ).toFixed(2));

        average.proteins = Number((
            rows.reduce((sum, item) => sum + item.total_proteins, 0) /
rows.length
        ).toFixed(2));

        average.fats = Number((
            rows.reduce((sum, item) => sum + item.total_fats, 0) / rows.length
        ).toFixed(2));

        average.carbs = Number((
            rows.reduce((sum, item) => sum + item.total_carbs, 0) / rows.length
        ).toFixed(2));
    }

    res.json({
        period: "7 днів",
        days: rows.reverse(),
        average
    });
});

router.delete("/api/meals/:id", requireAuth, (req, res) => {
    const meal = db.prepare(`
        SELECT id, meal_date
        FROM meal_entries
        WHERE id = ? AND user_id = ?
    `).get(req.params.id, req.session.user.id);

    if (!meal) {
        return res.status(404).json({ error: "Прийом їжі не знайдено" });
    }

    db.prepare(`
        DELETE FROM meal_entries
        WHERE id = ? AND user_id = ?
    `).run(req.params.id, req.session.user.id);

    recalculateDailySummary(req.session.user.id, meal.meal_date);
```

```
    res.json({ message: "Прийом їжі видалено" });
});

module.exports = router;
```

ДОДАТОК Г

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

Декларація про академічну доброчесність та використання ШІ

ДЕКЛАРАЦІЯ ЗДОБУВАЧА

Я, Бобко Микола Михайлович, здобувач(ка) НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

1. Моя бакалаврська кваліфікаційна робота (проєкт) на тему «_Програмне забезпчення для моніторингу калорій та харчування_» є результатом моєї особистої праці.
2. Усі запозичення з текстів інших авторів мають відповідні посилання згідно з чинними стандартами.
3. Щодо використання інструментів ШІ зазначаю, що (обрати необхідне):
    - [ ] інструменти генеративного ШІ не використовувалися.
    - [+] інструменти ШІ (вказати назву: ChatGPT, Gemini, Claude, DeepL, _____інші (вказати назву)) були використані для:
        - [ ] перекладу іншомовних джерел;
        - [ ] стилістичного редагування та перевірки граматики;
        - [+] допомоги у написанні/відлагодженні програмного коду;
        - [ ] інше: _____.

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

«___» _____ 2026 р. _____ **Микола БОБКО**
(підпис)