Державний торговельно-економічний університет

Кафедра комп’ютерних наук та інформаційних систем

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Технологія розробки мобільного додатку для iOS»

Студента 2 курсу, 4м групи
спеціальності
122 «Комп’ютерні науки» _____ *підпис студента* Антонов Іван Володимирович

Науковий керівник
кандидат фізико-математичних наук _____ *підпис керівника* Філімонова Тетяна Олегівна

Гарант освітньої програми
доктор фізико-математичних наук, професор _____ *підпис керівника* Пурський Олег Іванович

Київ 2023

Державний торговельно-економічний університет

Факультет інформаційних технологій
Кафедра комп'ютерних наук та інформаційних систем
Спеціальність 122 «Комп'ютерні науки»
Спеціалізація «Комп'ютерні науки»

Затверджую
Зав. кафедри _____Пурський О.І.
«9» грудня 2022р.

Завдання
на випускну кваліфікаційну роботу студенту

Антонов Іван Володимирович
(прізвище, ім'я, по батькові)

1. *Тема випускної кваліфікаційної роботи*

«Технологія розробки мобільного додатку для iOS»
Затверджена наказом ректора від *«06» грудня 2022 р. № 3284*

2. Строк здачі студентом закінченої роботи *24 листопада 2023 року*

3. Цільова установка та вихідні дані до роботи
Мета роботи: Розробка технологій створення мобільного додатку для iOS.
Об'єкт дослідження: Процес розробки мобільного додатку для iOS.
Предмет дослідження: Засоби створення мобільного додатку для iOS.

4. Перелік графічного матеріалу_____
_____
_____
_____

5. Консультанти по роботі із зазначенням розділів, за якими здійснюється консультування:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Філімонова Т.О.	09.12.2022 р.	09.12.2022 р.
2	Філімонова Т.О.	09.12.2022 р.	09.12.2022 р.
3	Філімонова Т.О.	09.12.2022 р.	09.12.2022 р.

6. Зміст випускного кваліфікаційної роботи (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1. Розробка та реалізація мобільного додатку

1.1. Планування та проектування додатку

1.2. Архітектура додатку та вибір технологій

1.3. Інтеграція Core Data для управління даними

1.4. Розробка інтерфейсу користувача (UI/UX)

РОЗДІЛ 2. Аналіз процесу розрахунку вітамінних потреб для мобільного додатку

2.1. Огляд вітамінних потреб людини.

2.2. Фактори, що впливають на вітамінні потреби.

2.3. Методологія розрахунку вітамінних потреб.

2.4. Харчові джерела та додаткові вітамінні препарати.

РОЗДІЛ 3. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ ДЛЯ МОБІЛЬНОГО ДОДАТКУ

3.1. Обґрунтування вибору програмних засобів

3.2. Структура програмного забезпечення клієнтської сторони

3.3. Структура програмного серверної сторони

3.4. Керівництво користувачів мобільного додатку

3.5. Тестування мобільного додатку

ВИСНОВКИ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

7. Календарний план виконання роботи

№ Пор.	Назва етапів випускної кваліфікаційної роботи	Строк виконання етапів роботи	
		За планом	фактично
1	2	3	4
1	*Вибір теми випускної кваліфікаційної роботи*	*01.11.2022*	*01.11.2022*
2	*Розробка та затвердження завдання на*	*09.12.2022*	*09.12.2022*

	випускну кваліфікаційну роботу		
3	*Вступ*	*01.05.2022*	*01.05.2022*
4	*РОЗДІЛ 1. Розробка та реалізація мобільного додатку*	*14.06.2022*	*14.06.2022*
5	*Підготовка статті у збірник наукових статей магістрів*	*20.06.2022*	*20.06.2022*
6	*РОЗДІЛ 2. Аналіз процесу розрахунку вітамінних потреб для мобільного додатку*	*08.09.2022*	*08.09.2022*
7	*РОЗДІЛ 3. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ ДЛЯ МОБІЛЬНОГО ДОДАТКУ*	*20.10.2022*	*20.10.2022*
8	*Висновки*	*02.11.2022*	*02.11.2022*
9	*Здача випускної кваліфікаційної роботи на кафедру науковому керівнику*	*07.11.2022*	*07.11.2022*
10	*Попередній захист випускної кваліфікаційної роботи*	*17.11.2022*	*17.11.2022*
11	*Виправлення зауважень, зовнішнє рецензування випускної кваліфікаційної роботи*	*22.11.2022*	*22.11.2022*
12	*Представлення готової зшитої випускної кваліфікаційної роботи на кафедру*	*24.11.2022*	*24.11.2022*
13	*Публічний захист випускної кваліфікаційної роботи*	*За розкладом роботи ЕК*	

8. Дата видачі завдання «5» грудня 2022 р.

9. Керівник випускного кваліфікаційного проєкту Філімонова Т.О.
(прізвище, ініціали, підпис)

10. Гарант освітньої програми Пурський О.І.
(прізвище, ініціали, підпис)

11. Завдання прийняв до виконання студент Антонов І.В.
(прізвище, ініціали, підпис)

12. Відгук керівника випускної кваліфікаційної роботи

У випускній кваліфікаційній роботі досліджено технології розробки мобільного додатку для iOS. Результатом є повноцінний мобільний додаток, який демонструє високу продуктивність, надійність у роботі з даними та зручність використання. Робота оформлена згідно з вимогами. Поставлені завдання виконані. Вважаю, що випускна кваліфікаційна робота може бути допущена до захисту.

Керівник випускної кваліфікаційної роботи

_____

(підпис, дата)

13. Висновок про випускну кваліфікаційну роботу

Випускна кваліфікаційна робота студента _____

(прізвище, ініціали)

може бути допущена до захисту в екзаменаційній комісії.

Гарант освітньої програми_____ Пурський О.І.

(підпис, прізвище, ініціали)

Завідувач кафедри _____ Пурський О.І.

(підпис, прізвище, ініціали)

«_____»_____2022 р.

Анотація

Ця випускна кваліфікаційна робота присвячена розробці мобільного додатку для операційної системи iOS з використанням технології Core Data. Робота включає детальний огляд архітектури та ключових компонентів iOS, а також глибоке дослідження фреймворку Core Data, який є важливим інструментом для ефективного управління даними в мобільних додатках. В роботі представлені основні концепції та методи розробки, які були застосовані для створення функціонального та зручного для користувача інтерфейсу. Окрему увагу приділено процесу інтеграції Core Data для забезпечення ефективного зберігання та витягування даних. Також в роботі описані випробування та оптимізація додатку, а також стратегії забезпечення безпеки та конфіденційності даних. Ключові аспекти роботи охоплюють дизайн додатку, вибір технологій, реалізацію функціоналу та тестування продукту. Результатом цієї роботи є повноцінний мобільний додаток, який демонструє високу продуктивність, надійність у роботі з даними та зручність використання.

Ключові слова: Core Data, мобільний додаток, інтерфейс, конфіденційності даних, технології.

Anotation

This graduation thesis is devoted to the development of a mobile application for the iOS operating system using Core Data technology. The work includes a detailed overview of the architecture and key components of iOS, as well as an in-depth study of the Core Data framework, which is an important tool for effective data management in mobile applications.

The work presents the main concepts and development methods that were applied to create a functional and user-friendly interface. Particular attention is paid to the Core Data integration process to ensure efficient data storage and retrieval. The work also describes testing and optimization of the application, as well as strategies for ensuring data security and privacy. Key aspects of the work include application design, technology selection, functionality implementation and product testing. The result of this work is a full-fledged mobile application that demonstrates high performance, reliability in working with data and ease of use.

Keywords: Core Data, mobile application, interface, data privacy, technology.

Зміст

Оглавление

ВСТУП

У світі стрімкого технологічного розвитку та непередбачуваних реалій сучасного життя, питання здоров'я та догляду за собою займають важливе місце у щоденному житті. У цьому контексті мобільні додатки, спрямовані на вдосконалення нашого фізичного та психічного благополуччя, стають необхідністю та об'єктом загального інтересу.

Ця дипломна робота присвячена вивченню та аналізу процесу створення мобільного додатку для iOS, спрямованого на відстеження денної норми вітамінів та надання користувачам загальних відомостей про вітаміни та їх вплив на здоров'я. Із зростанням інтересу до здорового способу життя та правильного харчування, важливість такого додатку стає непередбачуваною та актуальною.

Метою даної дипломної роботи є розгляд ключових етапів та викликів у процесі створення мобільного додатку для iOS, з фокусом на функціоналі відстеження денної норми вітамінів та інформаційної підтримки щодо вітамінів. Здійснюючи глибокий аналіз методів розробки, архітектурних рішень та практик в області мобільної розробки, робота спрямована на визначення оптимальних стратегій та рекомендацій для ефективної реалізації концепції додатку.

Важливість подальшого дослідження та розробки мобільних додатків для забезпечення нашого організму всіма необхідними вітамінами та мікроелементами підкреслюється сучасним підходом до здорового способу життя. Ця робота має на меті допомогти розробникам, інженерам та зацікавленим особам в збагаченні своїх знань та вмінь у сфері мобільної розробки, роблячи акцент на реалізації інноваційного та корисного для суспільства мобільного додатку.

Завдяки дослідженню та розробці додатку відстеження вітамінів, робота визначається як важливий внесок у розвиток мобільних додатків для поліпшення якості нашого життя та забезпечення найвищих стандартів в області мобільної розробки.

Мета і завдання дослідження. Метою даного дослідження є розробка технологій створення мобільного додатку для iOS. Для досягнення поставленої мети необхідно було вирішити наступні **завдання:**

- провести комплексне аналітичне дослідження розробки мобільних додатків та визначити найбільш підходящі технології розробки для мобільного додатку для iOS. Врахувати аспекти швидкості, безпеки та сумісності з різними пристроями.;
- дослідити методи аналізу вітамінних потреб людини;
- провести докладне дослідження та аналіз потреб цільової аудиторії та визначити, які вітаміни та яка інформація про них є наибільш актуальною та корисною для користувачів.;
- розробити механізм для введення та відстеження денного споживання вітамінів користувачами, забезпечивши можливість редагування та виведення статистики;
- дослідити методи існуючих тестувань мобільних додатків та провести комплексне тестування додатку на різних пристроях та відстежити та виправити всі виявлені помилки.;
- розробити зручний та естетичний інтерфейс додатку, що надає зручний доступ до функціоналу відстеження вітамінів та інформації про них та забезпечити інтуїтивність використання для різних категорій користувачів.;
- забезпечити можливість отримання користувачами інформації про

вітаміни, їхні джерела та корисні властивості, використовуючи надійні джерела даних.;

Об'єкт дослідження: Системи відстеження харчування та самопочуття користувачів з використанням технології Core Data в мобільних додатках для iOS.

Предмет дослідження: моделі, методи та інформаційні технології для створення мобільного додатку для iOS.

Методи дослідження: Теоретичною основою дослідження є загальнонауковий аналітичний метод, а також системний підхід і праці провідних вчених розробників мобільних додатків. Для практичного вирішення поставлених задач використовувався Xcode для створення мобільного додатку.

Наукова новизна одержаних результатів полягає в інформації про розробку мобільного додатку, а також сам додаток, що забезпечує відстеження харчування та самопочуття користувачів, що є дуже актуально в наші часи.

Практичне значення. Додаток забезпечує можливість швидко та зручно планувати графік прийому вітамінів, що дозволяє користувачам оптимізувати свій час та зосереджуватися на інших аспектах здорового способу життя. А також розвитку технічних навичок і інновацій.

Публікації. Результати дослідження опубліковано у збірнику наукових статей студентів, які здобувають освітній ступінь магістра за спеціалізацією «Комп'ютерні науки» КНТЕУ на тему: «Технологія розробки мобільного додатку для iOS», 2023 р.

Структура та обсяг випускної кваліфікаційної роботи. Випускна кваліфікаційна робота складається із вступу, трьох розділів, висновків, свиску використаних джерел,1 додатка і містить 64 сторінки основного тексту, 12 рисунків і 3 таблиці.

РОЗДІЛ 1.
Розробка та реалізація мобільного додатку

1.1. Планування та проектування додатку

У цьому розділі ми розглянемо ключові аспекти планування та проектування мобільного додатку для платформи iOS. Процес планування включає визначення цілей додатку, розуміння цільової аудиторії та формулювання функціональних та нефункціональних вимог. Проектування зосереджується на створенні архітектури додатку, дизайні інтерфейсу користувача та виборі технологічного стеку.

Першим кроком у процесі розробки є визначення основної мети додатку та його цільової аудиторії. Це допомагає зрозуміти, які особливості та функції будуть найбільш значущими для користувачів. Наприклад, якщо додаток призначений для відстеження харчування, важливо зосередитись на функціях, які забезпечують легке введення та аналіз харчових даних.

Після визначення цілей та аудиторії наступний крок - це формулювання як функціональних, так і нефункціональних вимог. Функціональні вимоги описують, що саме додаток повинен робити, наприклад, розрахунок добової норми вітамінів. Нефункціональні вимоги включають аспекти, такі як продуктивність, безпека та сумісність з різними пристроями iOS.

Дизайн інтерфейсу користувача (UI) є критичним елементом у проектуванні додатку. Він повинен бути інтуїтивно зрозумілим, зручним та привабливим. Використання інструментів, таких як Sketch або Figma, допомагає створювати макети інтерфейсу та прототипи, які можна тестувати та оптимізувати.

Вибір правильної архітектури додатку є ключовим для забезпечення його масштабованості та підтримки. Модель MVC (Model-View-Controller) часто використовується у розробці iOS-додатків, оскільки вона сприяє чіткому розділенню логіки додатку від його інтерфейсу.

Останнім етапом у процесі проектування є вибір технологій та інструментів для розробки. Для iOS-додатків зазвичай використовується мова програмування Swift, а також інструменти та сервіси, такі як Xcode, UIKit, та Core Data для управління даними.

Ефективне планування та проектування - це основа створення успішного мобільного додатку для iOS. Воно включає ретельне визначення цілей, аудиторії, формулювання вимог, проектування інтерфейсу, вибір архітектури та технологій. Зосередження уваги на цих аспектах забезпечує створення додатку, який не тільки відповідає потребам користувачів, але й є технологічно ефективним та легко адаптованим до майбутніх розширень.

1.2 Архітектура додатку та вибір технологій

Цей розділ розглядає ключові компоненти архітектури та технологічні рішення для розробки мобільного додатку iOS, призначеного для відстеження добової норми вітамінів. Зроблений акцент на виборі архітектурного підходу, мов програмування, фреймворків, а також інструментів і сервісів, які забезпечать ефективність, масштабованість та зручність використання додатку.

Для розробки мобільного додатку iOS важливо вибрати архітектуру, яка сприяє чистоті коду, легкості тестування та підтримки. Популярними варіантами є MVC (Model-View-Controller), MVP (Model-View-Presenter), та MVVM (Model-View-ViewModel). У контексті цього додатку, MVVM може бути оптимальним вибором, оскільки він сприяє розділенню логіки бізнес-процесів від інтерфейсу користувача, що спрощує тестування та підтримку.

Swift - сучасна, безпечна та швидка мова програмування, яка є ідеальним вибором для розробки iOS-додатків. Вона забезпечує як високу продуктивність, так і зручність роботи з кодом. Щодо фреймворків, UIKit є стандартним вибором для створення інтерфейсу, але також можна розглянути SwiftUI, який забезпечує більш декларативний та візуально-орієнтований підхід до дизайну інтерфейсу.

З огляду на потребу відстеження та аналізу даних харчування, важливо вибрати ефективний спосіб управління даними. Core Data є міцним та гнучким рішенням для зберігання даних на пристрої iOS. Він забезпечує об'єктно-орієнтоване зберігання та можливість легкої інтеграції з iCloud для синхронізації даних між пристроями.

Оскільки додаток вимагатиме обробки значної кількості даних користувачів, важливо забезпечити високу продуктивність та безпеку.

Використання Grand Central Dispatch (GCD) для асинхронної роботи з даними та забезпечення безпеки даних за допомогою шифрування і засобів безпеки iOS є критично важливими.

Архітектура та технологічний вибір є фундаментальними для створення ефективного та надійного мобільного додатку. Використання MVVM, Swift, UIKit або SwiftUI, Core Data та інших інструментів та технологій дозволяє створити додаток, який не тільки забезпечує чудовий користувацький досвід, але й є безпечним, швидким та легко адаптованим до майбутніх оновлень або змін.

1.3. Інтеграція Core Data для управління даними

У цьому розділі ми детально розглянемо процес інтеграції фреймворку Core Data в нашому мобільному додатку для відстежування добової норми вітамінів. Core Data - це потужний інструмент для роботи з базами даних в додатках iOS, який спрощує збереження, оновлення та видалення даних.

Створення моделі даних

Першим кроком у використанні Core Data є створення моделі даних, яка визначає структуру даних, які нам потрібно зберігати. Наша модель даних буде включати вітаміни і інформацію про їхнє споживання. Ми будемо мати наступну структуру моделі даних:

Vitamin:

- name: Назва вітаміну (рядок)
- dailyIntake: Добова норма вітаміну (дійсне число)
- consumedAmount: Кількість спожитого вітаміну (дійсне число)

Створення CoreData Stack

Core Data Stack - це набір компонентів, які дозволяють нам здійснювати операції з базою даних. Основними компонентами CoreData Stack є:

- NSPersistentContainer: Це контейнер, який містить нашу модель даних та дозволяє нам звертатися до бази даних.
- NSManagedObjectContext: Об'єкт, який представляє контекст для роботи з даними.
- NSPersistentStoreCoordinator: Координатор, який управляє зберіганням даних.

Робота з Managed Objects

Managed Objects - це класи, які відповідають сутностям в нашій моделі даних. У нашому випадку, для сутності "Вітаміни" ми будемо мати відповідний клас Managed Object.

Core Data дозволяє легко виконувати операції з даними, такі як додавання нових записів, зчитування існуючих, оновлення та видалення їх. Ці операції можна виконати за допомогою відповідних методів інтерфейсу Managed Object Context.

У цьому розділі ми розглянули процес інтеграції Core Data для управління даними в нашому мобільному додатку для відстежування добової норми вітамінів. Інтеграція Core Data дозволить нам зручно та ефективно зберігати та управляти даними, що є ключовим компонентом нашого додатку.

1.4. Розробка інтерфейсу користувача (UI/UX)

Розробка інтерфейсу користувача (UI/UX) є надзвичайно важливою частиною процесу створення мобільного додатку. Цей підрозділ зосереджується на тому, як забезпечити зручний, ефективний та привабливий інтерфейс для кінцевих користувачів.

Першим етапом у розробці інтерфейсу є аналіз потреб і очікувань користувачів. Важливо з'ясувати, які завдання вони планують виконувати за допомогою додатку і яким чином вони б взаємодіяли з ним.

На цьому етапі створюються концепції та макети інтерфейсу, включаючи розташування елементів, кольорову палітру, шрифти та іконки. Мета - створити привабливий та легкий для розуміння інтерфейс.

Розробка взаємодії користувача означає створення способів навігації, функціональності та взаємодії з додатком таким чином, щоб користувачі могли легко виконувати свої завдання та отримувати задоволення від використання продукту.

Після реалізації інтерфейсу проводяться тестування з метою виявлення помилок, а також збору відгуків користувачів. Отримана інформація використовується для внесення покращень та оптимізації інтерфейсу.

Оскільки мобільні додатки запускаються на різних пристроях з різними розмірами екранів, важливо забезпечити адаптацію інтерфейсу для різних пристроїв та їх роздільних здатностей.

Для того, щоб додаток був доступний для всіх користувачів, важливо дотримуватися принципів доступності, які допоможуть людям з різними обмеженнями використовувати додаток без перешкод.

Детальний опис дизайну інтерфейсу, включаючи всі необхідні графічні ресурси, передається розробникам для реалізації.

Розробка інтерфейсу користувача є важливим етапом у створенні мобільного додатку, оскільки від неї залежить зручність і задоволення користувачів від використання продукту.

РОЗДІЛ 2.

Аналіз процесу розрахунку вітамінних потреб для мобільного додатку

2.1. Огляд вітамінних потреб людини.

Вітаміни - це життєво необхідні органічні речовини, які відіграють ключову роль у підтримці здоров'я та нормального функціонування організму людини. Вони беруть участь у багатьох фізіологічних процесах, включаючи метаболізм, здоров'я шкіри та зору, а також підтримку імунної системи. Більшість вітамінів не можуть бути синтезовані організмом і мають надходити з їжею або харчовими добавками.

Класифікація Вітамінів

Вітаміни поділяються на дві категорії: жиророзчинні та водорозчинні.

- Жиророзчинні вітаміни (A, D, E, K) накопичуються в жирових тканинах та печінці, забезпечуючи резерв, що може бути використаний організмом при потребі.
- Вітамін A важливий для зору, росту клітин, та імунної системи.
- Вітамін D регулює баланс кальцію та фосфору, підтримуючи кістки та зуби.
- Вітамін E захищає клітинні мембрани від оксидативного стресу.
- Вітамін K необхідний для згортання крові та кісткового метаболізму.
- Водорозчинні вітаміни (C та всі вітаміни групи B) мають більш короткий час перебування в організмі та потребують регулярного поповнення.
- Вітамін C відомий своїми антиоксидантними властивостями та роллю у синтезі колагену.

- Вітаміни групи B (B1, B2, B3, B5, B6, B7, B9, B12) важливі для енергетичного метаболізму, нервової системи та формування крові.

Вітаміни є критично важливими для численних біохімічних та фізіологічних процесів. Наприклад, вони відіграють роль у синтезі ДНК, регуляції гормонів, захисті від хвороб та в окислювальних процесах. Дефіцит або надмірне споживання вітамінів може призвести до серйозних здоров'єзберігаючих проблем.

Щоденні норми споживання вітамінів варіюються в залежності від багатьох факторів, включаючи вік, стать, вагітність, лактацію, а також спосіб життя та загальний стан здоров'я. Наприклад, вагітним жінкам та людям похилого віку часто потрібно більше певних вітамінів. Важливо підкреслити, що денні норми встановлюються як рекомендації, а не точні вказівки.

Забезпечення адекватного вітамінного харчування є фундаментальним для підтримки здоров'я. Більшість людей можуть отримати необхідні вітаміни через різноманітне та збалансоване харчування. Однак у певних випадках можуть знадобитися додаткові вітамінні добавки, особливо в ситуаціях дефіциту або підвищеної потреби. Важливо консультуватися з медичними фахівцями для визначення індивідуальних потреб та уникнення ризику передозування вітамінів.

2.2. Фактори, що впливають на вітамінні потреби.

Вітамінні потреби людини варіюються в залежності від різноманітних факторів. Розуміння цих факторів дозволяє краще визначати індивідуальні потреби у вітамінах та оптимізувати харчування для підтримки здоров'я.

Індивідуальні Характеристики

1. Вік: Вітамінні потреби еволюціонують від дитинства до старості. Діти та підлітки потребують більше вітамінів для росту та розвитку. У літньому віці збільшується потреба у вітаміні D через зниження здатності шкіри синтезувати цей вітамін та у вітаміні B12 через зниження його засвоєння.

2. Стать: Жінки мають особливі вітамінні потреби, особливо під час вагітності та годування груддю, коли потребують більше фолієвої кислоти, вітамінів D, C, та B. Чоловіки, з іншого боку, можуть потребувати більше вітамінів, що підтримують м'язовий та клітинний ріст.

3. Стан Здоров'я та Хронічні Захворювання: Певні медичні стани, такі як захворювання кишківника, цукровий діабет, чи проблеми з серцем, можуть впливати на засвоєння вітамінів та мінералів, вимагаючи спеціалізованого підходу до дієти.

Зовнішні Фактори

1. Дієта та Харчування: Вегетаріанська або веганська дієта може вимагати додаткової уваги до вітамінів, які зазвичай знаходяться в продуктах

тваринного походження. Дієти, що обмежують споживання певних груп продуктів, можуть також впливати на загальний вітамінний баланс.

2. Спосіб Життя та Фізична Активність: Фізично активні люди можуть мати збільшені потреби у вітамінах, пов'язаних з відновленням м'язів та управлінням енергією. Куріння, алкоголь та інші звички також мають вплив на вітамінний метаболізм.

3. Екологічні Фактори: Сезонність та географічне положення можуть впливати на доступність та різноманітність продуктів харчування, багатих на вітаміни. Сонячне світло є ключовим фактором для синтезу вітаміну D.

Визначення індивідуальних вітамінних потреб вимагає уваги до цілого ряду факторів. Важливо, щоб дієта була збалансованою та різноманітною, щоб забезпечити всі необхідні вітаміни. У деяких випадках може бути корисним звернення до дієтолога або лікаря для розробки індивідуального плану харчування або прийняття вітамінних добавок.

2.3. Методологія розрахунку вітамінних потреб.

Розрахунок вітамінних потреб вимагає ретельного аналізу та врахування різноманітних факторів. Від точності цих розрахунків залежить не лише загальний стан здоров'я людини, але й її здатність ефективно протистояти різноманітним захворюванням.

Традиційні Методики

- Рекомендовані Денні Норми (РДН): Ці норми засновані на епідеміологічних даних та клінічних дослідженнях, які визначають середню потребу вітамінів для різних демографічних груп. РДН враховують середній рівень споживання, який вважається достатнім для підтримки здоров'я більшості людей в певній групі.
- Біохімічні Показники: Вимірювання рівнів вітамінів у крові, сечі або тканинах дає можливість оцінити стан вітамінного забезпечення конкретної особи. Цей метод може виявити не лише очевидні недоліки, але й приховані дефіцити вітамінів.

Індивідуалізовані Підходи

- Дієтичний Опитувальник: Систематичний аналіз дієтичних звичок, включаючи частоту та розмаїтість споживаних продуктів. Це дозволяє оцінити приблизне споживання вітамінів і виявити потенційні недоліки або надлишки.
- Аналіз Життєвого Стилю: Включає оцінку фізичної активності, звичок (таких як куріння та вживання алкоголю), а також індивідуальних особливостей метаболізму. Це важливо, оскільки ці фактори можуть суттєво впливати на потреби в вітамінах.

Моделі Передбачення

- Математичне Моделювання: Розробка математичних моделей для аналізу впливу різних факторів на вітамінні потреби. Це може включати моделі, які враховують вік, стать, вагу, рівень фізичної активності та інші змінні.
- Інтерактивні Мобільні Додатки: Розробка додатків, які дозволяють користувачам вводити свої персональні дані та отримувати індивідуальні рекомендації. Ці інструменти можуть включати алгоритми, що адаптуються до змін у дієті або способі життя.

Вплив Новітніх Досліджень

- Генетичні Фактори: Врахування генетичних особливостей, які можуть впливати на метаболізм вітамінів. Деякі люди мають генетичні варіації, які змінюють їх потреби в певних вітамінах.
- Епігенетичні Зміни: Дослідження впливу дієти на епігенетику та роль вітамінів у цих процесах. Ці зміни можуть впливати на експресію генів і, як наслідок, на потреби в певних вітамінах.

Методологія розрахунку вітамінних потреб є багатогранною та вимагає інтеграції традиційних та сучасних підходів. Розвиток технологій та новітніх досліджень відкриває шляхи для більш точного та індивідуалізованого визначення вітамінних потреб, що може суттєво покращити якість життя та здоров'я людей.

2.4. Харчові джерела та додаткові вітамінні препарати.

Забезпечення організму достатньою кількістю вітамінів є ключовим фактором для підтримки здоров'я та благополуччя. Харчові джерела та додаткові вітамінні препарати грають важливу роль у задоволенні цих потреб.

Харчові Джерела Вітамінів

1. Фрукти та Овочі: Багаті на вітаміни C, A, K, і більшість вітамінів групи B. Наприклад, цитрусові є джерелом вітаміну C, тоді як морква та батат містять вітамін A.

2. Цільнозернові Продукти: Містять вітаміни групи B, важливі для метаболізму. Цільнозернові продукти, такі як коричневий рис та овес, є відмінними джерелами цих вітамінів.

3. Білки: М'ясо, риба, яйця та молочні продукти є джерелами вітамінів B12 та D. Вегетаріанські джерела білка, такі як бобові, також містять вітаміни групи B.

4. Горіхи та Насіння: Являють собою джерела вітамінів E та B. Наприклад, мигдаль та соняшникове насіння містять вітамін E.

Використання Додаткових Вітамінних Препаратів

1. Дефіцит Вітамінів: Вітамінні добавки часто використовуються для запобігання або лікування дефіцитів вітамінів, особливо коли неможливо забезпечити потреби через дієту.

2. Специфічні Групи Населення: Вагітні жінки, люди похилого віку, ті, хто дотримується певних дієт (наприклад, веганство), та особи з медичними станами, що впливають на засвоєння вітамінів, можуть мати підвищену потребу в додаткових вітамінах.

3. Ризики та Міркування: Важливо пам'ятати, що надмірне споживання вітамінів, особливо жиророзчинних, може призвести до токсичності. Тому вживання добавок повинно відбуватися під контролем медичного фахівця.

Синергія Харчових Джерел та Добавок

- Збалансоване Харчування як Основа: Оптимальний підхід полягає у використанні збалансованої дієти як основного джерела вітамінів, з добавками, що використовуються як доповнення при специфічних потребах.

- Індивідуальний Підхід: Визначення оптимальної комбінації харчових джерел та добавок має бути індивідуалізованим, враховуючи дієтичні звички, стиль життя та здоров'я особи.

Правильний вибір харчових джерел та розумне використання вітамінних добавок є ключовими для забезпечення достатнього вітамінного забезпечення організму. Важливо вести збалансований спосіб життя, звертаючись за порадами до медичних фахівців щодо використання вітамінних добавок для забезпечення оптимального здоров'я та благополуччя.

РОЗДІЛ 3.
ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ ДЛЯ МОБІЛЬНОГО ДОДАТКУ

3.1 Обґрунтування вибору програмних засобів

З поміж доступної кількості програмних засобів, які можуть бути використані для реалізації мобільного iOS додатку, в даній роботі були обрані та використані мова програмування Swift; набір бібліотек та фреймворків UIKit, SwiftUI та Cocoa Touch, також фреймворк Core Data, середовище розробки Xcode, GitHub для контролю версій, Kingfisher для завантаження зображень, Interface Builder для створення інтерфейсу користувача, засоби тестування Інструменти для автоматизованого та ручного тестування, такі як XCTest та різні фреймворки для UI-тестування, Figma дизайн-інструмент. Дальше детальніше буде обґрунтовано чим зумовлено такий вибір.

Вибір мови програмування Swift, може бути обґрунтовано наступними причинами:

- Популярність: Swift є дуже популярною мовою серед розробників, які працюють з платформами Apple. Це означає, що ви зможете знайти багато ресурсів, бібліотек та фреймворків для роботи з цією мовою.
- Сучасність і актуальність: Swift є однією з найбільш сучасних мов програмування, розроблених компанією Apple. Вона постійно оновлюється і розвивається, що робить її актуальною для розробки на платформах Apple, таких як iOS, macOS, watchOS і tvOS.

- Зручність і читабельність коду: Swift була створена з метою полегшити роботу розробників і зробити код більш зрозумілим та читабельним. Вона має простий і чіткий синтаксис, що сприяє швидкому розробленню програм та утриманню коду.
- Підтримка багатьох платформ: Swift дозволяє розробникам писати код, який може бути використаний на різних платформах Apple, зменшуючи таким чином затрати на розробку.
- Інноваційні можливості: Swift має численні інновації, такі як безпека типів, гаряча заміна коду під час роботи додатку (live code updates), інтеграція з Objective-C і іншими мовами програмування.
- Зростаюча спільнота розробників: З роками спільнота розробників Swift зростає і активно співпрацює над створенням різних інструментів і бібліотек, що полегшують роботу розробників.
- Підтримка компанією Apple: Apple активно підтримує розвиток Swift і надає засоби для розробки програм для своїх платформ, що робить її природним вибором для розробки додатків на цих платформах.

Вибір UIKit може бути обгрунтований з рядом практичних і технічних причин:

- Широка популярність та прозорість: UIKit є однією з найпоширеніших і найбільш досліджених технологій для розробки iOS-додатків. Вона має довгу історію та велику спільноту розробників, що робить її стабільною і надійною платформою для дипломної роботи.

- Зрозумілість інструментів: Якщо ви вже маєте досвід роботи з Swift або Objective-C, вам буде легше розпочати роботу з UIKit. Це дозволить вам сконцентруватися на розробці функціональності вашої дипломної роботи, а не на вивченні нової технології.

- Контроль над дизайном: UIKit надає вам великий контроль над дизайном і поведінкою інтерфейсу користувача. Це особливо важливо, якщо ваша дипломна робота передбачає розробку складного інтерфейсу або вимагає специфічних візуальних ефектів.

- Підтримка для старших версій iOS: UIKit підтримує більшість старших версій iOS, що означає, що ваша дипломна робота буде доступною для більшого кола користувачів.

- Широка функціональність: UIKit має багатий набір компонентів і функціональності, що полегшує розробку різноманітних додатків. Ви можете легко використовувати таблиці, колекції, тексти, анімацію та інші компоненти для створення багатофункціональних додатків.
  Робота зі сторонніми бібліотеками: UIKit добре інтегрується зі сторонніми бібліотеками і фреймворками, що дозволяє використовувати готові рішення для реалізації певних функцій в вашій дипломній роботі.

Вибір SwiftUI для мобільного додатку можна обгрунтувати за допомогою декількох факторів:

- Сучасність та майбутність: SwiftUI є новим інструментом для розробки інтерфейсу для мобільних додатків від Apple. Він представлений у iOS 13 і вже отримав підтримку в наступних версіях операційної системи. Використання сучасних технологій може зробити вашу дипломну роботу більш актуальною та цікавою для оцінювачів.

- Простота та продуктивність: SwiftUI пропонує декларативний підхід до створення інтерфейсу, що робить код більш зрозумілим і легким для написання. Ви можете швидко створювати інтерфейс за допомогою вбудованих властивостей та компонентів, що зменшує час розробки і дозволяє зосередитися на інших аспектах вашого додатку.

- Інтеграція з іншими технологіями Apple: SwiftUI добре інтегрується з іншими технологіями Apple, такими як Swift, Combine, і Core Data. Це дозволяє легко працювати з даними, взаємодіяти з серверами та використовувати різноманітні функціональні можливості, які можуть знадобитися вам для дипломної роботи.

- Можливості анімації та взаємодії: SwiftUI надає потужні можливості для створення анімацій та інтерактивних інтерфейсів, що може бути корисним для демонстрації концепцій та ідей у вашій дипломній роботі.

Cocoa Touch - це фреймворк для створення програм під iPhone, iPod touch, і iPad.

Бібліотека Cocoa Touch надає рівень абстракції для iOS (операційної системи iPhone, iPad та iPod touch). Cocoa Touch заснована на класах фреймворку Cocoa, що використовується в Mac OS X, і, аналогічно до неї, використовує мову Objective-C. Cocoa Touch слідує шаблону проектування Model-View-Controller.

iOS-технології можна розглядати як набір шарів, де Cocoa Touch знаходиться на найвищому рівні, а Core OS та ядро macOS – на нижчих. Це дозволяє реалізовувати багато складних завдань, скорочуючи обсяг роботи, яку довелося б виконувати розробникам, працюй вони на нижчому рівні. Тим не менш, деякі низькі шари абстрагування можуть бути доступні розробникам при необхідності.

Основні особливості Cocoa Touch для iOS включають в себе:

- UIKit: UIKit - це один з основних фреймворків, який надає інструменти для створення користувацького інтерфейсу додатків на iOS. Він містить класи та компоненти, такі як UIView, UIViewController, UITableView і багато інших, які допомагають створювати інтерактивний та зручний для використання інтерфейс.

- Foundation: Foundation - це фреймворк, який містить базові класи для обробки даних та роботи з файлами, мережею, датами і багато іншого. Він є основою для багатьох інших фреймворків і забезпечує роботу з даними та логікою програми.

- Core Data: Core Data - це фреймворк для роботи з базами даних в додатках iOS. Він дозволяє створювати, зчитувати, оновлювати та видаляти дані з бази даних, що забезпечує ефективне управління даними в додатках.

- Core Animation: Core Animation дозволяє створювати анімацію та візуальні ефекти в інтерфейсі додатків. Він надає інструменти для створення різноманітних анімаційних переходів та відображення графічних об'єктів.
- Networking: Для роботи з мережею в iOS використовується бібліотека URLSession, яка дозволяє здійснювати HTTP-запити, завантажувати та відправляти дані через мережу.

Ці компоненти Cocoa Touch допомагають розробникам створювати різноманітні та потужні мобільні додатки для iOS, забезпечуючи широкі можливості для роботи з інтерфейсом, даними, анімацією та мережею. Вибір Cocoa Touch є стандартним підходом для розробки додатків на платформі iOS.

Вибір Core Data дипломній роботі можна обгрунтувати з різних причин:

- Легкість і швидкість розробки: Core Data надає інтерфейс для роботи з базами даних, який значно спрощує створення, зчитування, оновлення та видалення даних в додатку. Ви можете створювати моделі даних за допомогою графічного інтерфейсу Xcode, що робить розробку швидкою і зручною.
- Робота з об'єктами: Core Data використовує модель даних, яка дозволяє вам робити операції з об'єктами, подібними до тих, що ви використовуєте в коді додатку. Це полегшує взаємодію з даними та сприяє їх більш зрозумілому та безпечному використанню.
- Можливість структурувати дані: Core Data дозволяє структурувати дані у вигляді сутностей та встановлювати взаємозв'язки між ними.

Це корисно для зберігання та взаємодії зі складними даними, такими як користувачі, завдання, записи тощо.

- Підтримка версіонування та міграції: Core Data надає механізми для версіонування бази даних та автоматичної міграції даних при змінах в структурі моделі. Це дозволяє вам легко розширювати та оновлювати додаток без втрати даних.
- Інтеграція з іншими фреймворками: Core Data добре інтегрується з іншими фреймворками, такими як UIKit, SwiftUI, і CloudKit, що дозволяє використовувати його в різних аспектах додатку.
- Підтримка офлайн-режиму: Core Data дозволяє зберігати дані локально на пристрої, що робить його ідеальним вибором для додатків, які мають підтримувати роботу без інтернет-з'єднання.

Усі ці переваги роблять Core Data потужним інструментом для роботи з даними в мобільних додатках.

Середовище розробки Xcode — інтегроване середовище розробки (IDE) виробництва Apple. Дозволяє створювати програмне забезпечення з використанням таких технологій як GCC, GDB, Java та ін. На сьогодні є єдиним засобом написання «універсальних»((Universal Binary)) прикладних програм для Mac OS X.

Xcode включає в себе більшу частину документації розробника від Apple та (Interface Builder) — застосунок, який використовується для створення графічних інтерфейсів.

Пакет Xcode містить змінену версію вільного набору компіляторів GNU Compiler Collection і підтримує мови C, , Objective-C, Swift, Java, AppleScript, Python і Ruby з різними моделями програмування, включаючи (але не обмежуючись) Cocoa, Carbon і Java. Сторонніми розробниками реалізована підтримка (GNU Pascal), Free Pascal, Ada, C #, Perl, Haskell і D. Пакет Xcode використовує GDB як back-end для свого налагоджувача.

Обгрунтуванням вибору данного інтерфейсу послугують такі фактори:

- Офіційне середовище розробки для iOS: Xcode є офіційним інтегрованим середовищем розробки (IDE) від Apple для розробки додатків під iOS та macOS. Використання офіційного інструменту може покращити зручність та надійність вашої розробки.
- Набір інструментів: Xcode надає широкий набір інструментів для розробки, включаючи візуальний редактор інтерфейсу, налаштування емулятора для тестування додатків, інструменти для налагодження коду та багато інших корисних ресурсів.
- Мова програмування Swift: Xcode найкраще підтримує розробку мовою програмування Swift, яка є офіційною мовою для розробки додатків на платформі iOS. Вона володіє чистим і зрозумілим синтаксисом, що полегшує розробку та підтримку коду.
- Інтеграція з іншими технологіями Apple: Xcode добре інтегрується з іншими технологіями Apple, такими як Interface Builder для створення інтерфейсу, Instruments для профілювання та оптимізації додатків, а також Core Data для роботи з базами даних.

- Підтримка додатків для iOS та macOS: Xcode дозволяє розробляти додатки як для мобільних пристроїв під управлінням iOS, так і для комп'ютерів під управлінням macOS. Це може бути корисним, якщо ви хочете розробити додаток для обох платформ.

- Безкоштовність: Xcode є безкоштовним інструментом, доступним для завантаження з офіційного веб-сайту Apple. Це дозволяє вам легко отримати доступ до потужного середовища розробки без витрат.

GitHub — один з найбільших вебсервісів для спільної розробки програмного забезпечення. Існують платні та безплатні тарифні плани користування. Базується на системі керування версіями Git і розроблений на Ruby on Rails і Erlang компанією GitHub, Inc (спочатку вона називалася Logical Awesome).

Ми будемо за допомогою GitHub створювати Backapp застосунку, а також це дозволить спростити розробку та контролювати її.

Вибір GitHub я обгрунтую наступним чином:

- Контроль версій: GitHub надає потужні засоби для керування версіями вашого програмного коду. Ви можете створювати резервні копії коду, відстежувати зміни і відновлювати попередні версії. Це корисно під час виправлення помилок та розвитку проекту.

- Спільна робота: GitHub дозволяє спільно працювати над проектом з іншими розробниками. Ви можете легко спільно додавати, виправляти та оглядати код разом з колегами або науковими керівниками.

- Відкритий доступ: Ви можете зробити ваш репозиторій на GitHub публічним або приватним, в залежності від вашої потреби. Якщо ви

хочете, щоб інші студенти, викладачі або потенційні роботодавці могли переглядати вашу роботу, це легко налаштовується.

- Інструменти спільної роботи: GitHub має вбудовані інструменти для ведення обговорень (issues) та ведення списку завдань (project boards), що полегшує організацію та спілкування в команді.
- Інтеграція з іншими інструментами: GitHub легко інтегрується з багатьма іншими інструментами розробки, такими як Travis CI, CircleCI, а також іншими сервісами для автоматизації тестування та розгортання.
- Портфоліо: GitHub може служити вам як вірний спосіб показати вашу роботу та навички потенційним роботодавцям. Ви можете включити посилання на ваш репозиторій GitHub у своєму резюме або відправити його в університет як частину вашої дипломної роботи.
- Безкоштовність: GitHub пропонує безкоштовні облікові записи для проектів з відкритим вихідним кодом та студентів. Це дозволяє вам використовувати багато основних функцій безкоштовно.

Kingfisher - це популярна бібліотека для завантаження та кешування зображень в мобільних додатках на платформі iOS (Swift) та macOS. Вона надає розробникам простий спосіб завантажувати зображення з веб-серверів, кешувати їх локально для подальшого використання та відображати в інтерфейсі додатку.

Основні переваги Kingfisher включають:

- Простоту використання: Kingfisher надає легкий та інтуїтивний інтерфейс для завантаження та відображення зображень без необхідності писати багато коду.
- Кешування зображень: Бібліотека автоматично кешує завантажені зображення, що дозволяє швидше завантаження та зменшує використання мережі.
- Підтримка веб-серверів: Kingfisher може працювати з різними типами веб-серверів і підтримує завантаження зображень з HTTP і HTTPS.
- Асинхронні завантаження: Завантаження зображень відбувається асинхронно, що дозволяє уникнути блокування інтерфейсу користувача під час завантаження.
- Підтримка кількох форматів: Kingfisher підтримує різні формати зображень, такі як JPEG, PNG, GIF, а також підтримує анімовані GIF.
- Широкий вибір параметрів конфігурації: Розробники можуть налаштовувати різні параметри завантаження та відображення зображень, включаючи розмір, якість, кешування і багато інших.

Interface Builder - це інструмент для розробки графічного інтерфейсу користувача в середовищі розробки програмного забезпечення для платформи Apple, таких як iOS, macOS, watchOS та tvOS. Цей інструмент дозволяє розробникам створювати інтерфейси додатків без програмування в коді.

Основні риси та функції Interface Builder включають:

- Візуальна розробка: Interface Builder надає візуальний інтерфейс, в якому ви можете створювати інтерфейс користувача, перетягуючи та розміщаючи елементи інтерфейсу, такі як кнопки, тексти, зображення і багато інших.
- Редагування в реальному часі: Ви можете бачити, як зміни відображаються в інтерфейсі в реальному часі, що дозволяє швидко вирішувати проблеми з розміщенням та виглядом елементів.
- Взаємодія з кодом: Interface Builder інтегрований з мовами програмування для платформ Apple, такими як Swift та Objective-C. Ви можете легко зв'язувати інтерфейсні елементи з кодом, щоб реалізувати функціональність додатка.
- Підтримка Auto Layout: Interface Builder підтримує Auto Layout, що дозволяє створювати адаптивні інтерфейси, які підлаштовуються під різні розміри екранів і орієнтації.
- Перевірка доступності: Інструмент також надає можливість перевіряти доступність вашого інтерфейсу для людей з обмеженими можливостями та робити необхідні покращення.
- Попередній перегляд на різних пристроях: Ви можете переглядати ваш інтерфейс на різних пристроях (iPhone, iPad, Mac і т. д.) та в різних режимах, щоб переконатися, що він виглядає належним чином.

Interface Builder є незамінним інструментом для розробки інтерфейсу користувача для додатків для платформи Apple. Він спрощує процес розробки інтерфейсу та дозволяє розробникам швидко створювати зручні та естетично привабливі додатки.

Figma - це веб-платформа для дизайну і прототипування, яка призначена для роботи над проектами інтерфейсу користувача (UI) та інших графічних дизайнів. Цей інструмент дозволяє командам дизайнерів та розробників спільно працювати над проектами в реальному часі через Інтернет. Основні риси та функції Figma включають:

- Веб-платформа: Figma використовується через веб-браузер, тобто вам не потрібно встановлювати жодних додатків на свій комп'ютер. Це робить платформу досить доступною та зручною для користування.
- Спільна робота в реальному часі: Декілька користувачів можуть працювати над одним та самим проектом одночасно. Зміни відображаються миттєво для всіх учасників, що дозволяє ефективно спільно працювати над проектами навіть на великих відстанях.
- Дизайн і прототипування: Ви можете створювати макети інтерфейсів користувача, макети сторінок, інтерактивні прототипи, а також визначати переходи між сторінками і взаємодію елементів.
- Бібліотеки та компоненти: Figma дозволяє створювати бібліотеки компонентів та стилів, що спрощує повторне використання та забезпечує однорідний дизайн в усьому проекті.
- Зберігання та синхронізація: Всі ваші проекти зберігаються в хмарі, що дозволяє з легкістю отримувати доступ до них з будь-якого пристрою та забезпечує автоматичну синхронізацію.
- Інтеграція з іншими інструментами: Figma може інтегруватися з іншими інструментами для дизайну та розробки, такими як Sketch, Adobe XD, Zeplin, JIRA, і багатьма іншими.

- Коментування та обговорення: Ви можете додавати коментарі та залишати відгуки прямо в макетах, що полегшує комунікацію та зворотний зв'язок в команді.

XCTest - це фреймворк для тестування програмного забезпечення на платформі Apple, таких як iOS, macOS, watchOS та tvOS. Цей фреймворк надає розробникам інструменти для автоматизованого тестування їх програм для забезпечення їх надійності та відповідності очікуванням.

Деякі основні функції і можливості XCTest включають:

- Автоматизовані тести: XCTest дозволяє розробникам створювати автоматизовані тести для перевірки функціональності своїх додатків. Тести можуть бути написані на мовах програмування, таких як Swift або Objective-C.
- Уніт-тести: XCTest підтримує уніт-тести, які дозволяють перевіряти окремі компоненти або функції програми на відповідність очікуванням. Це сприяє виявленню та виправленню помилок на ранніх стадіях розробки.
- Інтеграція з Xcode: XCTest інтегрований з середовищем розробки Xcode, що робить створення, запуск та аналіз результатів тестів дуже зручним.
- Підтримка визначення очікуваних результатів: Ви можете визначати очікувані результати для вашого коду і перевіряти, чи вони співпадають з фактичними результатами в ході тестування.
- Відлагодження тестів: XCTest дозволяє відлагоджувати тести, щоб знайти та виправити проблеми в вашому коді.

- Тести в паралель: Ви можете виконувати тести в паралель, що дозволяє прискорити процес тестування.

3.2 Структура програмного забезпечення клієнтської сторони

Розглянемо підсистему «frontEnd», його структурна схема представлена на рисунку 3.1.

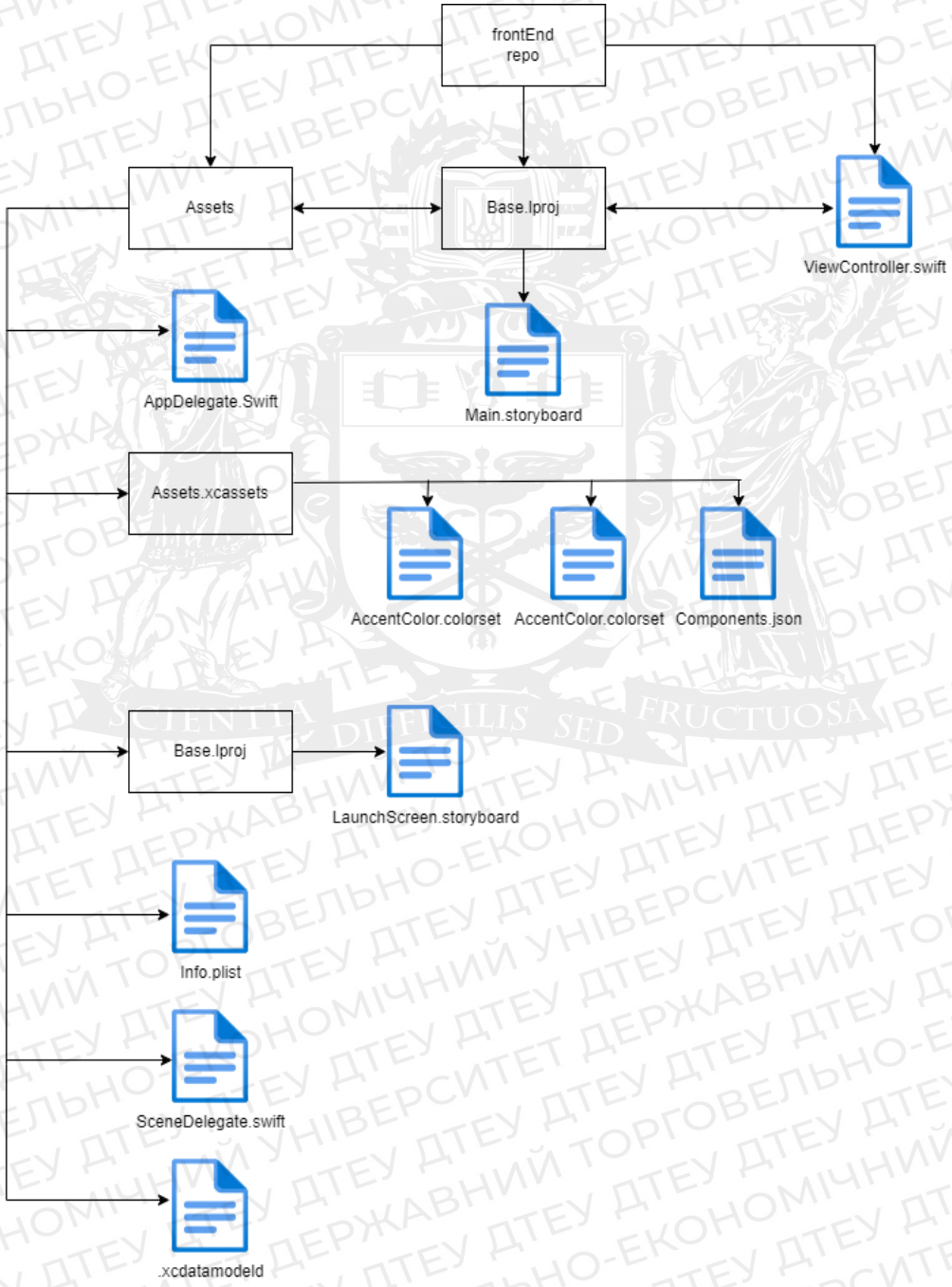


Рисунок 3.1 — Структурна схема підсистему «frontEnd»

З рисунку видно, що вона складається з репозиторію `Assets`, `Base.lproj` та та файлу користувацького інтерфейсу — `ViewController.swift`. В свою чергу `Base.lproj містить в собі файл інтерфейсу користувацької програми — `Main.storyboard`. Папка `Assets` містить статичні ресурси, такі як зображення, шрифти, стилі; Папка `Assets.xcassets` містить в собі файли в яких зберігаються данні кольору та компонентів `AccentColor.colorset`, `Components.json`; `AppDelegate.Swift` - це файл в програмах для платформи iOS і macOS, який містить клас AppDelegate. Цей клас відповідає за управління життєвим циклом додатка і взаємодію з системними подіями та подіями, що відбуваються під час запуску, виконання та закриття додатка; папка `Base.lproj` містить в собі файл інтерфейсу для платформи iOS — `LaunchScreen.storyboard`; Також в папці `Assets` містяться конфігураційний файл `Info.plist`, файл відповідає за керування сценами (scenes) і взаємодію з окремими вікнами та інтерфейсами вашого додатка `SceneDelegate.swift`, файл даних для моделі даних Core Data у програмах для платформи iOS і macOS `.xcdatamodeld`.

Розглянемо детальніше основні програмні модулі та структуруємо це наступною таблицею 3.1.

Таблиця 3.1 — Специфікація програмних модулів підсистеми «frontEnd»

Модуль	Пояснення
`Base.lproj`	
`Main.storyboard`	Файл інтерфейсу користувацької програми в середовищі розробки Xcode для платформи iOS і macOS. В

	цьому файлі визначається графічний інтерфейс програми, включаючи вигляд екранів, взаємодію з користувачем та переходи між різними частинами програми.
`ViewController.swift`	файл коду в програмах для платформи iOS (iPhone/iPad) і macOS, який зазвичай містить визначення класу ViewController. Цей клас відповідає за логіку відображення та взаємодії з користувачем на екрані додатка.
`Assets`	
`AppDelegate.Swift`	Цей клас відповідає за управління життєвим циклом додатка і реагує на системні події, які відбуваються під час його запуску, виконання та закриття.
`Info.plist`	Цей файл містить різні метадані та налаштування для вашого додатка, такі як ідентифікатор додатка, версія, дозволи, список підтримуваних пристроїв, обмеження на роботу з мережею, доступ до привілеїв, настройки вигляду додатка та багато

	іншого.
`SceneDelegate.swift`	Цей файл містить клас **SceneDelegate**, який відповідає за керування сценами (scenes) і взаємодію з окремими вікнами та інтерфейсами вашого додатка.
`Base.lproj`	
`LaunchScreen.storyboard`	Цей файл містить графічний інтерфейс, який коротко показується користувачеві під час завантаження додатка, перед відображенням головного інтерфейсу.
`Assets.xcassets`	
`AccentColor.colorset`	Це каталог (директорія) у проекті для платформи iOS, який містить налаштування кольорів, пов'язаних із відображенням акцентних (highlight) кольорів в інтерфейсі вашого додатка.
`Components.json`	Зазвичай є файлом конфігурації або даними у форматі JSON, який містить інформацію про компоненти, що використовуються в програмному забезпеченні або додатку. Зазвичай цей файл має важливе значення для

	розробки та налаштування додатка або системи.
`AppDelegate.Swift`	Цей клас відповідає за керування життєвим циклом додатка і реагує на системні події, які відбуваються під час його запуску, виконання та закриття.

Маючи опис усіх файлів додатку, необхідно показати як у додатку відбувається перехід між екранами, це можна зробити за допомогою графа переходу, який зображений на рисунку 3.2.

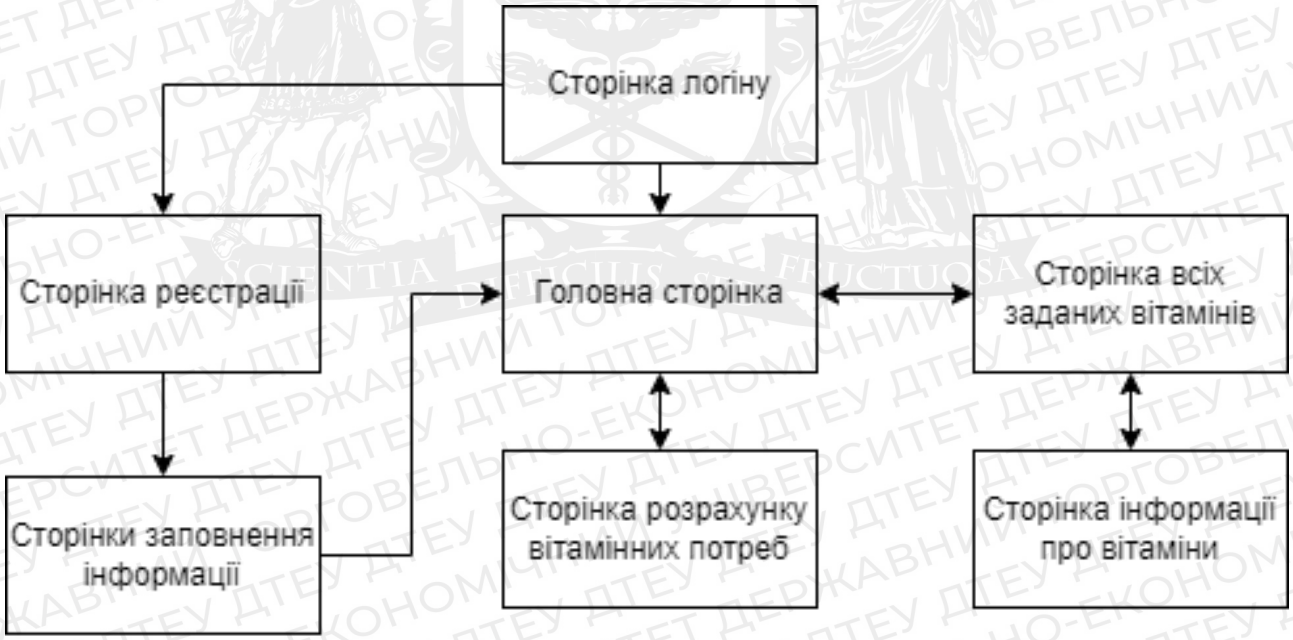


Рисунок 3.2 —Граф переходу між екранами додатку

Даний рисунок структурує загальну картину мобільного додатку. Кожен вузол в графі представляє екран мобільного додатку, а ребра між вузлами представляють допустимі переходи між екранами.

3.3 Структура програмного серверної сторони

Структура програмного забезпечення серверної сторони визначається компонентами, які відповідають за різні аспекти роботи сервера. Основними компонентами серверної сторони для проекта на Swift з використанням фреймворку CoreData є додаток, маршрутизація, база даних, системи аутентифікації та авторизації користувачів, а також різноманітні сервіси для роботи з іншими системами. Ці компоненти дозволяють створити функціональну та безпечну систему для обробки запитів від клієнтів та зберігання данних.

Розпочнемо з огляду підсистеми «backEnd», його структурна схема представлена на рисунку 3.3.

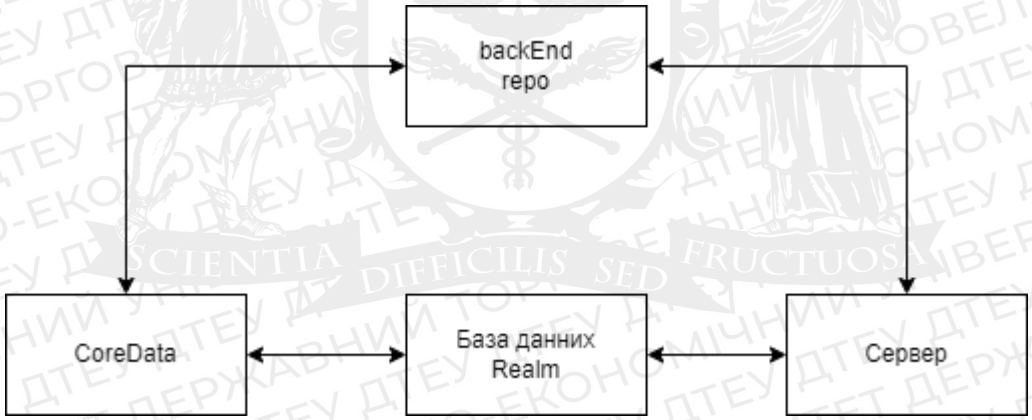


Рисунок 3.3 — Структурна схема ПЗ підсистеми «backEnd»

З рисунку видно, що структурна схема підсистеми «backEnd» складається з наступного:

- Фреймворк `CoreData`, файл що містить візуальне представлення структури даних вашого додатка. Ви можете визначити сутності, їх атрибути та відносини між сутностями

- `База данних Realm`, а тобто репозиторій, що містить файли з конфігураційними данних для бази данніх.
- `Сервер` – для зберігання та обробку даних, автентифікацію користувачів, надсилання повідомлень тощо.

Розглянемо детальніше основні програмні модулі backEnd та структуруємо це наступною таблицею 3.2.

Таблиця 3.2 — Специфікація програмних модулів підсистеми «backEnd»

Модуль	Пояснення
`CoreData`	Це фреймворк, розроблений компанією Apple для роботи з базами даних у вашому iOS або macOS додатку. Основна мета Core Data полягає в спрощенні управління і зберіганням даних у ваших програмах.
`База данних Realm`	Це мобільна база даних для розробки iOS та Android додатків. Realm пропонує простий та продуктивний спосіб зберігання та роботи з даними.
`Сервер`	Для мобільних додатків сервер може використовуватися для низки завдань, включаючи зберігання та обробку

	даних, автентифікацію користувачів, надсилання повідомлень тощо. Це може бути власний сервер, який ви розгортаєте і керуєте, або ви можете використовувати хмарні сервіси.

3.4 Керівництво користувачів мобільного додатку

Користувач — це клієнт, який використовує наш мобільний додаток з метою отримати данні об вітамінах та їх дозуванні. Важливо навести інструктивне керівництво для потенційних клієнтів, які не мають досвіду роботи з даною системою, але мають намір використовувати її.

Розпочнемо з реєстрації. Після успішної автентифікації, яка включає реєстрацію тобто відповідей на питання, система перенаправляє користувача-клієнта на головний екран нашого додатку, яке містить навігаційне меню з переліком доступних функцій додатку. (рисунок 3.4).

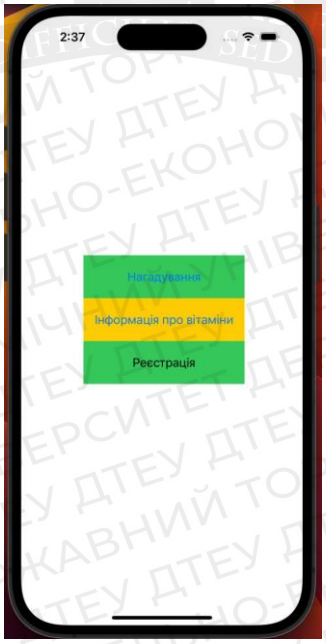


Рисунок 3.4 — Головний екран додатку

Відповідно до процесу отримання інформації про вітаміни, першим кроком є додавання інформації. Це можна зробити за допомогою підпункту меню "Реєстрація", що відкриє екран вводу даних додатку (рисунок 3.5). Дані екрани цього меню складається з послідовних форм, які необхідно заповнити коректними даними. У разі некоректного введення даних, система не дозволить відправити форму та видасть відповідні помилки.

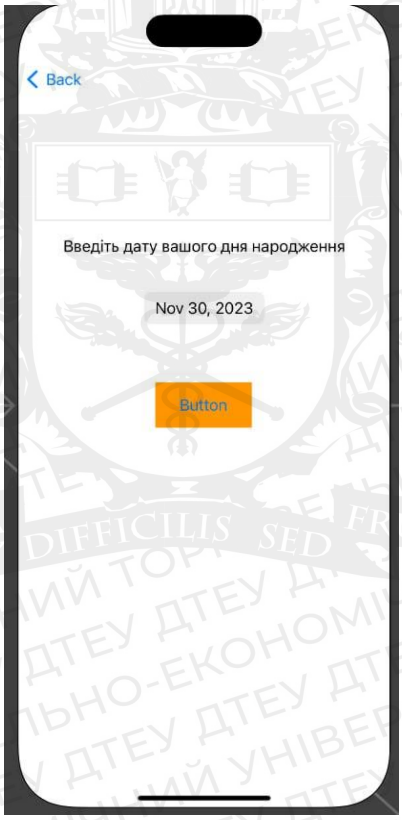


Рисунок 3.5 — Сторінка вводу дати народження

Після успішного додавання даних, додаток перенаправляє користувача на сторінку головного меню. Далі користувач може перейти компоненту меню "Інформація про вітаміни". (рисунок 3.6).

Рисунок 3.6 — Екран списку вітамінів

Користувач обирає вітамін який його цікавить та натискає на нього. Після цього додаток перенаправляє користувача на екран з інформацією про вітамін. В на цьому екрані міститься інормація про рекомендований час приймання вітаміну, дозування для обраного вітаміну яке автоматично розраховується з урахуванням введених данних при реєстрації. А також продукти які допомогають краще засвоюватися цій добавці. (рисунок 3.7).

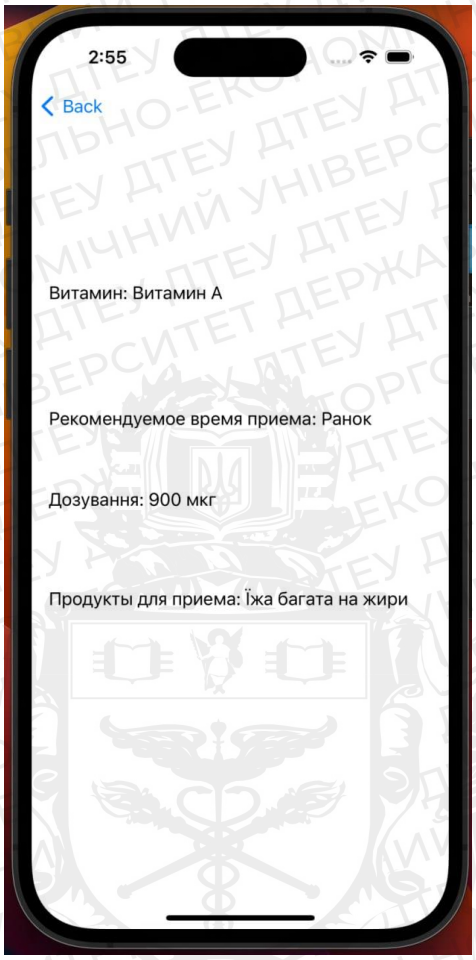


Рисунок 3.7 — Екран інформації про обраний вітамін

Після того як дізналися всю потрібну інформацію. Можна повернутися до головного меню та поставити нагадування у владці "Нагадування". Це допомагає не пропускати час прийому вітаміну, а також на цьому екрані ми можемо обрати певний вітамін, додаток автоматично підтягне рекомендований час прийому. (рисунок 3.8).

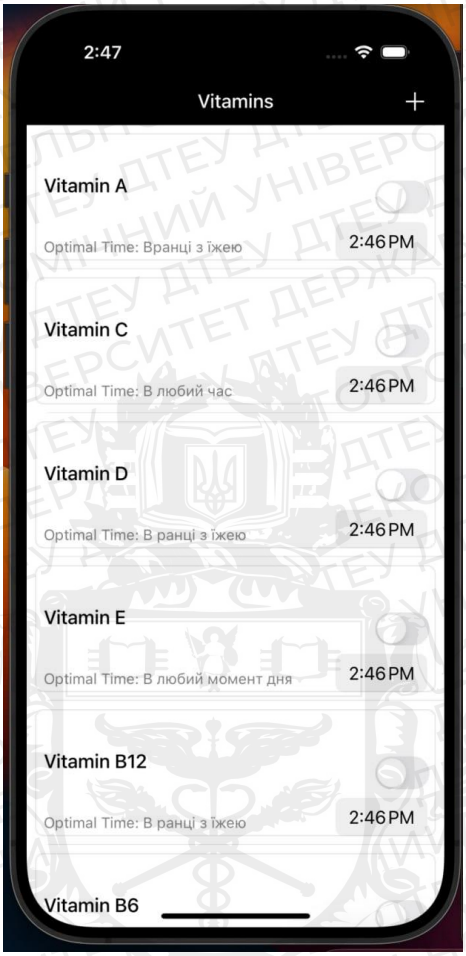


Рисунок 3.8 — Екран нагадування

Якщо ми хочемо додати певний будильник, ми натискаємо кнопку створення та нас перенаправляє на екран створення вітаміну, де ми можемо обрати час, вписати певну потрібну нам інформацію. (рисунок 3.9).

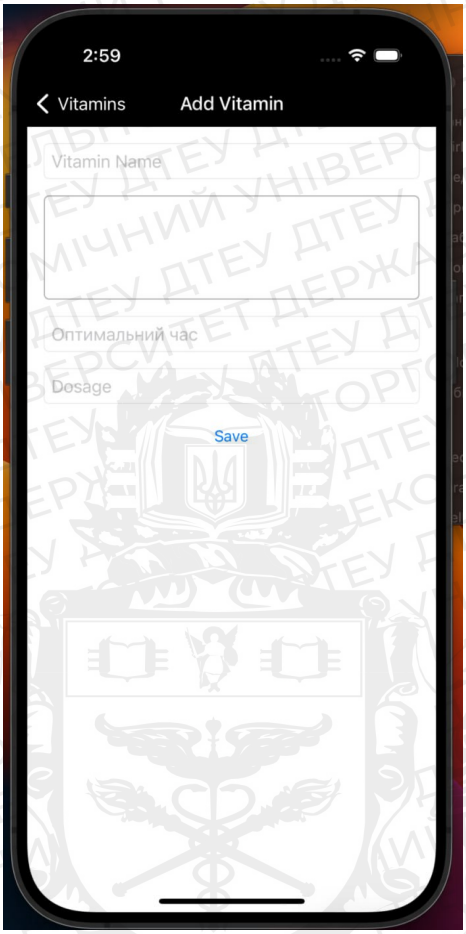


Рисунок 3.9 — Екран сворення нагадування

3.5 Тестування мобільного додатку

У сучасному цифровому світі мобільні додатки стали невіддільною частиною повсякденного життя користувачів. Популярність платформи iOS та швидкий розвиток мобільних технологій породжують величезну кількість нових додатків, які потребують якісного тестування для забезпечення надійності та ефективності.

Пропоную розглянути доступні методи тестування:

- Системне тестування
- UI тестування на iOS
- Регресійне тестування
- Тестування сумісності
- Тестування продуктивності

Розглянемо детальніше доступні методи тестування та структуруємо це наступною таблицею 3.3.

Таблиця 3.3 — Специфікація методів тестування

Метод тестування	Пояснення
Системне тестування	Цей тип тестування проводиться, щоб перевірити, чи працюють різні компоненти системи разом. Під час тестування додаток iOS запускається на реальному пристрої Apple з

	подальшою взаємодією з користувальницьким інтерфейсом для ініціювання певних дій користувача. Типовою дією користувача може бути проведення пальцем по екрану. Фактичний результат порівнюється з очікуваним результатом.
UI тестування на iOS	Тестування функціональності для вхідних даних, апаратних кнопок, програмні кнопки/клавіатура, тестування на різних розширеннях екранів.
Регресійне тестування	Щоб перевірити, чи працює програма однаково після змін, необхідно провести регресійне тестування. І оскільки це повторювана діяльність, автоматизація стане в нагоді для такого типу тестового запуску.
Тестування сумісності	Як вже було сказано, існує безліч пристроїв, випущених Apple. Це означає що програма повинна працювати на всіх цих пристроях, якщо вона є сумісною з ними. Мануальне тестування на усіх пристроях неможливе, тому в нагоді

	стає автоматизоване тестування.
Тестування продуктивності	Тестування поведінки програми під час використання, поведінки програми під час тривалої роботи, різне навантаження.

Пропоную протестувати наш додаток за допомогою тестування сумісності, бо це вважається найчастішою проблемую серед iOS розробників.

На цьому скріншоті ми можемо побачити результати тестування, на якому видно що на всіх молодших версіях Iphone від 15, компоненти нікуди не з’їзджають, та знаходятся у нормальному робочому стані. (рисунок 3.10).

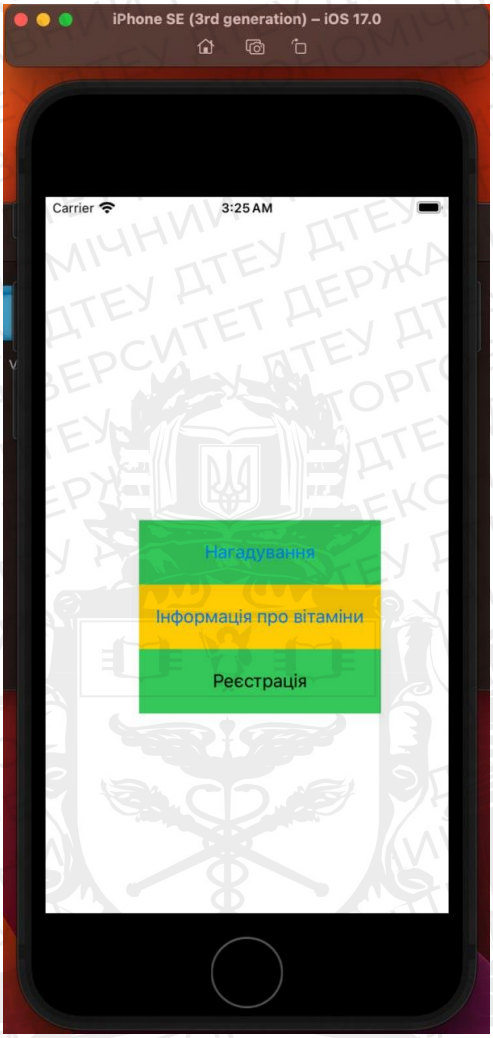
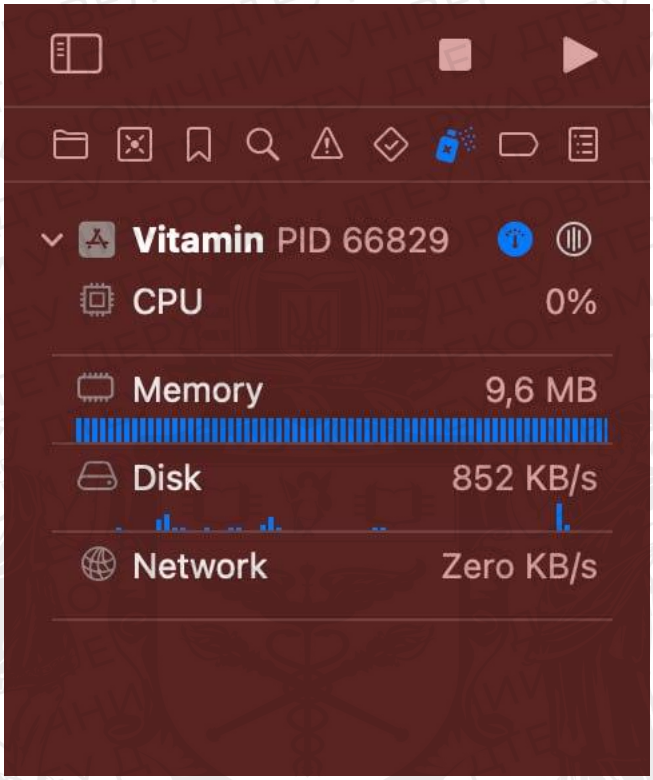


Рисунок 3.10 — Результати тестування сумісності

Проведемо тестування продуктивності нашого додатку в робочому стані, за допомогою вбудованого інструменту тестуваня який показує навантаженя на процесор, кількість оперативної та фізичної пам’яті яка використовується при роботі данного додатку. На даному рисунку ми бачимо результат тестуваня, на якому ми бачиному практично нульове використаня процесора, це означає що додаток написаний правильно. Стабільне використовуваня оперативної пам’яті вказує на то що додаток вигружає с пам’яті непотрібні елементи. Стабільне використовуваня фізичної пам’яті вказую на то що додаток під час

роботи правильно функціонує та не зберігає дані по кілька разів.(рисунок 3.12).



Рисунки 3.12 — Результати тестування продуктивності

ВИСНОВКИ

У випускній кваліфікаційній роботі представлено результати теоретичних і прикладних досліджень, що полягають у розробці мобільного додатку, що допомагає людям відстежувати їх добову норму вітамінів,харчування та загальне самопочуття. Результати прикладних досліджень стали основою для створення автоматизованої системи оцінювання показників. В результаті проведених досліджень були отримані такі **результати**:

1. Розроблено додаток, який успішно виконує поставлені завдання, а саме відстеження добової норми вітамінів та контроль харчування. Додаток дозволяє користувачам зручно записувати своє харчування та отримувати рекомендації щодо вживання вітамінів.

2. Інтеграція з CORE DATA дозволила зберігати дані користувачів локально на їхніх пристроях, що робить додаток більш надійним та швидкодійним.

3. В роботі була проведена аналіз сучасних тенденцій у галузі харчування та здорового способу життя, що дозволило розробити програмний алгоритм для данного додатку.

4. Описаний процес розробки додатка з використанням мов програмування для iOS-платформи, проведено тестування на різних пристроях та оцінено функціональність.

5. Описана архітектура програмного забезпечення додатка, розроблено інтерфейс користувача з врахуванням зручності використання.

В цілому, розробка додатку для відстеження вітамінів, контролю харчування та поліпшення самопочуття користувачів є актуальною та важливою, оскільки вона сприяє збереженню здоров'я та покращенню якості

життя. Результати дослідження та розробки цієї дипломної роботи можуть бути використані для подальшого вдосконалення додатків, спрямованих на підтримку здорового способу життя.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Василий Усов.Swift. Основы разработки приложений под iOS, iPadOS и macOS.

2. Эхаб Йосри Амер, Алексис Гэллахер, Мэтт Гэлловей, Эли Гэним. Swift Apprentice: Fundamentals (First Edition): Beginning Programming in Swift.

3. Чейрд Ин't Вейн. Swift подробно.

4. Джон Мэннинг, Пэрис Баттфилд-Эддисон. Head First. Изучаем Swift.

5. Майки Уорд. Swift Programming: The Big Nerd Ranch Guide 3rd Edition.

6. Дж. Д. Гаучат. SwiftUI for Masterminds.

7. Кит Мун. Swift Cookbook: Over 60 proven recipes for developing better iOS applications with Swift 5.3.

8. Thomas E. Levy, 2002. Curing the Incurable: Vitamin C, Infectious Diseases, and Toxins.

9. Pamela Wartian Smith, 2008. What You Must Know About Vitamins, Minerals, Herbs and So Much More—SECOND EDITION: Choosing the Nutrients That Are Right for You.

10. Phyllis A. Balch, 2023. Prescription for Nutritional Healing: The A-to-Z Guide to Supplements, 6th Edition: Everything You Need to Know About Selecting and Using Vitamins, Minerals, Herbs, and More.

11. Walt Larimore M.D., 2021. The Natural Medicine Handbook: The Truth about the Most Effective Herbs, Vitamins, and Supplements for Common Conditions.

12. Harold M. Silverman, 1991. The Vitamin Book: The Complete Guide to Vitamins, Minerals, and the Most Effective Herbal Remedies and Dietary Supplements.

13. Earl Mindell, 1979. Earl Mindell's Vitamin Bible.

14. James Fergusson, 2007. The Vitamin Murders: Who Killed Healthy Eating in Britain?

15. Jonathon Manning, Paris Buttfield-Addison. «Mobile Game Development with Unity: Build Once, Deploy Anywhere».

16. Iyanu Adelekan. «Kotlin Programming by Example: Build real-world Android and web applications the Kotlin way».

17. Вандад Нахавандипур. «iOS. Приемы программирования».

18. Дэвид Марк, Джек Наттинг, Джефф Ламарш, Фредрик Олссон, Ким Топли. «Swift. Разработка приложений в среде Xcode для iPhone и iPad с использованием iOS SDK».

19. Ханг Во. «Оптимизация производительности приложений для iOS».

ДОДАТОК

Програмний код реалізації мобільного додатку (на диску)