

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіоелектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ КОНТРОЛЮ
СПОЖИВАННЯ МІКРОЕЛЕМЕНТІВ І ВІТАМІНІВ. SOFTWARE
DEVELOPMENT FOR MONITORING THE CONSUMPTION OF MINERALS
AND VITAMINS

Виконав: студент(ка) 4 курсу, групи КНТ-217
Спеціальності 122 Комп'ютерні науки
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Комп'ютерні науки

Шатохін Є.Р.

(прізвище та ініціали)

Керівник Федорончак Т.В.

(прізвище та ініціали)

Рецензент Казурова А.Є.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ІРЕ, ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 122 Комп'ютерні науки
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
“ ” 20 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТА(КИ)

Шатохін Євгеній Русланович

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Розробка програмного забезпечення для контролю споживання мікроелементів і вітамінів. Software Development for Monitoring the Consumption of Minerals and Vitamins.

керівник проекту (роботи) Федорончак Тетяна Василівна, к.т.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “30” березня 2021 року № 103

2. Строк подання студентом проекту (роботи) 14.05.2021

3. Вихідні дані до проекту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Розробка архітектури програми. 3. Розробка програмного забезпечення. 4. Експлуатація, тестування та експериментальне дослідження програми. 5. Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4	Федорончак Т.В., доцент		
Нормоконтроль	Дейнега Л.Ю., ст. викладач		

7. Дата видачі завдання 21 лютого 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	2–3 тижні	Розділ 1
3	Розробка архітектури програми.	4 тиждень	Розділ 2
4	Розробка програмного забезпечення.	5–6 тижні	Розділ 3
5	Тестування та експериментальне дослідження програми.	7 тиждень	Розділ 4
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	8 тиждень	Додатки
7	Захист роботи.	9 тиждень	

Студент (ка) _____ Шатохін Є.Р.
(підпис) (прізвище та ініціали)

Керівник проекту (роботи) _____ Федорончак Т.В.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до випускної кваліфікаційної роботи бакалавра:
109 с., 11 табл., 32 рис., 3 дод., 15 джерел.

КОМП'ЮТЕРНА ПРОГРАМА, КОНТРОЛЬ ВІТАМІНІВ, АНАЛІЗ ДЕФІЦИТУ МІКРОЕЛЕМЕНТІВ.

Об'єкт дослідження – програмні засоби для контролю споживання мікроелементів та вітамінів.

Предмет дослідження – комп'ютерні програми для контролю мікронутрієнтів, завдяки споживанню їжі.

Мета роботи – створення програмної системи для контролю та аналізу дефіциту мікроелементів, або вітамінів для домашнього користування.

Матеріали, методи, та технічні засоби: структурне, та об'єктно-орієнтоване програмування, технологія, та пристрій клавіатура, та миш, мови програмування C# та Python, мова розмітки Xaml, персональний комп'ютер з процесором Intel Celeron CPU G530 під управлінням операційної системи Microsoft Windows 10.

Результати. Створено програмний Windows-застосунок, керований пристроєм клавіатурою, та мишею, який у формі двомірного вікна взаємодіє з користувачем, та стимулює його до аналізу споживаних мікронутрієнтів.

Висновки. Розроблено програмну систему для контролю, та аналізу дефіциту мікронутрієнтів, яка аналізує, та прискорює процес доповнення вітамінів, та мінералів в організмі.

Галузь використання – в медичних закладах лікарями-дерматологам, або в домашніх умовах.

ABSTRACT

Explanatory note to the final qualifying work of the bachelor: 109 pages, 11 tables, 32 figures, 3 appendixes, 15 sources.

COMPUTER PROGRAM, VITAMIN CONTROL, ANALYSIS OF MICROELEMENT DEFICIENCY.

The object of research - software to control the consumption of trace elements and vitamins.

The subject of the research is computer programs for control of microelements and vitamins, thanks to the consumed food.

The purpose of the work is to create a software system for control and analysis of micronutrient or vitamin deficiencies for home use.

Materials, methods and technical means: structural and object-oriented programming, keyboard and mouse technology and device, C # and Python programming languages, Xaml markup language, personal computer with Intel Celeron CPU G530 processor running Microsoft Windows operating system 10.

Results. A Windows-based software application controlled by the device with a keyboard and mouse has been created, which interacts with the user in the form of a two-dimensional window and stimulates him to analyze the consumed elements.

Conclusions. A software system for control and analysis of micronutrient or vitamin deficiency has been developed, which analyzes and accelerates the process of supplementing vitamins and minerals in the body.

Field of use - in medical institutions by dermatologists or at home.

ЗМІСТ

	С.
Перелік скорочень та умовних ознак	8
Вступ	9
1 Аналіз проблеми та постановка завдань роботи	10
1.1 Аналіз предметної області	10
1.2 Аналіз аналогів	11
1.3 Постановка задач розробки	15
1.4 Висновок	15
2 Розробка архітектури програми	17
2.1 Аналіз інформаційного забезпечення	17
2.2 Аналіз технологій	17
2.3 Розробка структури віконної програми	19
2.4 Діаграма потоків даних	21
2.5 Загальний принцип та алгоритм роботи віконної програми	23
2.6 Розробка структури баз даних	24
2.7 Аналіз вимог до розробки дизайну програми	25
2.8 Висновок	26
3 Розробка програмного забезпечення	27
3.1 Аналіз і обґрунтування вибору мови програмування	27
3.2 Аналіз і обґрунтування вибору середовища розробки	29
3.3 Розробка дизайну програми	32
3.4 Розробка модулів програми	36
3.5 Висновок	49
4 Експлуатація, тестування та експериментальне дослідження	50
4.1 Тестування програми	50
4.2 Опис програмного забезпечення	53
4.3 Тестування в критичних середовищах	53
4.4 Висновок	54
Висновки	55

Перелік джерел посилання.....	57
Додаток А Технічне завдання.....	59
Додаток Б Текст програми	63
Додаток В Слайди презентації.....	103

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

WPF	–Windows Presentation Foundation;
ПЗ	–програмне забезпечення;
XAML	–eXtensible Application Markup Language,
MVVM	–Model-View-ViewModel;
БД	–База даних;
VM	–Virtual machine;

ВСТУП

На сьогодні більшість мінералів та вітамінів необхідні для підтримання хорошого здоров'я. Основний спосіб отримання мікронутрієнтів це через їжу. Наприклад якщо в організмі буде мало мікронутрієнтів, а якщо в організмі дефіцит вітамінів то це можливий розвиток різних захворювань.

Більшість лікарів при огляді запитують у пацієнтів скільки вони п'ють води, та що вони їдять. Це важливе питання оскільки, лікарі аналізують інформацію і завдяки отриманій інформації вони можуть визначити можливу причину певної хвороби. Головна проблема в тому що, люди не контролюють та не аналізують рівень вживаних вітамінів та мікроелементів.

Актуальність даної теми полягає в розробці та створенні програмного забезпечення, що являє собою рішення для аналізу вживаних мікронутрієнтів.

Створення програмного забезпечення дозволяє заздалегідь визначати дефіцит мікронутрієнтів тим самим зменшити виникнення можливих захворювань.

Об'єктом дослідження є програмні засоби для контролю споживання мікроелементів та вітамінів.

Предметом дослідження комп'ютерні програми для контролю мікронутрієнтів, завдяки споживаній їжі.

Метою бакалаврської роботи є покращення візуалізації інформації шляхом розробки віконної програми для якісного аналізу споживаних мікронутрієнтів отриманої після роботи програми.

Для досягнення поставленої мети у роботі вирішувалися такі завдання: огляд стану питання, рішення цього завдання це аналіз аналогів, вивчення аспектів які допоможуть формулювати якісні методи аналізу вживаних вітамінів, розроблення архітектури програми, дизайну, розроблення програмного забезпечення, тестування результатів та перевірка даних.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ РОБОТИ

1.1 Аналіз предметної області

На сьогоднішній день в мережі є дуже багато інформації про те, як впливають вітаміни та мікроелементи на організм, але дуже мало реалізації, які групують інформацію в одному місці. Якщо брати інформацію з різних ресурсів, то це займе безліч часу, це і є головною причиною того, що люди не контролюють та не аналізують отримання всіх важливих мікронутрієнтів. І це може викликати безліч різних захворювань.

Структурована програмна інформація дозволяє користувачам отримати повну картину стану всіх рівнів мікронутрієнтів в організмі. Користувач отримує графік рівнів і короткий список можливих захворювань. Користувачеві необхідно ввести тільки вхідні дані. Для користувача цей метод аналізу більш ефективний, ніж звичайні засоби пошуку інформації. Використовуючи інтерфейс, або навігацію, користувач може вибрати необхідну інформацію.

Процес створення і розробки віконної програми включає в себе:

- затвердження технічного завдання;
- визначення розробленої структурної схеми ПЗ та розташування розділів, контенту і навігації;
- проектування графічних елементів, стилів та елементів навігації в ПЗ;
- розробка програмного коду, бази даних та інших програмних елементів;
- тестування ПЗ.

На прикладі вже існуючих програм розробимо свою, при цьому необхідно врахувати недоліки та переваги вже наявних систем.

Для початку розглянемо такі поняття як: вітаміни, мікроелементи в організмі, гіповітаміноз, гіпервітаміноз.

Вітаміни – це група незамінних для організму людини і тварин органічних сполук, що володіють високою біологічною активністю, доступних

в невеликих кількостях в їжі. Більшість з них потрапляє в організм з їжею, і тільки деякі синтезуються мікроорганізмами в кишечнику.

Мікроелементи – це мінеральні елементи, що є в тканинах організму і харчових продуктах в значних кількостях (десятки і сотні міліграмів на 100 г продукту) [1].

Гіповітаміноз – це стан, який розвивається внаслідок зменшення вмісту певного вітаміну в організмі [2].

Гіпервітаміноз – вид патології, що виникає при надмірному надходженні вітамінів до організму [3].

Основні цілі програм контролю:

- поширення основної інформації;
- аналіз інформації;
- завчасне виявлення можливої хвороби та вказування на розв’язування проблеми.

Отже, основна мета будь-якої програми для аналізу мікронутрієнтів, полягає в обслуговуванні користувачів способом, який принесе йому вигоду по часу та зручності. Кінцевий результат використання програми полягає в підштовхування людей на повторне використання програми.

1.2 Аналіз аналогів

На сьогодні не існує реалізацій у вигляді програми на робочий стіл, але ця ідея закладена в web-технологіях і в мережі існує багато сайтів тому для порівняння будуть використані web-сайти для контролю мікронутрієнтів в організмі.

Проаналізуємо три аналоги веб-сайтів у галузі контролю мікронутрієнтів: ”Щоденник харчування”, “Cronometer” та “Fit Audit”.

Розглянемо сайт для аналізу мікроелементів "Щоденник харчування". Інтерфейс сайту дуже нагромаджений, що збільшує кількість помилок при

використанні сайту. Дизайн виконаний у мінімалістичному стилі, тут немає скупчення кольорів та елементів, які відволікають увагу [4].

Сайт є досить функціональним. На головній сторінці присутній фільтри для виведення загальної інформації. Присутня можливість знаходження та додавання продуктів для розрахунку мікронутрієнтів. Форма додавання є досить інтуїтивна та зрозуміла. Присутня можливість динамічного перегляду змін. Головна сторінка сайту "Щоденник харчування" (рис. 1.1).

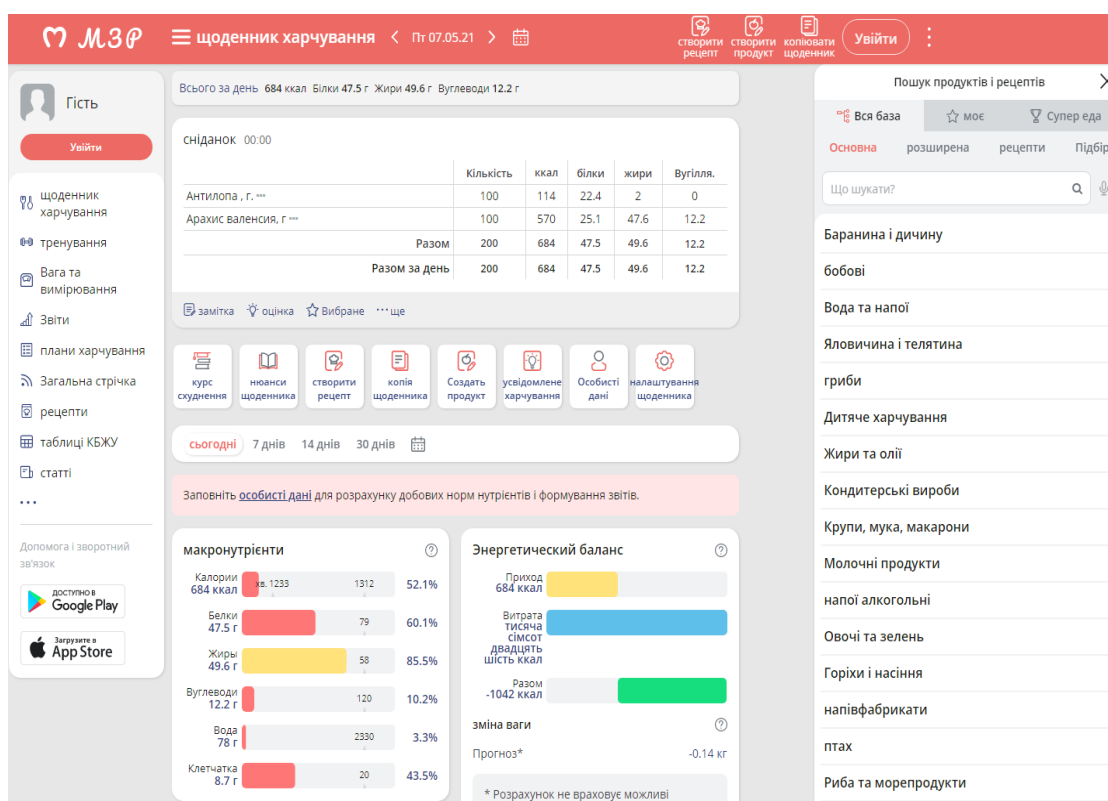


Рисунок 1.1 – Аналітична сторінка сайту "Щоденник харчування"

Наступним аналогом для порівняння візьмемо сайт "Cronometer". Інтерфейс сайту дуже складний і не зрозумілий. Хоча сайт створений для зарубіжної аудиторії. Дизайн виконаний в мінімалістичному стилі [5].

Погано реалізована можливість знаходження та додавання продуктів для розрахунку мікронутрієнтів. На сайті присутні фільтри з великими інтервалами від семи днів до восьми місяців, завдяки фільтрам виводиться

загальна інформація. Також присутня можливість динамічного перегляду змін. Головна сторінка сайту “Cronometer” (рис. 1.2).

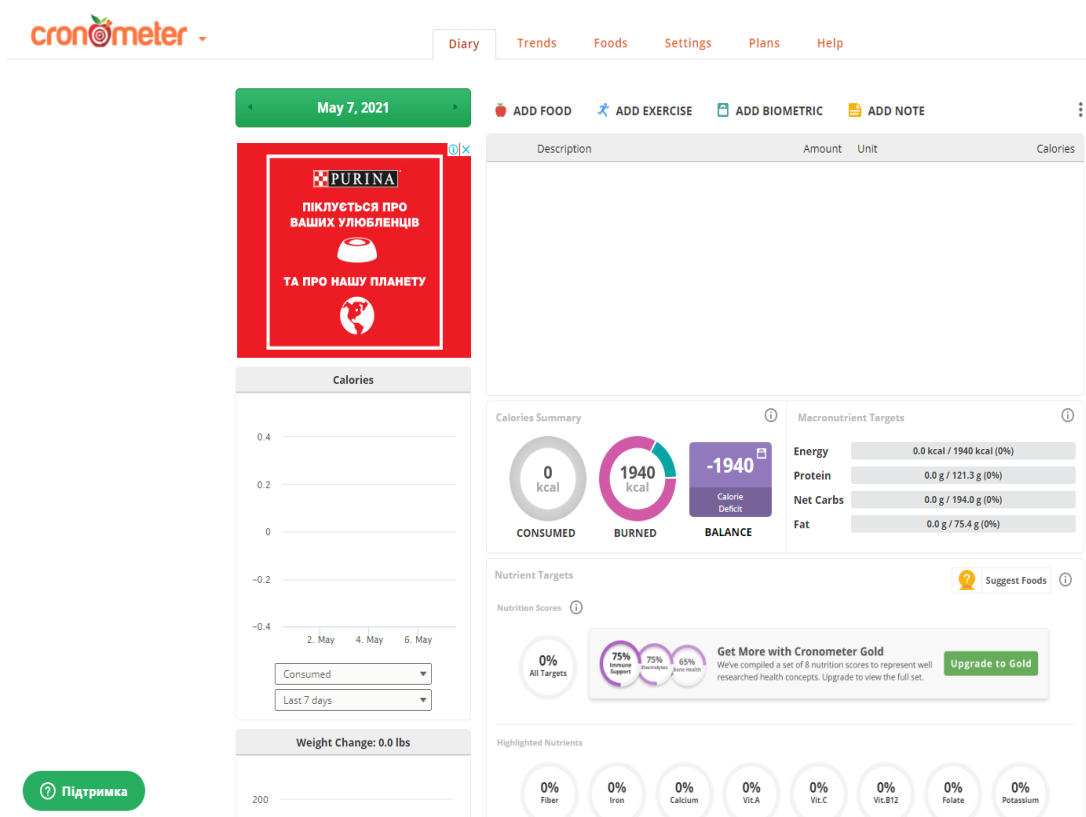


Рисунок 1.2 – Аналітична сторінка сайту “Cronometer”

Розглянемо сайт “Fit Audit”. Дизайн сайту простий і зрозумілий [6]. Але на сайті не має можливості підрахунку і контролю, прийдеться самотійно шукати інформацію на сайті, але це займе велику кількість часу. Також відсутні будь-які фільтри. З позитивного – присутня велика кількість продуктів для створення раціону. При перегляді продукту змінюється тематика шапки сторінки на картинку самого продукту. Головна сторінка сайту “Fit Audit” (рис. 1.3).

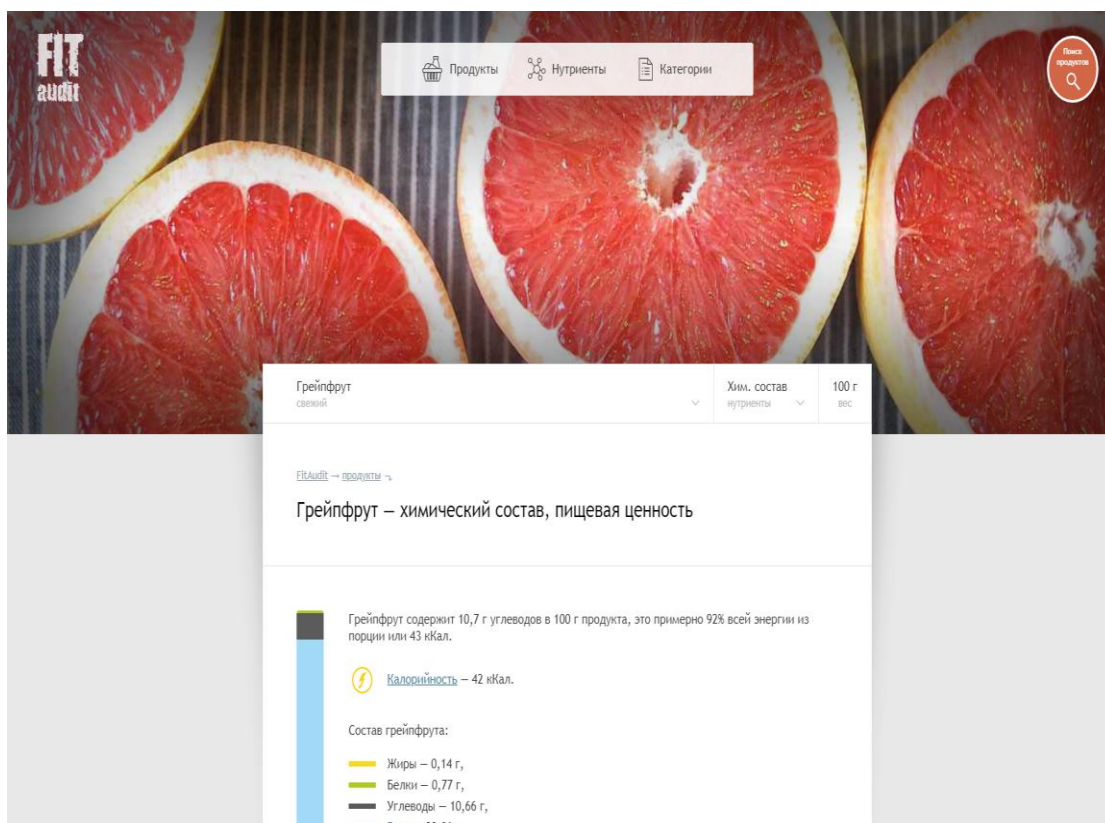


Рисунок 1.3 – Аналітична сторінка сайту “Fit Audit”

Усі вищеперераховані веб-сайти дозволяють здійснювати аналіз мікронутрієнтів. Порівняльний аналіз аналогів (табл. 1.1).

Таблиця 1.1 – Порівняльна характеристика аналогів

Характеристика	”Щоденник харчування”	“Cronometer”	“Fit Audit”
Зрозумілий дизайн	+/-	+/-	+
Зрозумілий функціонал	+	—	+
Динамічний перегляд змін	+	+	+
Присутність алгоритмів	+	+	—

Продовження таблиці 1.1

Характеристика	”Щоденник харчування”	“Cronometer”	“Fit Audit”
аналізу			
Фільтри	+	+	+/-
Особистий кабінет	+	+	—
Пошук	+	+	+

1.3 Постановка задач розробки

Оскільки завданням роботи це розробка інтерфейсної програми, призначеної для аналізу та виведення графіків та стислого переліку можливих захворювань, можна виділити такі задачі для виконання:

- проаналізувати інформаційне забезпечення;
- розробити організаційну та функціональну структуру;
- проаналізувати логічну структуру інтерфейсної програми;
- розробити структуру бази даних;
- розробити інтерфейс;
- провести паралель між можливостями мов програмування;
- проаналізувати потенціал інтегрованих середовищ розробки;
- розробити систему аналізу нутрієнтів;
- провести тестування та виправлення помилок.

1.4 Висновок

В ході виконання першого розділу було проаналізовано предметну область. Виявлено, що в мережі немає аналогів у вигляді інтерфейсних програм, але багато реалізацій web-додатків. Завдяки аналізу аналогів було визначено переваги та недоліки аналізу нутрієнтів. На базі отриманих даних було сформульовано список переваг, який буде реалізований при розробці

інтерфейсної програми, а саме зручний інтерфейс, динамічний перегляд змін, фільтри, особистий кабінет та функція пошуку.

Було визначено основні задачі розробки.

2 РОЗРОБКА АРХІТЕКТУРУ ПРОГРАМИ

2.1 Аналіз інформаційного забезпечення

Розроблювана інтерфейсна програма буде містити та оперувати інформацією. В базі присутні більш ніж п'яти тисяч позицій продуктів харчування. Крім того, є можливість доповнювати базу даними. Вони мають свій набір характеристик (код продукту, назва та найменування нутрієнтів). Також програма буде оперувати базою зареєстрованих користувачів та можливість доповнення.

Програма для контролю нутрієнтів надаватиме користувачеві можливості оперування даними. Реалізована можливість реєстрації для зберігання подальшої інформації. Також присутня можливість добавлення продуктів до панелі, крім того, панель нутрієнтів буде відновлюватись, присутня можливість видалення із динамічної панелі. Динамічну панель можна використовувати, як для створення, так і для аналізу раціону. Також наявна можливість аналізу з подальшим виведенням в графік, для графіків існують інтервальні фільтри. Програма надає можливість формулювати стислий перелік можливих захворювань завдяки аналізу даних.

2.2 Аналіз технологій

Для вирішення проблеми розробки програми аналізу поживних речовин необхідно визначити мовні технології, які будуть використані при розробці. Давайте розглянемо основні базові технології, що використовуються для побудови графічних програм.

WPF – аналог WinForms, система для побудови клієнтських додатків Windows з візуально привабливими можливостями взаємодії з користувачем, графічна (презентаційна) підсистема у складі .NET Framework (починаючи з версії 3.0), яка використовує мову XAML [7].

XAML – це декларативна мова розмітки. Застосовано до моделі програмування .NET Core, XAML спрощує створення інтерфейсу для програми .NET Core [8].

C# – це сучасна, об'єктно-орієнтована та безпечна мова для програмування. C# дозволяє розробникам створювати багато типів безпечних та надійних додатків, що працюють в екосистемі .NET [9].

Завдяки WPF, XAML та C # ми зможемо створити сучасний, простий, швидкий інтерфейс з багатою функціональністю. Крім того, завдяки C # ми зможемо створити високоякісну функціональність, яка буде взаємодіяти з XAML.

Python – це поки відносно молода мова програмування, але вже дуже популярна в області технологій створення алгоритмів та штучного інтелекту. На цей час, вдосконалення мови не закінчене. Мова постійно допрацьовується та вдосконалюється, останню версію, яку розробники випустили це 3.9. Технічний комітет працює над істотними розширеннями.

Завдяки Python ми зможемо швидко написати програмний модуль для парсинга та оброблювань інформації.

MVVM - шаблон проектування архітектури програми.

За допомогою цієї проектної архітектури ми зможемо повторно використовувати інтерфейс, що дозволить нам зменшити код та швидкість виконання програми.

Програми з інтерфейсом мають можливість використовувати велику кількість різноманітних баз даних, таких як MySQL, PostgreSQL, MS SQL Server та SQLite. Найпоширенішим рішенням при виборі бази даних для зберігання великої кількості інформації є бази даних MySQL, або легкий портативний SQLite. Обидві ці бази даних є безкоштовними продуктами та з відкритим кодом. Водночас вони відповідають серйозним вимогам щодо своїх можливостей та характеристик.

SQL (структурована мова запитів) – створена для роботи з базами даних. Вона дозволяє користувачам взаємодіяти з базами даних (переглядати, шукати, додавати, керувати даними).

SQLite – це портативна база даних SQL. Має хорошу швидкість, розмір і гнучкість, якщо використовувати для зберігання тексту.

2.3 Розробка структури віконної програми

Інтерфейсна програма реалізована за допомогою архітектурного шаблону MVVM. У шаблонах проектування MVVM, зміни в інтерфейсі не впливають на саму модель, а попередньо йдуть через контролер (рис. 2.1).

Завдяки цьому шаблону швидкість роботи програми збільшується, а також зменшується нагромадження коду в розробці, а саме це впливає на швидкість розробки ПЗ. Крім того, в шаблоні присутні такі характеристики: можливість розробки гнучкого дизайну для ПЗ, надавання можливості повторного використання окремих компонентів програми. Крім того, використання шаблону у великих проектах призводить до певної впорядкованості в коді.

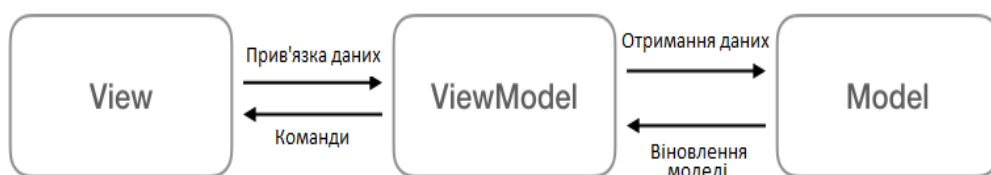


Рисунок 2.1 – Взаємодія компонентів шаблону

Для реалізації функціоналу програми розроблювалися такі модулі:

- модуль база даних;
- модуль реєстрації;

- модуль аутентифікація;
- модуль для створення раціону та додавання до бази;
- модуль для створення графіків;
- модуль для формування списку можливих захворювань.

Модуль “База даних ” включає: базу даних користувачів котра зберігає особисту інформацію користувачів, які користуються програмою, базу даних продукту, яка використовується для утримання інформації (назви та нутрієнтний склад), базу даних записів, яка використовується для збереження записів користувача, а саме вітамінно-мінеральну статистику.

Модуль “Реєстрації” призначений для розмежування інформації, а саме дати можливість користуватися одною програмою безліч користувачам.

Модуль “Аутентифікація” призначений для входу, а саме дає можливість користуватися особистим записом.

Модуль “Створення раціону та додавання до бази” призначений для введення інформації для подальшого аналізу, крім того, модуль дає можливість записувати інформацію, яка буде використовуватись для створення графіку.

Модуль “Створення графіків” призначений для створення графіку на базі введеної інформації, а саме щоденних записів. В даному модулі присутні інтервальні фільтри, які покращують аналіз.

Модуль ”Формування списку можливих захворювань” призначений для створення стислого списку можливих захворювань. Присутні інтервальні фільтри, які покращують аналіз, крім того, модуль дозволяє імпортувати та експортувати стислий перелік в іншу програму.

Структура модулів (рис. 2.2).

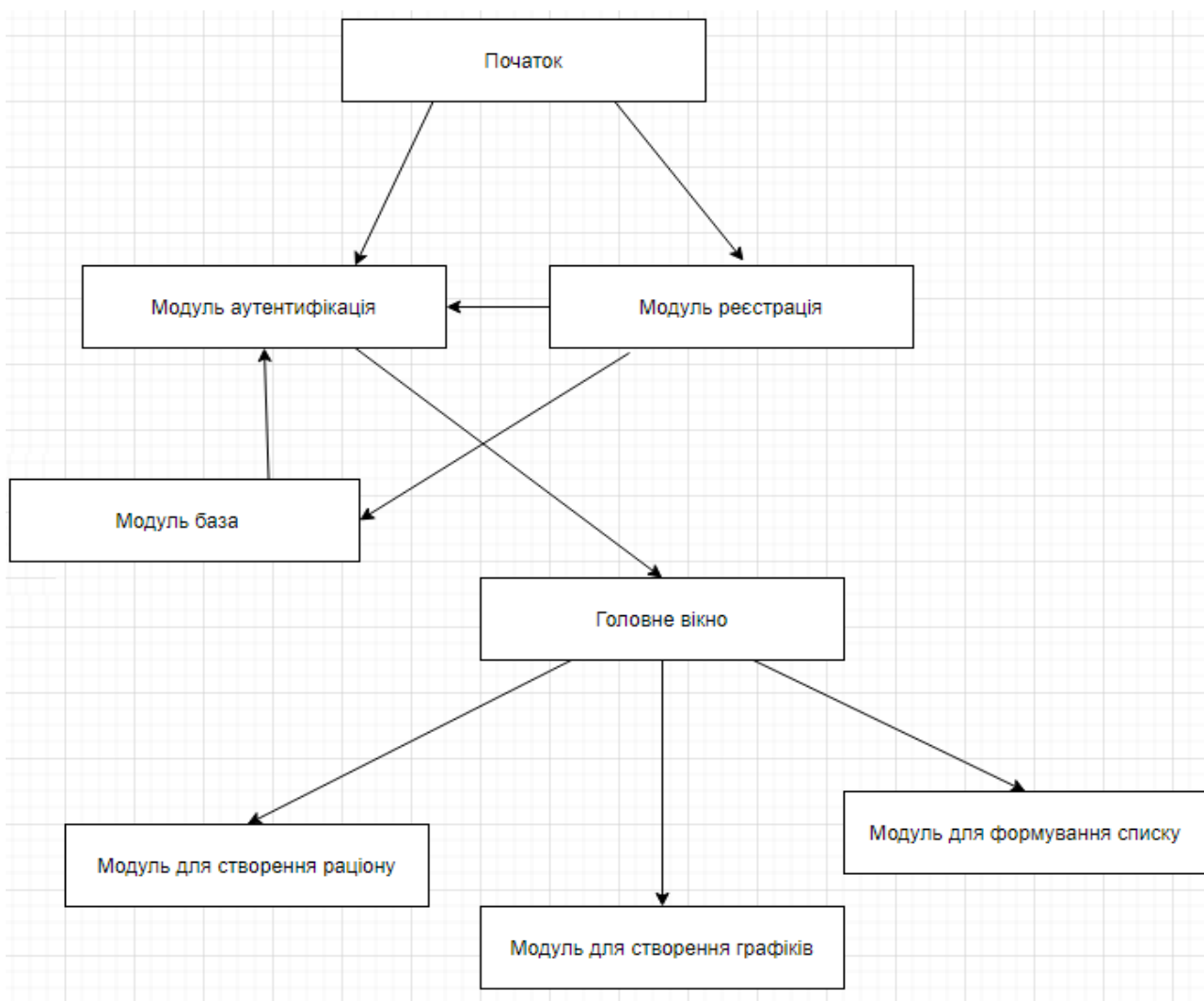


Рисунок 2.2 – Структура модулів інтерфейсної програми

2.4 Діаграма потоків даних

Діаграми потоків даних це ієрархія процесів котрі пов'язані з потоками даних. Мета подання цієї інформації – це ознайомити, як кожен процес трансформує свої вхідні дані у вихідні, а також виявити взаємовідносини серед цих процесів.

В діаграмах потоків міститься інформацію о процесах, які конвертують інформацію в потоки з інформацією, які транспортуються в активні об'єкти, які виробляють і споживають дані, сховища даних, які утримують дані.

Сховище даних – це об'єкт у складі діаграм потоків даних. Сховище даних допускає доступ до збережених даних. Агрегатні сховища, наприклад,

списки та таблиці, забезпечують право доступу до даних в порядку їх надходження до бази, або по унікальних ключів [10].

Сховище даних (рис. 2.3, 2.4 та 2.5) , потоки даних (рис. 2.6).

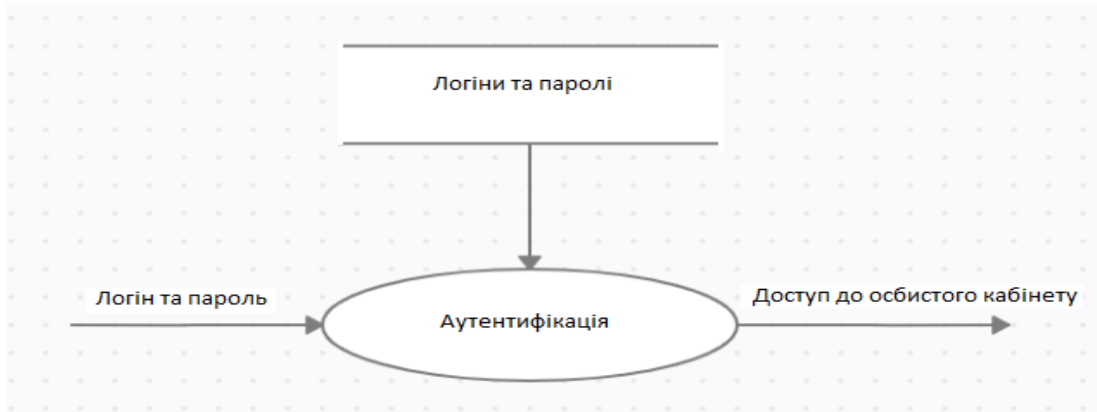


Рисунок 2.3 – Сховище даних аутентифікації



Рисунок 2.4 – Сховище даних продуктів



Рисунок 2.5 – Сховище даних денних списків нутрієнтів

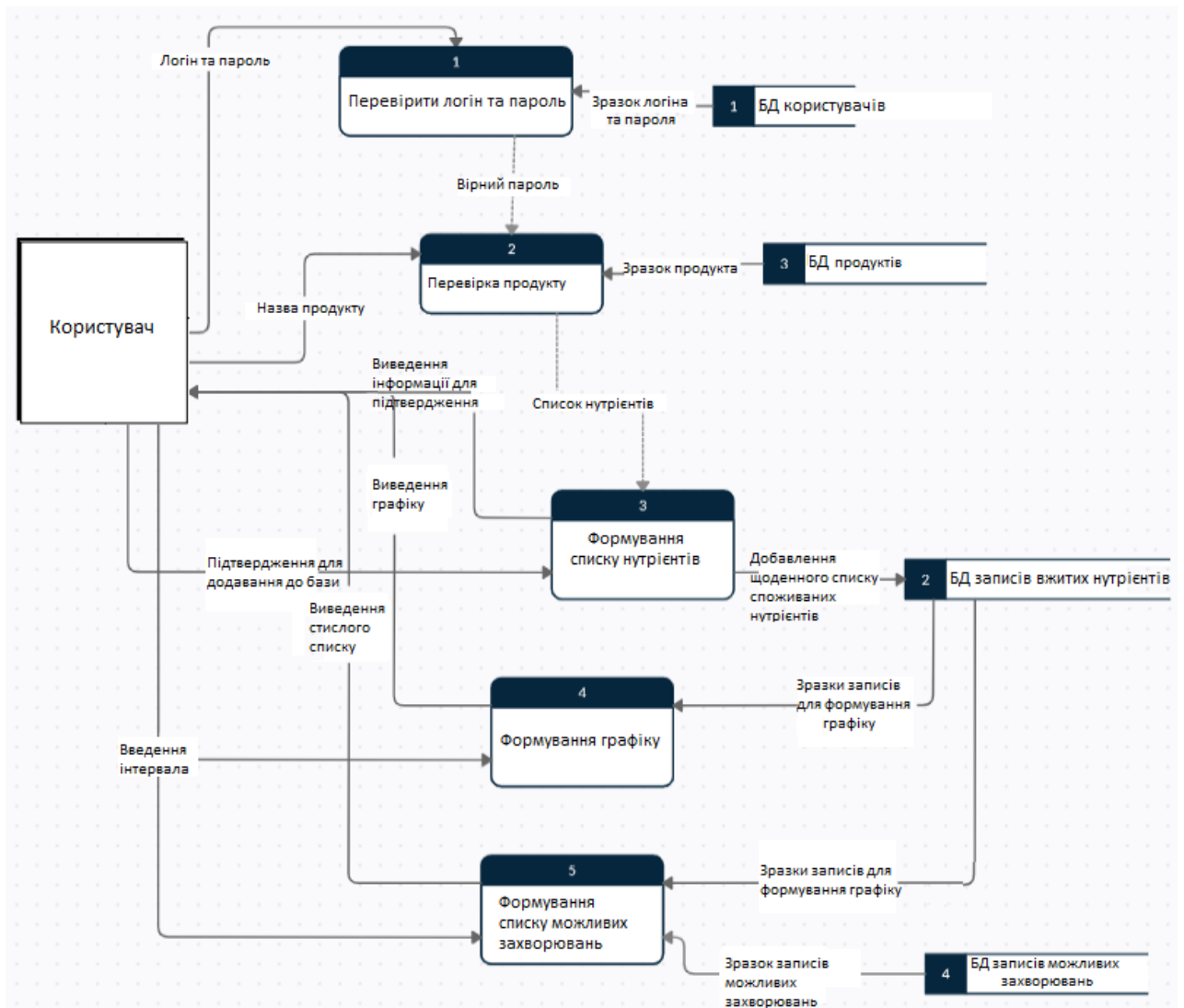


Рисунок 2.6 – Діаграма потоків даних

На підставі проведеного аналізу, була побудована діаграма потоків даних інтерфейсної програми для аналізу вживаних нутрієнтів, з використанням методології Data Flow Diagrams.

2.5 Загальний принцип та алгоритм роботи віконної програми

Програма виводить сторінки аутентифікації та реєстрації через файл MainWindow.cs, а після проходження аутентифікації всі інші сторінки оброблює Main.cs і контент сторінки визначається зміною PrimaryView та CurrenView.

2.6 Розробка структури баз даних

Оскільки в ПЗ використовується не значна інформація, то це дає можливість використовувати технологію SQLite. SQLite дуже компактна та вбудована в програму це дає більшу швидкість між взаємодією програми та бази даних.

Під час проектування було зроблено чотири бази даних це InfoComponents.db, ProductBD.db, Users.db та neturalistBD.db. Атрибутна структура баз даних (рис. 2.7, 2.8 та 2.9).

```
1 CREATE TABLE "components" (  
2     "NAME" TEXT NOT NULL UNIQUE,  
3     "HYPERVITAMINOSIS" TEXT,  
4     "HYPOVITAMINOSIS" TEXT,  
5     "FULL" TEXT,  
6     PRIMARY KEY("NAME")  
7 );
```

Рисунок 2.7 – Атрибутна структура InfoComponents.db

```
1 CREATE TABLE "Users" (  
2     "id" INTEGER NOT NULL UNIQUE,  
3     "login" TEXT NOT NULL UNIQUE,  
4     "email" TEXT NOT NULL,  
5     "password" TEXT NOT NULL,  
6     PRIMARY KEY("id" AUTOINCREMENT)  
7 );
```

Рисунок 2.8 – Атрибутна структура Users.db

```

1 CREATE TABLE "Products" (
2     "ID" INTEGER NOT NULL UNIQUE,
3     "NAME" TEXT NOT NULL,
4     "CALORIES" TEXT,
5     "PROTEIN" TEXT,
6     "FATS" TEXT,
7     "CARBOHYDRATES" TEXT,
8     "ALIMENTARY" TEXT,
9     "WATER" TEXT,
10    "ZOLA" TEXT,
11    "A" TEXT,
12    "B1" TEXT,
13    "B2" TEXT,
14    "XOLIN" TEXT,
15    "B5" TEXT,
16    "B6" TEXT,
17    "B9" TEXT,
18    "C" TEXT,
19    "D" TEXT,
20    "E" TEXT,
21    "H" TEXT,
22    "K" TEXT,
23    "PP" TEXT,
24    "KALIY" TEXT,
25    "CA" TEXT,
26    "SI" TEXT,
27    "MG" TEXT,
28    "NA" TEXT,
29    "P" TEXT,
30    "CL" TEXT,
31    "FE" TEXT,
32    "I" TEXT,
33    "CO" TEXT,
34    "MN" TEXT,
35    "CU" TEXT,
36    "MO" TEXT,
37    "SE" TEXT,
38    "F" TEXT,
39    "CR" TEXT,
40    "ZN" TEXT,
41    PRIMARY KEY ("ID" AUTOINCREMENT)
42 );

```

Рисунок 2.9 – Атрибутна структура ProductBD.db

2.7 Аналіз вимог до розробки дизайну програми

Оскільки розроблювана програма створена для аналізу та контролю, то основний акцент потрібно робити в аналітичному стилі, а саме в контентному блоці потрібно розміщувати важливу інформацію для швидкого аналізу, який

не займе багато часу. Тому дизайн сайту створено у мінімалістичному стилі, а саме з максимальним дотриманням його принципів.

Мінімалізм – стиль в дизайні, що характеризується лаконічністю виразних засобів, простотою, точністю і ясністю композиції. Відкидаючи класичні прийоми творчості та традиційні художні матеріали, мінімалісти використовують промислові та природні матеріали простих геометричних форм, нейтральних кольорів (чорний, сірий) і малих обсягів [11].

Програмний мінімалізм на сьогодні дуже популярний. У цього стилю дизайну безліч переваг: користувач завжди ясно розуміє де знаходиться і що йому робити в програмі, увагою користувачів легко керувати, крім того, завдяки мінімалістичному стилю користувач створює менше помилок, а це сприяє до швидкої роботи, завдяки цьому програма мотивує користувача використовувати її повторно, а деякі елементи добре запам'ятовуються.

Основним фактором мінімалізму у дизайні є простий та зрозумілий інтерфейс, який не представляє різних стилів (градієнт, тінь).

Головний принцип в мінімалізмі при створенні дизайну – це не більше трьох кольорів, для монохромних кольорів це правило не стосується. Він також використовується в створенні іконок, дизайну сайтів, картинок.

2.8 Висновок

Отже, для розробки програми аналізу та контролю нутрієнтів, було обрано архітектуру проектування MVVM, яка допомагає досягнути максимальної гнучкості та зручності при користуванні програмою, крім того, було спроектовано базу даних, а саме розроблення архітектури атрибутів в базі даних. В ході виконання другого розділу було проаналізовано вимоги до розробки. При аналізі вимог було обрано мінімалістичний.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз і обґрунтування вибору мови програмування

Для розробки інтерфейсної програми доцільно використати об'єктно-орієнтовану мову програмування. До найпопулярніших із них можна віднести C#, Java та Python.

C # - об'єктно-орієнтована мова програмування. Розроблено групою інженерів компанії Microsoft під керівництвом Андерса Хейлсберг і Скотта Вільтаумота, як мову розробки додатків для платформи Microsoft .NET Framework. Ця мова виявилася досить потужною, тому отримала велику популярність та використовується в проектах будь-якого масштабу: від простої програми до найбільших проектів.

Переваги C#:

- ефективна побудова складних ієрархій класів;
- повторне використання раніше написаного коду;
- зручність в супроводі;
- зменшення кількості логічних помилок;
- легкість в узгодженні різних частин програмного коду;
- створення бібліотек коду;
- використання переваг поліморфізму;
- можливість реалізовувати відомі шаблони проектування.

Недоліки C#:

- неможливо змінити успадковану реалізацію;
- велика розмірність програм;
- збільшення складності програми;
- складність засвоєння початківцями програмістами основ побудови програм [12].

Java - це мова програмування загального призначення, яка слідує парадигмі об'єктно-орієнтованого програмування. Java використовується для настільних, мережових, мобільних і корпоративних додатків. Крім того, це не

тільки мова програмування та її екосистема охоплює майже все що потрібно для ПЗ.

Переваги Java:

- стандарт для корпоративних обчислювальних систем;
- незалежність від платформи;
- мова для розподіленого програмування і комфортної віддаленої спільної роботи;
- автоматичне управління пам'яттю;
- багатопоточність.

Недоліки Java:

- платне комерційне використання;
- низька продуктивність;
- багатослівний і складний код;
- відсутність нативного дизайну для створення інтерфейсу на ПК [13].

Python широко застосовується, як інтерпретована мова для скриптів різного призначення. Як і Ruby, Python має на меті наблизити синтаксис реальної програми, написаної на ній, до опису задачі псевдокоду, що дозволяє програмісту зменшити обсяг програми [14].

Переваги Python:

- відкрита розробка;
- простий і чистий синтаксис;
- досить проста у вивченні, особливо на початковому етапі;
- механізми модульності, які добре продумані;
- відносно все в Python є об'єктами в сенсі ООП.

Недоліки Python:

- не надто вдала підтримка багатопоточності;
- низька продуктивність;
- обмеженість засобів для роботи з базами даних;
- відсутність комерційної підтримки засобів [15].

Загальні особливості мов (табл. 3.1).

Таблиця 3.1 – Порівняння мов програмування

Характеристика	C#	Java	Python
Рік створення	1998	1995	1991
Тип виконання	напівкомпільовані (компілюється C # у CIL)	напівкомпільована	інтерпретований, компільований в байт-код Java
Система типізації	статична, динамічна та явна типізація	статична типізація та явна типізація	динамічна типізація та неявна типізація
Підтримка ООП	+	+	+
Стандарти	ISO ECMA	–	PEP 8
Парадигми програмування:	Мульти- парадигмальний	Мульти- парадигмальний	Мульти- парадигмальний
Управління пам'яттю	Створення об'єктів на стеку, некеровані показчики, ручне управління та прибирання сміття	Прибирання сміття	Прибирання сміття

Отже, враховуючи перераховані особливості та переваги, для розробки було обрано мову програмування C#, крім того, також було обрано мову Python для накопичення контенту в базу даних.

3.2 Аналіз і обґрунтування вибору середовища розробки

Для розробки ПЗ мовою програмування C# та Python на сьогодні існує велика кількість інтегрованих середовищ розробки та редакторів. Відмінність між ними – в тому, що інтегроване середовище завантажує весь проект цілком,

тому може надавати зручну навігацію між його файлами. А редактори редагують конкретний файл і самі файли не знають про зв'язки між іншими файлами та це і є головною складністю при розробці ПЗ. Розглянемо деякі популярні з них, а саме Visual Studio, PyCharm та Subline Text 2.

PyCharm демонструє себе, як інтегроване середовище розробки для мови Python. Середовище PyCharm надає засоби: аналізу, графічний відклачик та засіб для запуску тестів. PyCharm розроблений компанією JetBrains на основі IntelliJ IDEA. PyCharm це крос-платформна інтегроване середовище розробки, яка сумісна з Windows, macOS та Linux.

Системні вимоги Python:

- 64-бітна версія Microsoft Windows 10, 8, 7;
- оперативна пам'ять не менше 4 ГБ;
- рекомендовано двоядерний процесор;
- тип жорсткий диск: SSD;
- вільне місце на жорсткому диску: 1.5 ГБ;
- відеоадаптер який підтримує роздільну здатність 1024×768;
- Python 2.7, Python 3.5 або більш пізня версія.

Visual Studio це лінійка продуктів компанії Microsoft. Visual Studio демонструє себе, як інтегроване середовище і ряд других інструментів. Середовище дає змогу програмувати на великій кількості мов, завдяки цьому можна створити програму, яка спроектована на кількох мовах програмування, крім того, середовище містить: редактор вихідного коду з підтримкою IntelliSense, найпростіший рефакторинг коду та редактор форм для спрощення створенню графічних інтерфейсів.

Системні вимоги Visual Studio:

- 64-бітна версія Microsoft Windows 10, 8, 7;
- оперативна пам'ять не менше 4 ГБ;
- рекомендовано двоядерний процесор;
- тип жорсткий диск: SSD;
- вільне місце на жорсткому диску: 20 ГБ;

- відеоадаптер який підтримує роздільну здатність 1280x720.

Sublime Text – багатоплатформовий текстовий редактор. Підтримує плагіни мовою програмування Python. Інструментарій програми зручний для редагування вихідних текстів програм. Присутня командна палітра для покращення написання коду, крім того, має велику степінь налаштуваності.

Sublime Text підтримує велику кількість мов програмування та має можливість підсвічування синтаксису для C, C ++, C #, CSS, D, Dylan, Erlang, HTML, Groovy, Haskell, Java, JavaScript, LaTeX, Lisp, Lua, Markdown, MATLAB, OCaml, Perl, PHP, Python, Ruby, SQL, TCL і XML. Це додаток мов програмування, які включені в початковий пакет. Користувач має можливість завантажувати пакети для підтримки додаткових мов.

Системні вимоги Sublime Text:

- 32 та 64-бітна версія От Windows XP по Windows 10;
- оперативна пам'ять не менше 1 ГБ;
- рекомендовано двоядерний процесор;
- тип жорсткий диск: SSD;
- вільне місце на жорсткому диску: 1 ГБ.

Порівняння вищеписаних середовищ розробки (табл. 3.2).

Таблиця 3.2 – Порівняння вищеписаних середовищ розробки

Характеристика	Visual Studio	Python	Sublime Text
Мінімальні вимоги оперативної пам'яті	4 Гб	4 Гб	1 Гб
Мінімальні вимоги до процесора	2 Ядра	2 Ядра	2 Ядра
Відлагоджувач -Windows	+	+	–

Продовження таблиці 3.2

Характеристика	Visual Studio	Python	Sublime Text
- Linux	+	+	+
- Mac OS X	+	+	+
Механізм автодоповнення	+	+	+
Статичний аналіз коду	+	+	+
Методи рефакторингу	+	+	+
Можливість створення графічних інтерфейсів	Вбудований в середовище	Не вбудований, але є можливість доповнити бібліотекою	Не вбудований

Враховуючи особливості, наведені вище, а також наявність великої кількості інструментів для швидкої розробки інтерфейсної програми мовою C# для розробки, було обрано Visual Studio. Головним фактором обрання Visual Studio це те, що в середовище інтегрована можливість створювати графічні інтерфейси для програм. Оскільки мову Python будемо використовувати лише для формування контенту в базі даних то цю мову ми будемо використовувати також у Visual Studio.

3.3 Розробка дизайну програми

Оскільки програмний інтерфейс використовує мінімалістичний стиль, тому для вікна аутентифікації було використано два кольори: білий (#ffffff), чорний (#000000) та аквамариновий градієнт. У білому кольорі виконано фон панелі для входу до особистого кабінету та панелі для перемикання режимів, а для фону екрану було використано аквамариновий градієнт (рис. 3.1).

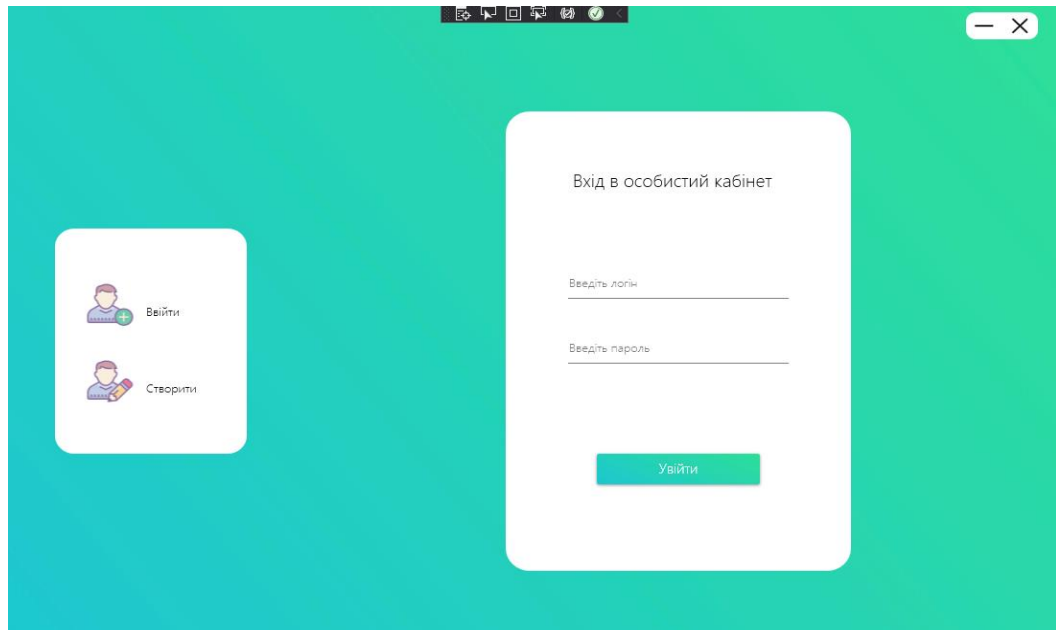


Рисунок 3.1 – Аутентифікаційне вікно

Чорним кольором було позначено шрифт, а сімейство шрифтів було обрано Microsoft YaHei UI Light. Якщо проаналізувати цей шрифт, то можна виділити такі аспекти: шрифт лаконічний, мінімалістичний та легкий в розпізнанні (рис. 3.2).

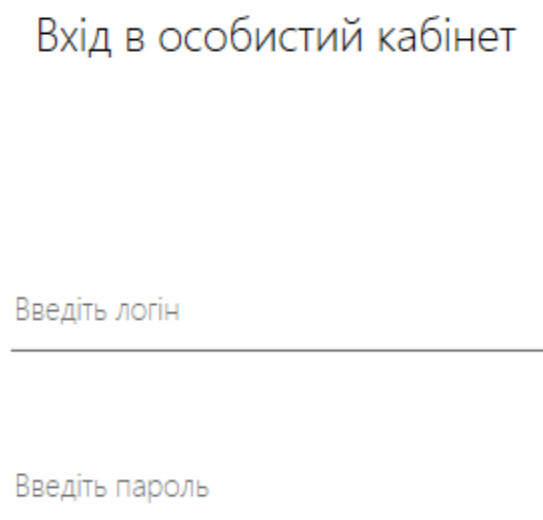


Рисунок 3.2 – Використання шрифту Microsoft YaHei UI Light

Після аутентифікації йде головне вікно на якому буде виконуватись основні дії (рис. 3.3)

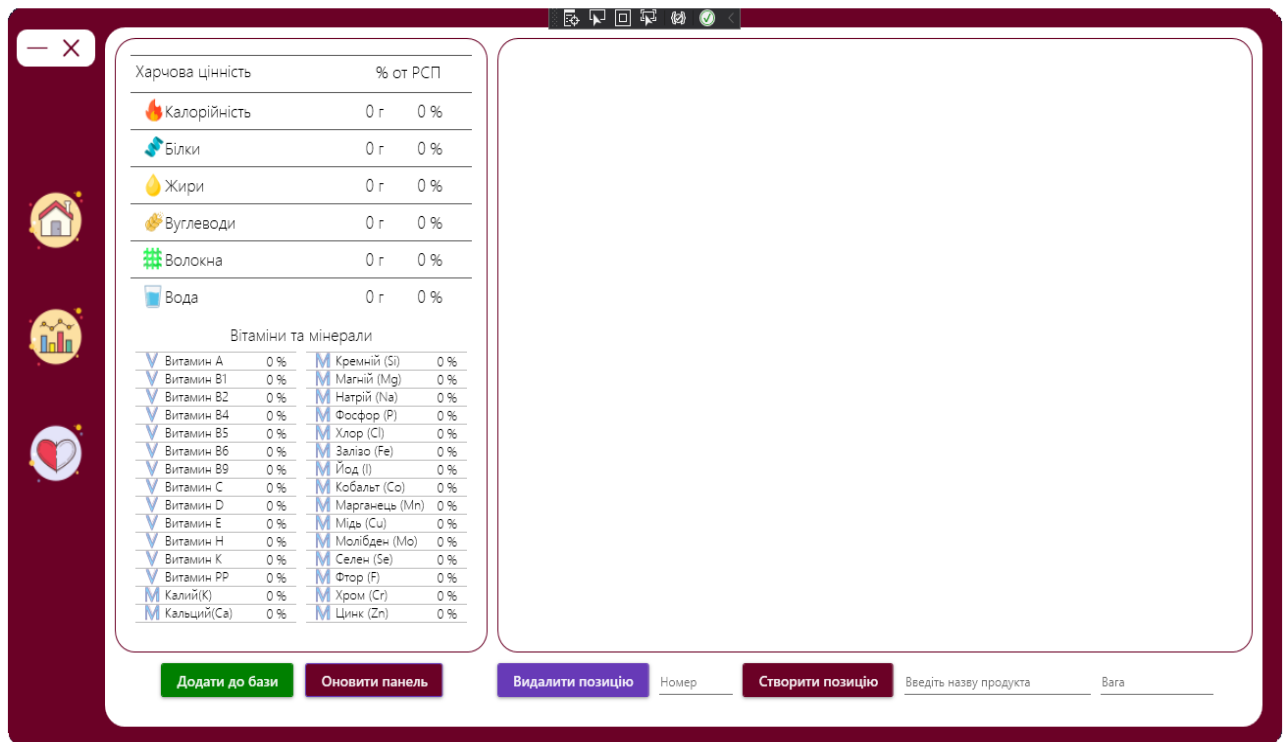


Рисунок 3.3 – Головне вікно для введення данив

Оскільки програма аналітична то під час розробки інтерфейсу було обрано рішення розділити вікно на дві панелі, а саме вікно для динамічного виведення нутрієнтів та вікно на якому зображена інформація доданих продуктів. Якщо розглядати панель меню, було обрано рішення мінімізувати кнопки, але щоб не втратити адаптивність при роботі з програмою, було додано іконки які відображують функціонал ,крім того, при наведенні на іконку спливає підказка. Аналітичні панелі (рис. 3.4, 3.5 та 3.6).

Харчова цінність		% от РСП	
	Калорійність	0 г	0 %
	Білки	0 г	0 %
	Жири	0 г	0 %
	Вуглеводи	0 г	0 %
	Волокна	0 г	0 %
	Вода	0 г	0 %
Вітаміни та мінерали			
	Вітамин А	0 %	 Кремній (Si) 0 %
	Вітамин В1	0 %	 Магній (Mg) 0 %
	Вітамин В2	0 %	 Натрій (Na) 0 %
	Вітамин В4	0 %	 Фосфор (P) 0 %
	Вітамин В5	0 %	 Хлор (Cl) 0 %
	Вітамин В6	0 %	 Залізо (Fe) 0 %
	Вітамин В9	0 %	 Йод (I) 0 %
	Вітамин С	0 %	 Кобальт (Co) 0 %
	Вітамин D	0 %	 Марганець (Mn) 0 %
	Вітамин Е	0 %	 Мідь (Cu) 0 %
	Вітамин Н	0 %	 Молібден (Mo) 0 %
	Вітамин К	0 %	 Селен (Se) 0 %
	Вітамин РР	0 %	 Фтор (F) 0 %
	Калій(K)	0 %	 Хром (Cr) 0 %
	Кальцій(Ca)	0 %	 Цинк (Zn) 0 %

Рисунок 3.4 – Панель для динамічного виведення нутрієнтів

 Головна сторінка		Жири
		Вуглеводи
		Волокна
		Вода
 	Вітаміни	
		Вітамин А
		Вітамин В1
		Вітамин В2
		Вітамин В4
		Вітамин В5
		Вітамин В6
		Вітамин В9
		Вітамин С

Рисунок 3.5 – Панель меню

Макарони з борошна	125 гр
Антилопа , запечена	450 гр

Видалити позицію
Номер
Створити позицію
Введіть назву продукту
Вага

Рисунок 3.6 – Панель для відображення доданої інформації

Якщо проаналізувати рішення розділу панелі, то можна виділити такі аспекти: інтерфейс дає можливість швидкого аналізу, панель додавання нової інформації не нагромаджена, це сприяє до швидкого редагування позиції, текстові бокси та кнопки на панелі меню мають підказки, які допоможуть новому користувачеві.

3.4 Розробка модулів програми

Розроблювана програма з графічним інтерфейсом має модульну структуру, тому опишемо детально розробку та принципи роботи кожного із модулів.

Модуль “Реєстрація” виконується сполученням з модулем “База”. Модуль дозволяє реєструвати нових користувачів. Графічне віно модулю “Реєстрація” (рис. 3.7).

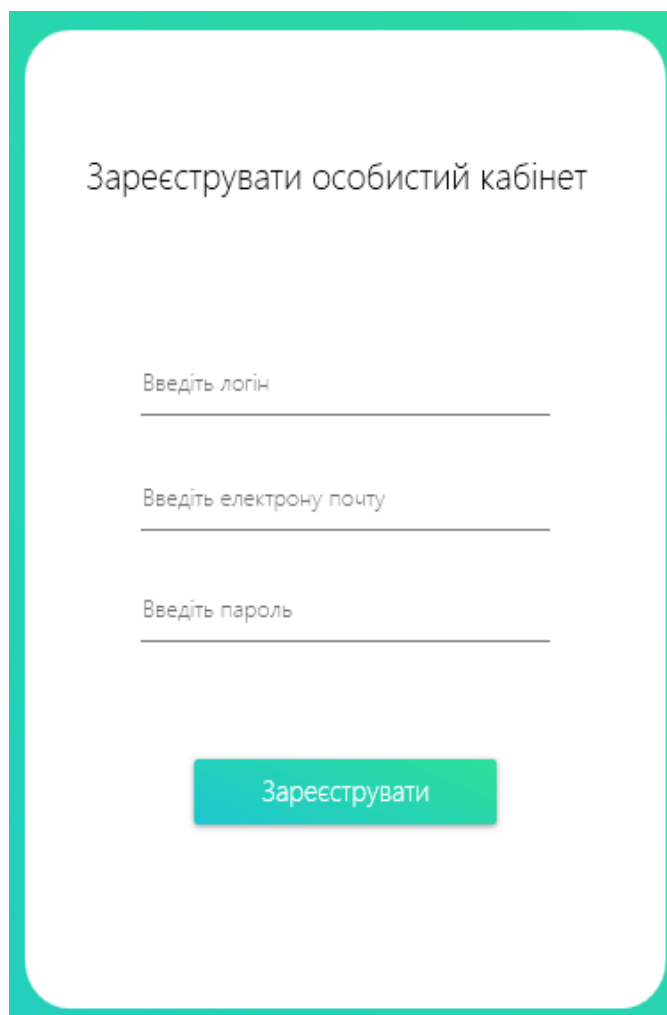
The image shows a registration form within a light blue rounded rectangle. At the top, the text 'Зареєструвати особистий кабінет' is centered. Below it are three input fields, each with a label above it: 'Введіть логін', 'Введіть електронну пошту', and 'Введіть пароль'. At the bottom of the form is a blue button with the text 'Зареєструвати' in white.

Рисунок 3.7 – Графічне віно модулю “Реєстрація”

Функціонал модулю представлений у вигляді класу `RegistrationPage`, через який здійснюється реєстрація нового користувача. Далі наведено розроблену функцію та опис.

Функція `Registration` викликається після натиску по кнопці, функція отримує із текстових блоків три параметра, а саме логін, пароль та пошту. Після отримання параметрів вона перевіряє на дотримання правил: довжина логіну не менше п’яти символів, довжина паролю не менше восьми символів та довжина пошти не менше п’яти символів також з обов’язковими символами точки і собачки. Після проходження перевірки інформація додається до бази даних завдяки модулю “Бази даних”.

Опис використаних внутрішніх структур даних в класі `RegistrationPage` (табл. 3.3).

Таблиця 3.3 – Опис використаних внутрішніх структур даних в класі RegistrationPage

Назва змінної	Тип структури даних змінної	Призначення змінної
login	string	Зміна, яка зберігає ведену інформацію
Password	string	Зміна, яка зберігає ведену інформацію
Email	string	Зміна, яка зберігає ведену інформацію
Status	int	Зміна, яка відповідає за валідацію

Модуль “Аутентифікація” виконується сполученням з модулем «База». Модуль дозволяє виконувати вхід, а після успішної операції реалізується створення нового вікна. Графічне вікно модулю “Аутентифікація (рис. 3.8).

Вхід в особистий кабінет

Введіть логін

Введіть пароль

Увійти

Детальний опис: Це графічне інтерфейс (GUI) для модуля аутентифікації. Він має білий фон з округленими кутами, оточений синьою рамкою. У центрі розташовано заголовок "Вхід в особистий кабінет". Під ним є два текстові поля для введення даних: "Введіть логін" і "Введіть пароль". Нижче цих полів розташований синій кнопка з білим текстом "Увійти".

Рисунок 3.8 – Графічне вікно модулю “Аутентифікація ”

Функціонал модулю представлений у вигляді класу LoginPage. Далі наведено опис функції, що здійснює перевірку та викликання модуля "База даних" для вивантаження зразків, які будуть використовуватись для перевірки.

Функція Button_Authentication викликається після натиску по кнопці, функція отримує із текстових блоків два параметра, а саме логін та пароль. Після отримання параметрів вона перевіряє на існування такого логіну та паролю в базі даних. Після підтвердження інформації буде створюватись нове графічне вікно. Для виведення контенту вікно буде використовувати клас TwoVieModule.

Опис використаних внутрішніх структур даних в функції Button_Authentication (табл. 3.4).

Таблиця 3.4 – Опис використаних внутрішніх структур даних в функції Button_Authentication

Назва змінної	Тип структури даних змінної	Призначення змінної
Login	string	Зміна, яка зберігає ведену інформацію
Password	string	Зміна, яка зберігає ведену інформацію
Status	int	Зміна, яка відповідає за валідацію
Auth	object	Екземпляр класу Authentication відповідає за перевірку логіну та паролю
NewWindows	object	Екземпляр класу Main відповідає за створення головного вікна

Клас TwoVieModule дозволяє програмі використати модель MVVM, а саме можливість розділяти контент на сторінки, а ці сторінки використовують всі інші модулі. Далі наведено опис метода конструктора, що здійснює можливість перемикає сторінки.

Конструктор викликається після натиску по кнопці, конструктор отримує нову сторінку, яка змінює минулу сторінку.

Опис використаних внутрішніх структур даних в класі TwoVieModule (табл. 3.5).

Таблиця 3.5 – Опис використаних внутрішніх структур даних в класі TwoVieModule

Назва змінної	Тип структури даних змінної	Призначення змінної
InfoVM	object	Екземпляр класу InfoViewModel відповідає за збереження головної сторінки
DiaVM	object	Екземпляр класу DiaViewModel відповідає за збереження сторінки формування графіку
PrimeryViewTwo	object	Екземпляр, який приймає сторінку, яка буде виведена на головне вікно
HealVM	object	Екземпляр класу DiaViewModel відповідає за збереження сторінки формування списку
primery_	object	Об'єкт який зберігає значення сторінки

Модуль “Створення раціону та додавання до бази”, використовується для створення раціону, виведення швидкого аналізу та збереження щоденної інформації. Графічне віно модуля (рис. 3.9).

Харчова цінність % от РСП

Калорійність	22.5 ккал	1.1 %
Білки	1.10 г	0.9 %
Жири	0.25 г	0.3 %
Вуглеводи	3.36 г	1.9 %
Волокна	1.50 г	7.5 %
Вода	118.15 г	3.9 %

Вітаміни та мінерали

Вітамін А	6 %	Кремній (Si)	0 %
Вітамін В1	0 %	Магній (Mg)	2 %
Вітамін В2	0 %	Натрій (Na)	1 %
Вітамін В4	1 %	Фосфор (P)	3 %
Вітамін В5	0 %	Хлор (Cl)	0 %
Вітамін В6	0 %	Залізо (Fe)	0 %
Вітамін В9	2 %	Йод (I)	0 %
Вітамін С	2 %	Кобальт (Co)	0 %
Вітамін D	0 %	Марганець (Mn)	0 %
Вітамін Е	0 %	Мідь (Cu)	8 %
Вітамін Н	0 %	Молібден (Mo)	0 %
Вітамін К	1 %	Селен (Se)	0 %
Вітамін РР	0 %	Фтор (F)	0 %
Калій (K)	33 %	Хром (Cr)	0 %
Кальцій (Ca)	1 %	Цинк (Zn)	0 %

Помідор (томат) 125 г

Очі оленя карібу (Аляска)

Додати до бази Оновити панель Видалити позицію Номер Створити позицію Очі Вага

Рисунок 3.9 – Графічне віно модулю “Створення раціону та додавання до бази ”

Модуль надає такий функціонал:

- додавання нових записів на панель;
- видалення записів;
- відновлювання статистики;
- додавання статистики до БД.

Модуль представлений у вигляді класу InfoPage, через який здійснюються основні дії з інтерфейсом та з базою даних.

Додавання нових записів на панель. Далі наведено опис функцій.

Функція Button_Add викликається після натиску по кнопці “Створити позицію”. Функція отримує два параметра назву та масу. Функція перевіряє на валідацію, а саме довжина назви та маси повинна бути більше одного символу. Після проходження валідацію на панелі створюються текстові блоки, які містять назву та масу продукту.

Опис використаних внутрішніх структур даних в функції Button_Add (табл. 3.6).

Таблиця 3.6 – Опис використаних внутрішніх структур даних в класі Button_Add

Назва змінної	Тип структури даних змінної	Призначення змінної
textName	string	Зміна, яка зберігає назву продукту
textMassa	string	Зміна, яка зберігає масу продукту
txtOne	object	Екземпляр класу TextBox відповідає за виведення інформації на панель

При додаванні продукту змінюється панель, а саме інформація додається до панелі. Зміни на панелі (рис. 3.10).

Помідор (томат) 125 г

Макарони з борошна 1 сорт 135

Видалити позицію

Номер

Створити позицію

Введіть назву продукта

Вага

Рисунок 3.10 – Зміни на панелі

Видалення записів з панелі. Далі наведено опис функцій.

Функція `Button_Remove` викликається після натиску по кнопці “Видалення позицію”. Функція використовує один параметр, а саме номер позиції, котру треба видалити. Після видалення позиції панель відновлюється. Також присутня валідація на правильність введеного номеру.

Опис використаних внутрішніх структур даних в функції `Button_Add` (табл. 3.7).

Таблиця 3.7 – Опис використаних внутрішніх структур даних в функції `Button_Remove`

Назва змінної	Тип структури даних змінної	Призначення змінної
<code>Id</code>	<code>int</code>	Збереження номеру
<code>register</code>	<code>List<int></code>	Збереження індексів позицій
<code>TextPanelTwo</code>	<code>object</code>	Використовується для стягнення списку мас
<code>TextPanelOne</code>	<code>object</code>	Використовується для стягнення списку назв
<code>status</code>	<code>bool</code>	Використовується для валідації <code>id</code>

При відновленні аналітичної панелі, інформація додається до панелі. Відновлювана аналітична панель (рис. 3.11).

Харчова цінність		% от РСП	
 Калорійність	154 ккал		7.8 %
 Білки	4.00 г		3.4 %
 Жири	0.80 г		0.9 %
 Вуглеводи	32.60 г		18.1 %
 Волокна	2.80 г		14.0 %
 Вода	157.20 г		5.2 %
Вітаміни та мінерали			
 Вітамін А	1 %	 Кремній (Si)	11 %
 Вітамін В1	0 %	 Магній (Mg)	5 %
 Вітамін В2	0 %	 Натрій (Na)	1 %
 Вітамін В4	2 %	 Фосфор (P)	13 %
 Вітамін В5	0 %	 Хлор (Cl)	13 %
 Вітамін В6	0 %	 Залізо (Fe)	0 %
 Вітамін В9	2 %	 Йод (I)	1 %
 Вітамін С	4 %	 Кобальт (Co)	1 %
 Вітамін D	0 %	 Марганець (Mn)	0 %
 Вітамін Е	0 %	 Мідь (Cu)	31 %
 Вітамін Н	0 %	 Молібден (Mo)	2 %
 Вітамін К	0 %	 Селен (Se)	0 %
 Вітамін РР	0 %	 Фтор (F)	7 %
 Калій(K)	126 %	 Хром (Cr)	2 %
 Кальцій(Ca)	2 %	 Цинк (Zn)	0 %

Додати до бази

Оновити панель

Рисунок 3.11 – Відновлювана аналітична панель

Реалізація пошукового меню, яке надає допомогу користувачу, а саме додавання продукту до панелі.

Опис використаних внутрішніх структур даних в функції ProductText_KeyUp (табл. 3.8).

Таблиця 3.8 – Опис використаних внутрішніх структур даних в функції ProductText_KeyUp

Назва змінної	Тип структури даних змінної	Призначення
isFound	bool	Дає доступ на додавання Label до панелі

Продовження таблиці 3.8

Назва змінної	Тип структури даних змінної	Призначення
		бази даних
find	string	Зберігає схожі назви продуктів
tb1	object	Виводить повідомлення про те що продукту не має в базі даних
block	object	Створює Label на панелі
tb_one	object	Встановлює border
lb	object	Встановлює задній фон панелі

При веденні тексту з'являється бар з підказкою назви продуктів які є в базі даних. Бар з підказкою (рис. 3.12).

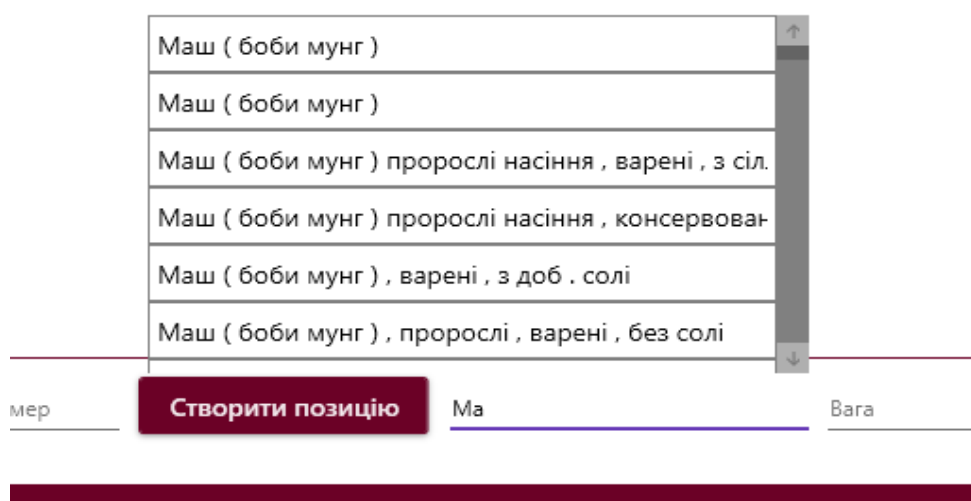


Рисунок 3.12 – Бар з підказкою

Модуль "створення графіків", використовується для створення аналітичного графіка для швидкого аналізу. Графічне вікно модуля (рис. 3.13).

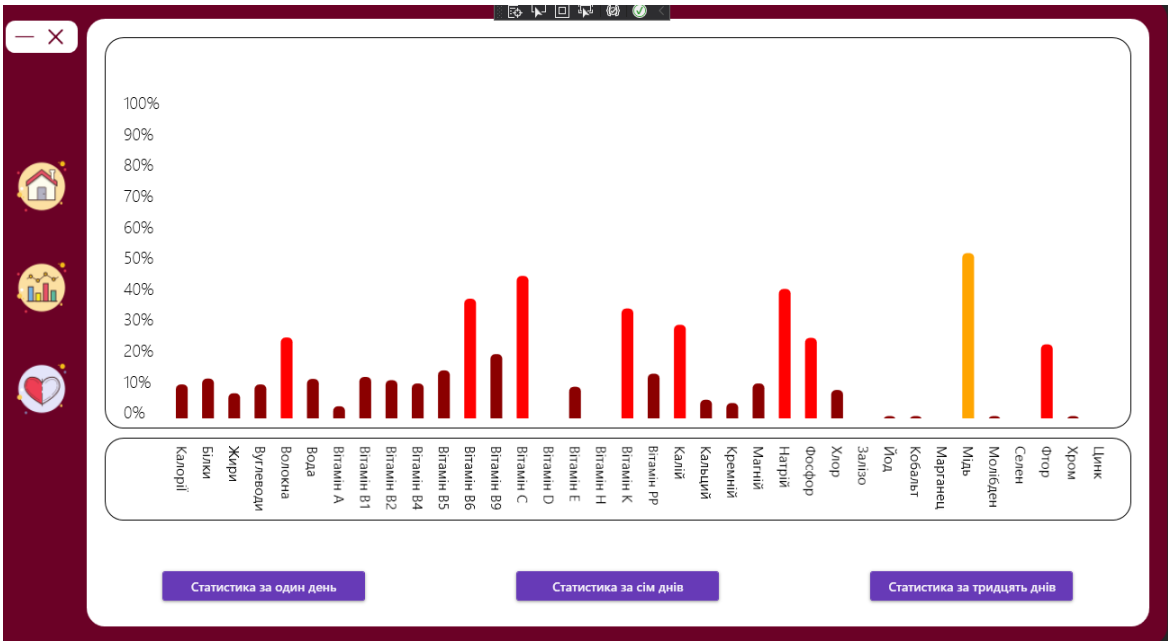


Рисунок 3.13 – Відновлювана аналітична панель графіку

Модуль представлений у вигляді класу `DiagnostickPage`, через який, здійснюється основні дії з інтерфейсом та з базою даних. Модуль дає можливість будувати графік.

Функція `MethodBuild` викликається після натиску по кнопці. Функція отримує один параметр, а саме інтервал для будування графіка.

Опис використаних внутрішніх структур даних в функції `MethodBuild` (табл. 3.9).

Таблиця 3.9 – Опис використаних внутрішніх структур даних в функції `MethodBuild`

Назва змінної	Тип структури даних змінної	Призначення
id	string	Зберігає id користувача
procent	List<Double>	Зберігає список процентів вживаних нутрієнтів
StackPanelList	List<string>	Зберігає список Border
TextPanelOne	object	Зафарбовує Border

Для користувача була реалізована можливість обирати інтервали формування графіку.

Методи які передають інтервали в функцію MethodBuild:

- Button_One();
- Button_Seven();
- Button_Thirty().

Модуль ”формування списку можливих захворювань”, використовується для формування стислого списку можливих захворювань для швидкого аналізу, крім того, модуль надає змогу експортувати дані в інший екземпляр програми. Графічне вікно списку можливих захворювань (рис. 3.14).

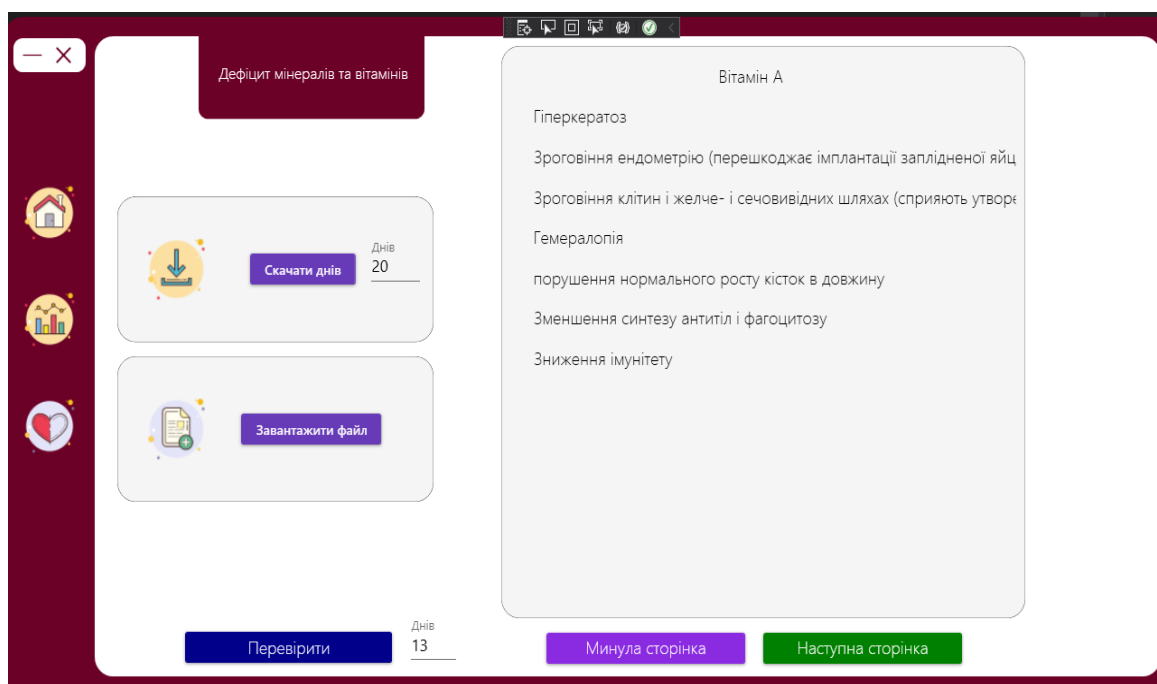


Рисунок 3.14 – Графічне вікно списку можливих захворювань

Модуль представлений у вигляді класу InfoHeal, через який, здійснюється основні дії з інтерфейсом та з базою даних. Модуль дає можливість будувати, імпортувати та експорту стислого списку.

Далі наведено опис функцій які використовувалися в класі InfoHeal:

- button_check — функція, яка перевіряє на валідацію значення та запускає метод Start();

- `button_tomorrow` — функція, яка відображує наступний список можливих захворювань;
- `button_pass` — функція, яка відображує минулий список можливих захворювань;
- `start()` — функція викликає метод `ParseDateOne()` та передає повернене значення в функцію `CountPageMehtod()`;
- `parseDateOne()` — функція отримує інформацію з бази даних;
- `button_download` — функція викликає файловий менеджер для збереження файлу можливих захворювань у вигляді процентів;
- `button_input` — функція викликає файловий менеджер для добавлення файлу в програму.

Реалізація методу експорту (рис. 3.15).

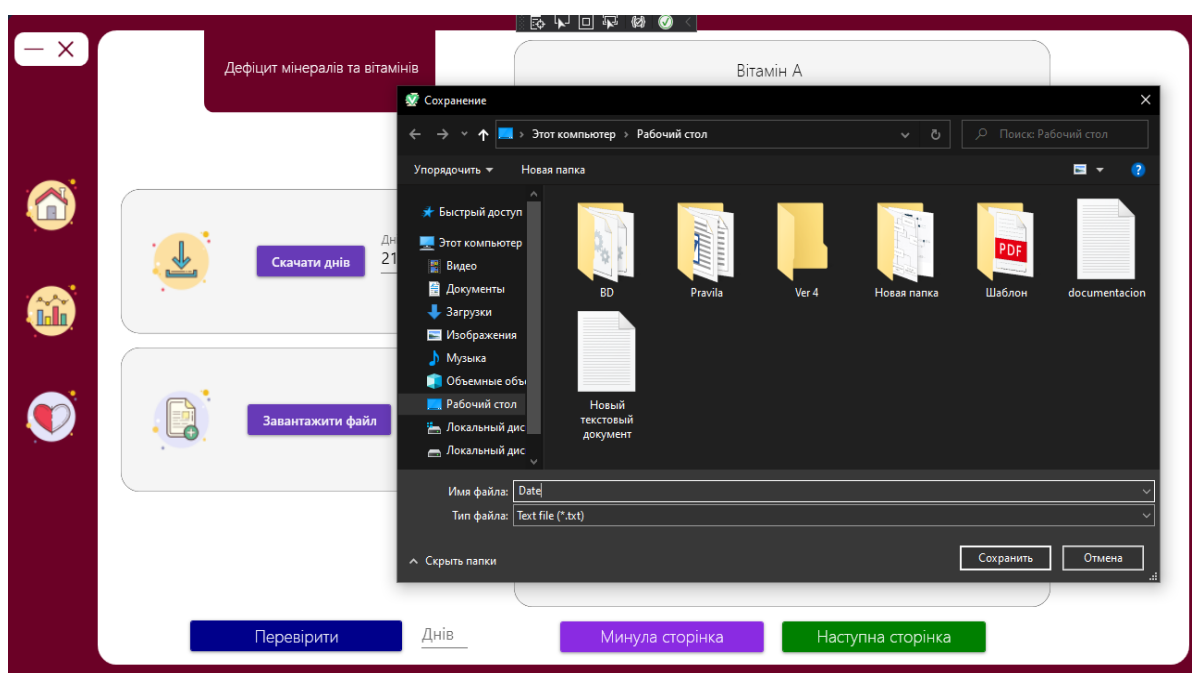


Рисунок 3.15 – Графічне вікно експорту

Реалізація методу імпорту (рис. 3.16).

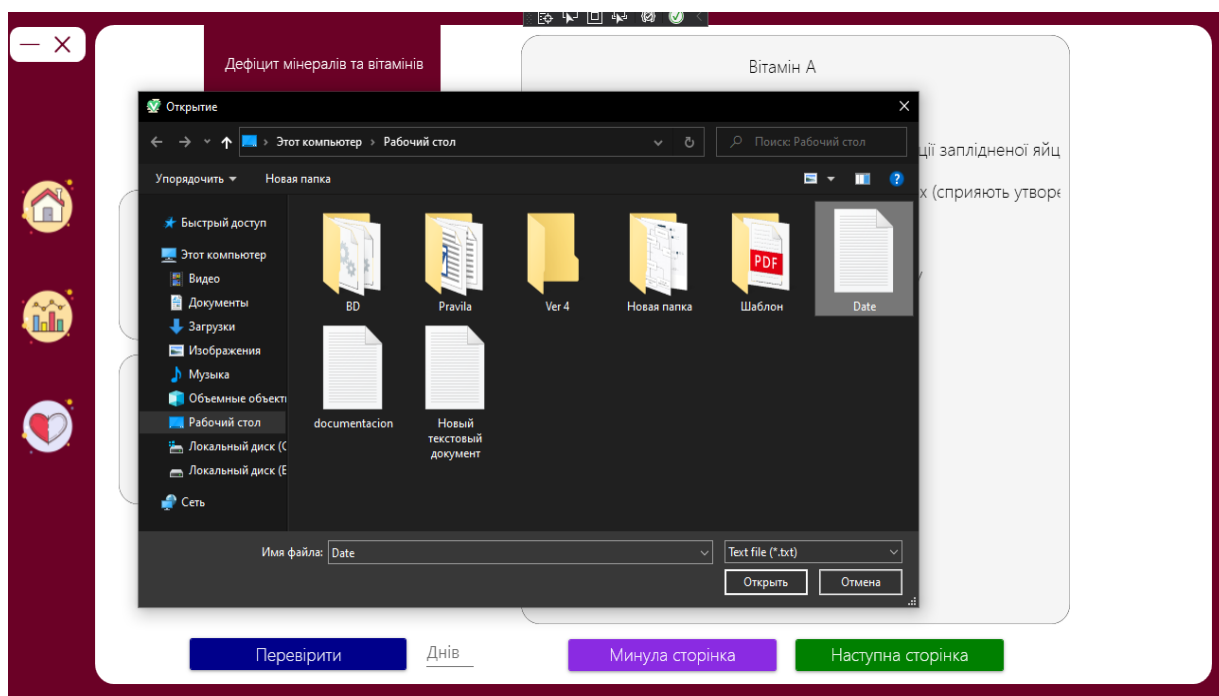


Рисунок 3.16 – Графічне вікно імпорту

3.5 Висновок

В ході виконання третього розділу було здійснено пошук оптимальної мови програмування для виконання поставленого завдання. В результаті було обрано мову програмування C# та Python, оскільки вони відповідають основним вимогам краще ніж інші мови програмування, крім того, мову Python ми використовували для початкового формування баз даних, а саме базу продуктів.

Для обраної мови програмування необхідно було вибрати середовище розробки, вибір упав на Visual Studio, тому що в середовищі є можливість розроблювати графічні інтерфейси без встановлення додаткових бібліотек.

Також розроблено основні модулі програми та описано їх функціонал з наведеним кодом та рисунками.

4 ЕКСПЛУАТАЦІЯ ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ

4.1 Тестування програми

Для розроблюваного ПЗ було проведене повноцінне тестування: перевірка кнопок, перевірка відповідності текстів заданій темі, перевірка відповідності до індивідуального завдання, перевірка на адаптивність, перевірка на зрозумілість інтерфейсу, коректність ведення даних та перевірки інформації в базі даних.

Перевірка інформаційного наповнення, показала відповідність текстів заданій тематиці, відсутність орфографічних, стилістичних та пунктуаційних помилок, також була перевірена інформація в базі даних, крім того, було виконано мовну перевірку. Результат перевірки без помилок.

Під час перевірки пригодності до використання, виявленню, що ПЗ має зручний та інтуїтивний інтерфейс, має гарно збудований інтерфейс, а саме добре підібрана кольорова гама та не складна логічна структура, тим самим покращує інтегрованість нового користувача з програмою.

Були проведені тести на валідацію введеної інформації. Результати валідації (рис. 4.1, 4.2 та 4.3).

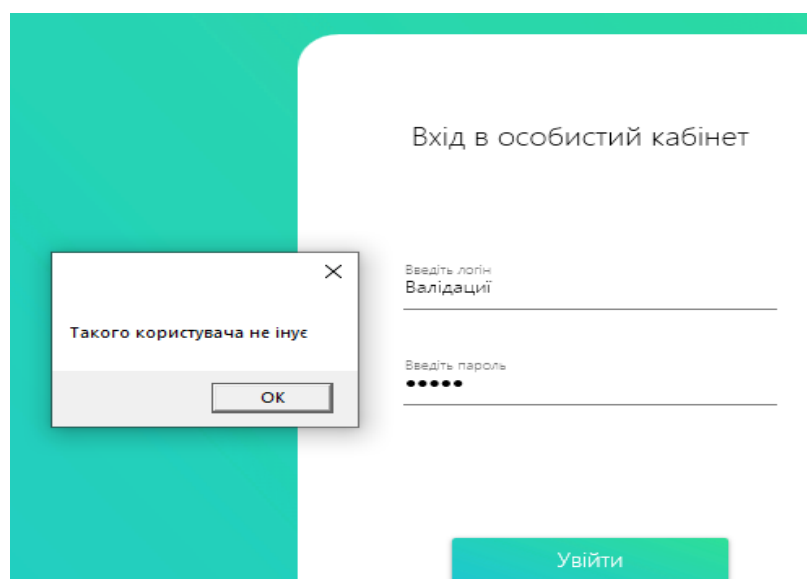


Рисунок 4.1 – Валідація аутентифікації

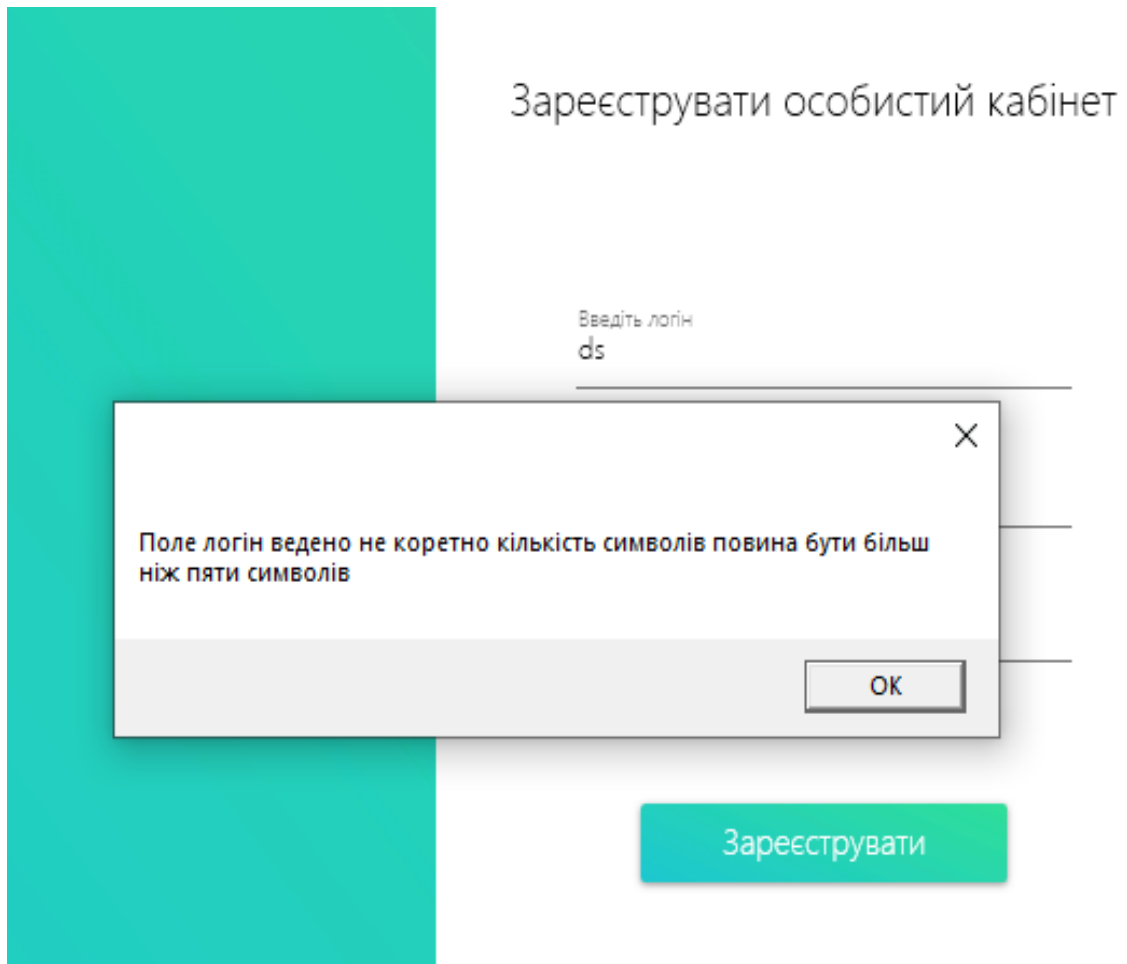


Рисунок 4.2 – Валідація реєстрації

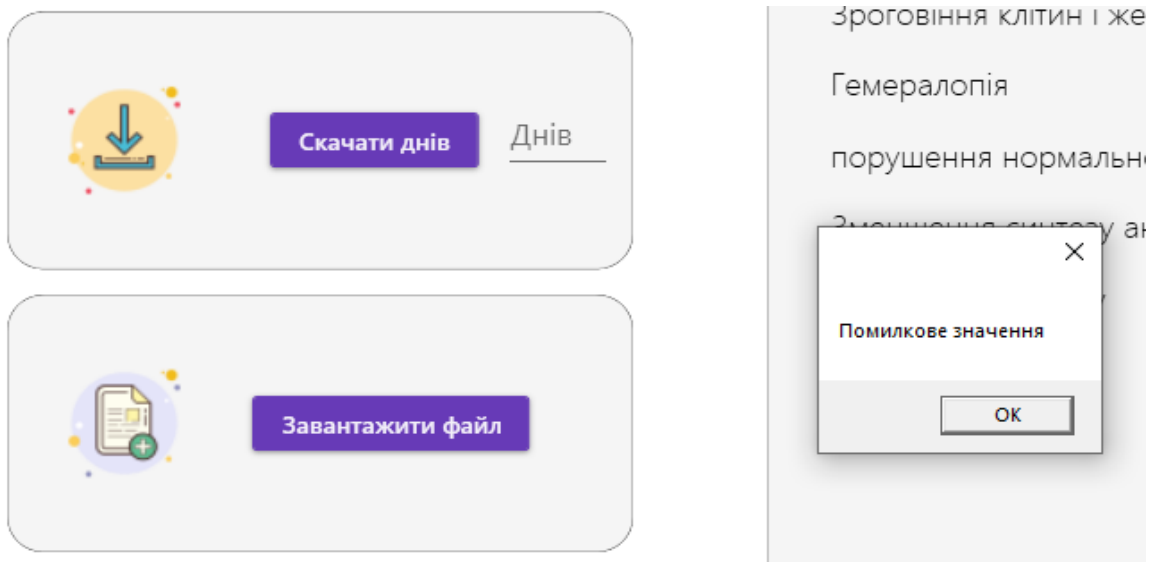


Рисунок 4.3 – Валідація експорту файлу

Тестування ПЗ призначено для того, щоб зробити процес відвідування програм приємніше – отже, тестування має велике значення. Воно являє собою поєднання людського управління і можливостей комп'ютера. Тестування дозволить вам створити якісну та оптимізовану програму.

Також були проведенні тести на VM. Результат тестів це мінімальні вимоги для запуску.

Мінімальні вимоги такі:

- рекомендована оперативна пам'ять 1 Гб;
- рекомендовано двоядерний процесор;
- тип жорсткий диск: SSD;
- вільне місце на жорсткому диску: 200 мб;
- Підтримка ОС: - Windows від 7 до 10 версії.

Під час перевірки кнопок та полів для введення інформації, ніяких помилок у роботі кнопок і полів не виявлено. Всі кнопки правильно викликають потрібні методи.

У тому числі була проведена перевірка перемикання між сторінками, ніяких помилок не виявлено.

Для перевірки роботи "Toolkit", введемо назву продукту. Результат "Toolkit" (рис. 4.4). Після цього відбувається перевірка та виводиться список подібних продуктів.

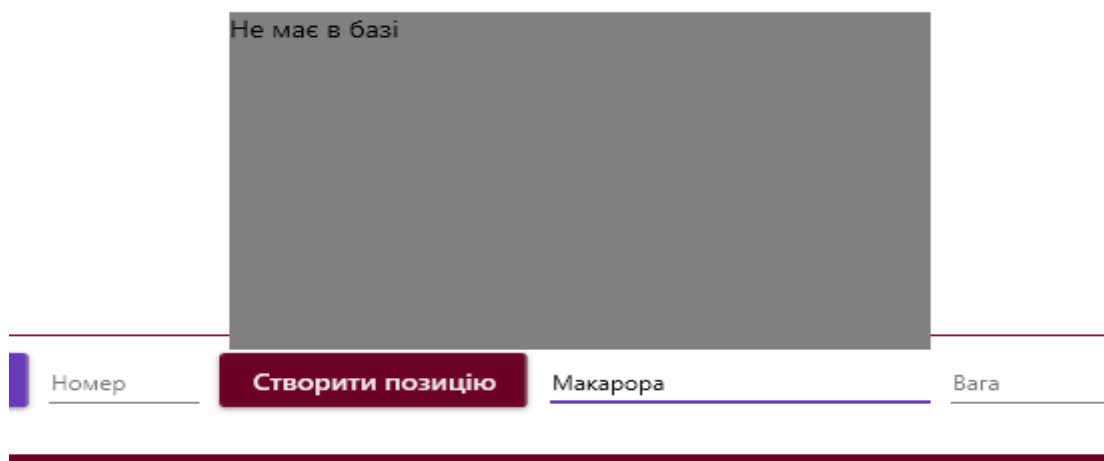


Рисунок 4.4 – Валідація " Toolkit "

4.2 Опис програмного забезпечення

Назва програми Nutritionist.

Мова програмування C#.

Призначення програми - це швидкий аналіз вжитих вітамінів та мікроелементів.

Застосування програмного забезпечення в медичних закладах в області дієтології або в домашньому використанні

Методи, які були застосовані, статична характеристика кожного продукту, методи для побудови графіків та стислого списку.

Функціональні обмеження такі:

- Реєстрація нового користувача;
- Авторизація користувача;
- Введення даних для аналізу;
- Побудова графіку нутрієнтів;
- Побудова стислого списку можливих захворювань;
- Експорт та імпорт стислого списку.

Мінімальні вимоги такі:

- рекомендована оперативна пам'ять 512 Гб;
- рекомендовано двоядерний процесор;
- тип жорсткий диск: SSD;
- вільне місце на жорсткому диску: 200 мб;
- Підтримка ОС: - Windows від 7 до 10 версії.

4.3 Тестування в критичних в середовищах

Для тестування було взято операційну систему Windows 7, 8 та 10, різну виділену оперативну пам'ять, певна кількість ядер на процес. Призначення критичного тестування це те, щоб отримати час, за який виконується певна дія,

в нашому прикладі, це процес авторизація в моменті між закриттям та створенням нового вікна. Порівняння вищеописаного тесту процесу (табл. 4.1).

Таблиця 4.1 – Тест процесу авторизації

Системні характеристики	Windows 7	Windows 8.1	Windows 10
Оперативна пам'ять 256 мб Процесор 1 ядра	2.4 секунд	3.0 секунд	3.1 секунд
Оперативна пам'ять 512 мб Процесор 1 ядра	2 секунд	2.5 секунд	2.4 секунд
Оперативна пам'ять 256 мб Процесор 2 ядра	2.3 секунд	2.8 секунд	2.7 секунд
Оперативна пам'ять 512 мб Процесор 2 ядра	1.9 секунд	2.3 секунд	2.0 секунд

4.4 Висновок

Описано основні методики для проведення повноцінного тестування програми контролю вітамінів та мікроелементів "Nutritionist". Під час проведення тестування програми виявлено, що програма інтуїтивна та має зручний інтерфейс, сайт зручний для користувача, має зручну та логічну структуру. Також були проведенні тестування в екстремальних середовищах, результат цих тестувань це те що програма може працювати і в поганих умовах, але буде гальмувати, а це не відповідає вимогам швидкого аналізу. Після проведених тестувань можна зробити висновок, що загалом ПЗ працює коректно та відповідає поставленим до нього вимогам.

ВИСНОВКИ

Під час виконання бакалаврської роботи було розроблено програму з графічним інтерфейсом "Nutritionist".

Під час дослідження було виявлено три проблеми; це мала кількість людей, які контролюють свій рівень нутрієнтів, важко згрупувати інформацію цілком, наприклад, щоб це зробити потрібно шукати інформацію на різних ресурсах та остання проблема, це те що аналоги ідеї існують в загалом у web-технологіях та на мобільні програми.

Для розв'язання проблем було створено програму вона поєднала розв'язання трьох проблем, а саме групування інформації в одному місці, створення програми на Windows та завдяки розв'язанню двох проблем, зацікавлених користувачів стане більше.

Програму "Nutritionist" можна використовувати в медичних закладах, а саме для лікарів дієтологів, педіатрів та сімейних лікарів. Результат цієї програми це контроль та можливість заздалегідь уникнути можливих захворювань.

Проаналізовано сучасні технології у створенні програмного забезпечені. Найпопулярнішою тенденцією останніх років є мінімалістичний дизайн, який виключає нагромадження об'ємних об'єктів.

Проаналізовано наявні засоби розробки програм: порівняння об'єктно-орієнтовані мови програмування C#, Java, Python та різноманітні середовища розробки та редакторів. Розроблені основні модулі для швидкого аналізу, створення раціону та додавання до бази, створення графіків інтервального аналізу та створення стислого списку можливих захворювань.

Розроблено дизайн програми згідно з основними принципами мінімалістичного дизайну. Створено інтерфейси сторінок аутентифікації, реєстрації, введення даних, створення графіку та створення стислого списку.

Протестована програма на перевірку введеної інформації та придатність ПЗ в екстремальних умовах, а саме зменшення рекомендованих вимог. В ході

тестування виявлено, що програма працює коректно, крім того, було зроблено опис програми.

Для готового програмного продукту було проведене повноцінне тестування: перевірка роботи кнопок, перевірка відповідності текстів заданій темі, перевірка відповідності програми до індивідуального завдання, перевірка на поведінку в екстремальних середовищах, а саме отримання часу для аналізу та природність до використання.

Отже, в ході виконання бакалаврської роботи було продемонстровано практичні навички створення та розробки програм на операційну систему Windows.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Основи харчування: підручник/ [М.І. Кручаниця, І.С. Миронюк, Н.В. Розумикова, В.В. Кручаниця, В.В. Брич, В.П.]. – Ужгород: Вид-во УжНУ “Говерла”, 2019. – Розд. 1. – С. 9–58.
2. Кравченко Г.Б. Гіповітаміноз [Електрон. ресурс] / Г.Б Кравченко. – Режим доступу: <https://www.pharmencyclopedia.com.ua/article/2957/gipovitaminoz>.
3. Кравченко Г.Б. Гіпервітаміноз [Електрон. ресурс] / Г.Б Кравченко. – Режим доступу: <https://www.pharmencyclopedia.com.ua/article/3254/gipervitaminoz>.
4. Мой здоровый рацион. Дневник питание [Электронной ресурс]. – Режим доступа: https://health-diet.ru/diary/foodDiary?utm_source=hd&utm_medium=content&utm_campaign=calculatorList&utm_term=foodDiary.
5. Cronometer. Diary [Electronic resource]. – Access mode: <https://cronometer.com/#diary>.
6. Fit Audit. Продукты [Электронной ресурс]. – Режим доступа: <https://fitaudit.ru/food>.
7. Введения в WPF в Visual Studio [Электронной ресурс]. – Режим доступа: <https://u.to/u0xNGw>.
8. Xaml overview [Electronic resource]. – Access mode: <https://docs.microsoft.com/en-us/dotnet/desktop/wpf/xaml/?view=netdesktop-5.0>.
9. A tour of the C# language [Electronic resource]. – Access mode: <https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/>.
10. Буч Г. Язык UML. Руководство пользователя [Электронный ресурс]. – Режим доступа: <http://needworld.ru/attachments/buch-g-rambo-d-jakobson-i-jazyk-uml-rukovodstvo-polzovatelja-2-e-izdanie-2006-pdf.22103/>.
11. Джамса К. Эффективный самоучитель по креативному Web-дизайну[Текст] / К. Джамса – Санкт-Петербург: ДиаСофтЮп, 2005. – 672 с..

12. Наследование. Базовые понятия. Преимущества и недостатки [Электронный ресурс]. – Режим доступа: <https://www.bestprog.net/ru/2020/02/24/c-inheritance-basic-concepts-advantages-and-disadvantages-general-form-the-simplest-examples-access-modifier-protected-ru/>.

13. The Good and the Bad of Java Programming [Electronic resource]. – Access mode: <https://www.altexsoft.com/blog/engineering/pros-and-cons-of-java-programming/>.

14. PHP, Ruby, Python – краткая характеристика языков программирования [Электронный ресурс]. – Режим доступа: http://www.internettechnologies.ru/articles/article_1991.html.

15. Саммерфилд М. Программирование на Python 3 [Текст] / М.Саммерфилд. : – Санкт-Петербург: Символ-Плюс, 2009. – 608 с.

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення для контролю споживання мікроелементів та вітамінів може використовуватися в медичному, або для особистого користування.

A.1 Підставою для розробки

Підставою для розробки програмного забезпечення є завдання на дипломну кваліфікаційну роботу бакалавра на тему: “Розробка програмного забезпечення для контролю споживання мікроелементів та вітамінів”. Затверджене наказом №103 від 30.03.2021 р. по Національному університету “Запорізька політехніка”.

A.2 Призначення розробки

Функціональне призначення: програма дозволяє створювати щоденний раціон та контролювати споживаних мікроелементів та вітамінів.

A.3 Вимоги до програми чи програмному виробу

A.3.1 Вимоги до функціональних характеристик

Програма повинна забезпечувати:

- реєстрацію нового користувача;
- авторизація в програмі;
- занесення продуктів для аналізу;
- аналіз продуктів на склад вітамінів та мінералів;
- будування графіку;
- будування стислого списку можливих захворювань;
- експорт та імпорт стислого списку.

A.3.2 Вимоги до надійності

Надійне функціонування системи має забезпечуватись наступними факторами:

- безперебійне живлення апаратно-технічних засобів;
- система повинна мати можливість відновлювати працездатність програми при збоях.

A.3.3 Вимоги експлуатації

Для забезпечення експлуатації програмного забезпечення необхідно встановити наступні програмні застосунки:

- Підтримка ОС: - Windows від 7 до 10 версії.
- .NET Core версії 2.2.0.

A.3.4 Вимоги до складу й параметрів технічних засобів

Мінімальні системні вимоги:

- рекомендована оперативна пам'ять 512 Гб;
- рекомендовано двоядерний процесор 2.5 GHz;
- тип жорсткий диск: SSD;
- вільне місце на жорсткому диску: 200 мб.

A.3.5 Вимоги маркування й упакування

Програма може бути записана на оптичному, жорсткому магнітному диску, або USB-носії. На упаковці, або на відповідному пристрої має бути вказано назву роботи “Програмне забезпечення для контролю споживання мікроелементів та вітамінів”.

А.3.6 Вимоги транспортування й збереження

Вимоги до транспортування та збереження повністю співпадають з вимогами до транспортування та збереження DVD-дисків.

А.4 Стадії та етапи розробки

Для роботи над дипломною кваліфікаційною роботою бакалавра заплановано виконання наступних етапів:

- аналіз проблеми;
- розробка архітектури програми;
- розробка програмного забезпечення
- експлуатація, та тестування та експериментальне дослідження програми;
- оформлення документації.

А.5 Порядок контролю й приймання

Дипломна кваліфікаційна робота бакалавра повинна бути виконана у відповідності з календарним планом. Дипломна кваліфікаційна робота допускається до захисту керівником, нормоконтролером, затверджується завідувачем кафедри і захищається в присутності членів екзаменаційної комісії.

ДОДАТОК Б
Текст програми

Б.1 Файл LoginPage.cs

```
using System.Windows;
using System.Windows.Controls;
using System.Windows.Media;
using Nutritionist.SQLiteCode;

namespace Nutritionist.MVVM.View{
    public partial class LoginPage : UserControl{
        public LoginPage(){
            InitializeComponent();
        }

        private void Button_Authentication(object sender, RoutedEventArgs e)
        {
            string login = LoginText.Text.Trim();
            string Password = PasswordText.Password.Trim();
            int Status = 0;

            if (login.Length < 5)
            {
                LoginText.ToolTip = "Це поле ведено не коретно";
            }
            else
            {
                LoginText.ToolTip = "";
                LoginText.Background = Brushes.Transparent;
                Status += 1;
            }
            if(Password.Length < 8)
            {
                PasswordText.ToolTip = "Це поле ведено не коретно";
            }
            else
```

```

{
    PasswordText.ToolTip = "";
    PasswordText.Background = Brushes.Transparent;
    Status += 1;
}
if (Status == 2)
{
    LoginText.ToolTip = "";
    LoginText.Background = Brushes.Transparent;

    PasswordText.ToolTip = "";
    PasswordText.Background = Brushes.Transparent;

    Authentication auth = new Authentication(login, Password);
    if(auth.status == true)
    {
        Main NewWindows = new Main();
        Application.Current.MainWindow.Close();
        NewWindows.ShowDialog();
    }
    else
    {
        MessageBox.Show("Такого користувача не існує");
    }
}
}
}
}

```

Б.2 Файл RegistrationPage.cs

```

using System.Windows;
using System.Windows.Controls;
using System.Windows.Media;

```

```
using Nutritionist.SQLite;
```

```
namespace Nutritionist.MVVM.View
```

```
{
```

```
    public partial class RegistrationPage : UserControl
```

```
    {
```

```
        public RegistrationPage()
```

```
        {
```

```
            InitializeComponent();
```

```
        }
```

```
        private void Registration(object sender, RoutedEventArgs e)
```

```
        {
```

```
            string login = LoginText.Text.Trim();
```

```
            string Password = PasswordText.Password.Trim();
```

```
            string Email = EmailText.Text.Trim();
```

```
            int Status = 0;
```

```
            if (login.Length < 5)
```

```
            {
```

```
                LoginText.ToolTip = "Це поле ведено не коретно";
```

```
                MessageBox.Show("Поле логін ведено не коретно кількість символів повина бути  
більш ніж п'яти символів");
```

```
            }
```

```
            else
```

```
            {
```

```
                Status += 1;
```

```
                LoginText.ToolTip = "";
```

```
                LoginText.Background = Brushes.Transparent;
```

```
            }
```

```
            if (Email.Length < 5 || !Email.Contains('@') || !Email.Contains('.'))
```

```
            {
```

```
                EmailText.ToolTip = "Це поле ведено не коретно";
```

```

        MessageBox.Show("Поле логін ведено не коретно кількість символів повина бути
білльш ніж пяти символів");
    }
    else
    {
        Status += 1;
        EmailText.ToolTip = "";
    }
    if (Password.Length < 8)
    {
        PasswordText.ToolTip = "Це поле ведено не коретно";
        MessageBox.Show("Поле логін ведено не коретно кількість символів повина бути
білльш ніж пяти символів");
    }
    else
    {
        Status += 1;
        PasswordText.ToolTip = "";
    }
    if (Status == 3)
    {
        LoginText.ToolTip = "";
        LoginText.Background = Brushes.Transparent;

        PasswordText.ToolTip = "";
        PasswordText.Background = Brushes.Transparent;

        EmailText.ToolTip = "";
        EmailText.Background = Brushes.Transparent;

        MessageBox.Show("Все окей");

        new MySQL(login, Password, Email);
    }
}

```

```

    }
}

```

Б.3 Файл Product.cs

```

using System.Collections.Generic;
using System.Data.SQLite;

namespace Nutritionist.BDproduct
{
    class Product
    {
        private SQLiteConnection sql_con;
        public List<string> NameProduct = new List<string>();
        public Product()
        {
            string command_load = "select * from Products";

            sql_con = new SQLiteConnection
                ("Data Source= ProductBD.db;"
                + "Version=3;" +
                "New=False;" +
                "Compress=True;"
                );

            sql_con.Open();
            SQLiteCommand command = new SQLiteCommand(command_load, sql_con);
            SQLiteDataReader reader = command.ExecuteReader();

            while (reader.Read())
            {
                NameProduct.Add(reader.GetString(1).ToString());
            }
            reader.Close();

```

```

        sql_con.Close();
    }
}

```

Б.4 Файл Authentication.cs

```

using System.Collections.Generic;
using System.IO;
using System.Data.SQLite;

namespace Nutritionist.SQLiteCode
{
    class Authentication
    {
        public bool status = false;
        private string login;
        private string password;

        private SQLiteConnection sql_con;

        public Authentication(string login, string password)
        {
            this.login = login;
            this.password = password;
            SetDate();
        }

        private void SetDate()
        {
            string command_load = "select * from users";

            List<int> id = new List<int> { };

```

```

List<string> listName = new List<string> { };
List<string> listPassword = new List<string> { };

SetConnection();
sql_con.Open();

SQLiteCommand command = new SQLiteCommand(command_load, sql_con);
SQLiteDataReader reader = command.ExecuteReader();

while (reader.Read())
{
    listName.Add(reader.GetString(1));
    listPassword.Add(reader.GetString(2));
    id.Add(reader.GetInt32(0));
}

for (int i = 0; i < listName.Count; i++)
{
    if(listName[i].Equals(this.login) == true)
    {
        if(listPassword[i].Equals(this.password) == true)
        {
            status = true;
            File.WriteAllText("Sessia.txt", id[i].ToString());
        }
    }
}

reader.Close();
sql_con.Close();
listName.Clear();
listPassword.Clear();
id.Clear();
}

private void SetConnection()

```

```

    {
        sql_con = new SQLiteConnection
            ("Data Source=naturalistBD.db;Version=3;New=False;Compress=True;");
    }
}
}

```

Б.5 Файл MySqlCode.cs

```

using System.IO;
using System.Data.SQLite;
using System.Collections.Generic;

namespace Nutritionist.SQLite
{
    class MySQL
    {
        private SQLiteConnection sql_con;
        private SQLiteCommand sql_cmd;

        private int id;
        private string login;
        private string password;
        private string email;

        public MySQL(string login, string password, string email)
        {
            this.login = login;
            this.password = password;
            this.email = email;

            find_id();
            ExecuteQuery();
        }

        private void find_id()

```

```

{
    string command_load = "select * from users";
    List<string> id_list = new List<string> { };

    SetConnection();
    sql_con.Open();

    SQLiteCommand command = new SQLiteCommand(command_load, sql_con);
    SQLiteDataReader reader = command.ExecuteReader();

    while (reader.Read())
    {
        id_list.Add(reader.GetInt32(0).ToString());
    }

    this.id = int.Parse(id_list[id_list.Count - 1]) + 1;

    reader.Close();
    sql_con.Close();
    id_list.Clear();
}

private void ExecuteQuery()
{
    string command = "insert into users (ID, LOGIN, EMAIL, PASSWORD)values('" + this.id
+ "', '" + this.login + "', '" + this.password + "', '" + this.email + "')";

    SetConnection();
    sql_con.Open();
    sql_cmd = sql_con.CreateCommand();
    sql_con.CreateCommand();
    sql_cmd.CommandText = command;
    sql_cmd.ExecuteNonQuery();
    sql_con.Close();
}

```

```

private void SetConnection()
{
    sql_con = new SQLiteConnection
        ("Data Source= naturalistBD.db;Version=3;New=False;Compress=True;");
}
}
}

```

Б.6 Файл MainViewModel.cs

```

using Nutritionist.Core;

namespace Nutritionist.MVVM.ViewModel
{
    class MainViewModel : ObjectOne
    {
        public ReComands HomeViewComand { get; set; }
        public ReComands RegistrationViewComand { get; set; }

        public HomeViewModel HomeVM { get; set; }
        public RegistrationViewModel RegistrationVM { get; set; }

        private object _currentView;

        public object CurrenView
        {
            get { return _currentView; }
            set
            {
                _currentView = value;
                OnPropertyChanged();
            }
        }
    }
}

```

```

public MainViewModel()
{
    HomeVM = new HomeViewModel();
    RegistrationVM = new RegistrationViewModel();
    CurrenView = HomeVM;
    HomeViewComand = new ReComands(o =>
    {
        CurrenView = HomeVM;

    });
    RegistrationViewComand = new ReComands(o =>
    {
        CurrenView = RegistrationVM;
    });
}
}
}

```

Б.7 Файл ObjectOne.cs

```

using System;
using System.Windows.Input;

namespace Nutritionist.Core
{
    class ReComands : ICommand
    {
        private Action<object> execute_;
        private Func<object, bool> canExecute_;

        public event EventHandler CanExecuteChanged
        {

```

```

        add { CommandManager.RequerySuggested += value; }
        remove { CommandManager.RequerySuggested -= value; }
    }

    public ReComands(Action<object> execute, Func<object, bool> canExecute = null)
    {
        execute_ = execute;
        canExecute_ = canExecute;
    }

    public bool CanExecute(object parameter)
    {
        return canExecute_ == null || canExecute_(parameter);
    }

    public void Execute(object parametr)
    {
        execute_(parametr);
    }
}

```

Б.8 Файл InfoPage.cs

```

using System;
using System.Collections.Generic;
using System.Data.SQLite;
using System.Globalization;
using System.Threading;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Media;
using Nutritionist.BDproduct;
using System.IO;

```

```

namespace Nutritionist.MVVM.View{
    public partial class InfoPage : UserControl
    {
        public InfoPage(){
            InitializeComponent();
        }

        int count = 1;
        List<int> register = new List<int>();
        private SQLiteConnection sql_con;
        private SQLiteCommand sql_cmd;

        private void Button_AddBase(object sender, RoutedEventArgs e){
            List<string> id_list = new List<string> { };
            int id;
            int id_User = int.Parse(File.ReadAllText("Sessia.txt"));
            string command_load = "select * from productUsers";
            sql_con = new SQLiteConnection
                ("Data Source= ProdustUser.db;" +
                "Version=3;" +
                "New=False;" +
                "Compress=True;"
                );

            sql_con.Open();
            SQLiteCommand command = new SQLiteCommand(command_load, sql_con);
            SQLiteDataReader reader = command.ExecuteReader();

            while (reader.Read()){
                id_list.Add(reader.GetInt32(0).ToString());
            }
            if(id_list.Count > 0){
                id = int.Parse(id_list[id_list.Count - 1]) + 1;
            }
            else{

```

```

        id = 0;
    }
    sql_cmd = sql_con.CreateCommand();
    sql_con.CreateCommand();
    sql_cmd.CommandText = AddBase;
    sql_cmd.ExecuteNonQuery();
    sql_con.Close();
}

```

```

private void Button_Update(object sender, RoutedEventArgs e)
{
    CultureInfo temp_culture = Thread.CurrentThread.CurrentCulture;
    Thread.CurrentThread.CurrentCulture = CultureInfo.CreateSpecificCulture("en-US");
    string nameProduct = " ";
    string test = "";
    string command_load = "select * from Products";

    sql_con = new SQLiteConnection
        ("Data Source= ProductBD.db;" +
        "Version=3;" +
        "New=False;" +
        "Compress=True;"
        );

    List<string> name = new List<string>();
    List<string> name_textblock = new List<string> { "VitamunA", "VitamunB1",
    "VitamunB2", "VitamunB4", "VitamunB5", "VitamunB6",
        "VitamunB9", "VitamunC", "VitamunD", "VitamunE", "VitamunH", "VitamunK",
    "VitamunPP", "K", "Ca", "Si", "Mg", "Na",
        "P", "Cl", "Fe", "I", "Co", "Mn", "Cu", "Mo", "Se", "F", "Cr", "Zn", };

    for(int l = 0; l < name_textblock.Count; l++){
        TextBlock txt = (TextBlock)this.StacPanelStatic.FindName(name_textblock[l]);
        txt.Text = "0 %";
    }
}

```

```

for (int i = 0; i < register.Count; i++){
    sql_con.Open();

    SQLiteCommand command = new SQLiteCommand(command_load, sql_con);
    SQLiteDataReader reader = command.ExecuteReader();

    TextBox tb_one = (TextBox)this.StackPanelOneOne.FindName("TextNumberOne" +
register[i].ToString());

    TextBox tb_two = (TextBox)this.StackPanelOneTwo.FindName("TextNumberTwo" +
register[i].ToString());

    while (reader.Read()){
        if (reader.GetString(1).ToString() == tb_one.Text && nameProduct !=
reader.GetString(1).ToString()){
            for (int j=2; j < 8; j++){
                if (reader.GetString(j).ToString() != "None"){
                    name.Add(reader.GetString(j).ToString());
                }
                else{
                    name.Add("0 мг");
                }
            }
            for(int j=9; j<39; j++){
                if (reader.GetString(j).ToString() != "None"){
                    name.Add(reader.GetString(j).ToString());
                }
                else {
                    name.Add("0 %");
                }
            }
        }

        Kalories.Text = (Convert.ToDouble(Kalories.Text.Split(' ')[0]) +
Convert.ToDouble(name[0].Split(' ')[0]) * (Convert.ToDouble(tb_two.Text.Split(' ')[0]) *
0.01)).ToString() + " ккал";

```

```
Protein.Text = (Convert.ToDouble(Protein.Text.Split(' ')[0]) +
Convert.ToDouble(name[1].Split(' ')[0]) * (Convert.ToDouble(tb_two.Text.Split(' ')[0]) *
0.01)).ToString("N2") + " r";
```

```
Fats.Text = (Convert.ToDouble(Fats.Text.Split(' ')[0]) +
Convert.ToDouble(name[2].Split(' ')[0]) * (Convert.ToDouble(tb_two.Text.Split(' ')[0]) *
0.01)).ToString("N2") + " r";
```

```
Carbon.Text = (Convert.ToDouble(Carbon.Text.Split(' ')[0]) +
Convert.ToDouble(name[3].Split(' ')[0]) * (Convert.ToDouble(tb_two.Text.Split(' ')[0]) *
0.01)).ToString("N2") + " r";
```

```
Volocna.Text = (Convert.ToDouble(Volocna.Text.Split(' ')[0]) +
Convert.ToDouble(name[4].Split(' ')[0]) * (Convert.ToDouble(tb_two.Text.Split(' ')[0]) *
0.01)).ToString("N2") + " r";
```

```
Water.Text = (Convert.ToDouble(Water.Text.Split(' ')[0]) +
Convert.ToDouble(name[5].Split(' ')[0]) * (Convert.ToDouble(tb_two.Text.Split(' ')[0]) *
0.01)).ToString("N2") + " r";
```

```
for(int s = 6; s < name.Count; s++){
    TextBlock txtStatic = (TextBlock)this.StacPanelStatic.FindName(
name_textblock[s - 6]);
    txtStatic.Text = (Convert.ToDouble(
txtStatic.Text.Split(' ')[0]) + Convert.ToDouble(name[s].Split(' ')[0])
* (Convert.ToDouble(tb_two.Text.Split(' ')[0]) * 0.01)).ToString() + " Test";
}
name.Clear();
KaloriesP.Text = ((Convert.ToDouble(Kalories.Text.Split(' ')[0]) / 1968) *
100).ToString("N1") + " %";
ProteinP.Text = ((Convert.ToDouble(Protein.Text.Split(' ')[0]) / 118) *
100).ToString("N1") + " %";
FatsP.Text = ((Convert.ToDouble(Fats.Text.Split(' ')[0]) / 86) * 100).ToString("N1")
+ " %";
CarbonP.Text = ((Convert.ToDouble(Carbon.Text.Split(' ')[0]) / 180) *
100).ToString("N1") + " %";
VolocnaP.Text = ((Convert.ToDouble(Volocna.Text.Split(' ')[0]) / 20) *
100).ToString("N1") + " %";
```

```

        WaterP.Text = ((Convert.ToDouble(Water.Text.Split(' ')[0]) / 3000) *
100).ToString("N1") + " %";

```

```

        nameProduct = tb_one.Text;
    }
}
if (register.Count - 1 == i)
{
    for (int l = 0; l < name_textblock.Count; l++)
    {
        TextBlock txtConvert = (TextBlock)this.StackPanelStatic.FindName(
name_textblock[l]);
        txtConvert.Text = ((Convert.ToDouble(
txtConvert.Text.Split(' ')[0]) / 900) * 100).ToString("N0") + " %";
    }
}
reader.Close();
sql_con.Close();
}
Thread.CurrentThread.CurrentCulture = temp_culture;
}

```

```

private void Button_Add(object sender, RoutedEventArgs e){
    string textName = ProductText.Text;
    string textMassa = CountText.Text;

    if(ProductText.Text.Length > 0 && CountText.Text.Length > 0){
        ProductText.Clear();
        CountText.Clear();
        Border tb_one = (Border)this.StackBack.FindName("Back");
        tb_one.Visibility = Visibility.Collapsed;

        if (register.Count < 13){
            register.Add(count);
            TextBox txtOne = new TextBox();

```

```

txtOne.Name = ("TextNumberOne" + count.ToString());
txtOne.Text = textName;
txtOne.Margin = new Thickness(10, 10, 10, 10);

StackPanelOneOne.Children.Add(txtOne);
StackPanelOneOne.RegisterName(txtOne.Name, txtOne);

TextBox txtTwo = new TextBox();
txtTwo.Name = ("TextNumberTwo" + count.ToString());
txtTwo.Text = textMassa;
txtTwo.Margin = new Thickness(10, 10, 10, 10);
StackPanelOneTwo.Children.Add(txtTwo);
StackPanelOneTwo.RegisterName(txtTwo.Name, txtTwo);
count += 1;
}
else if (register.Count < 26){
    register.Add(count);
    TextBox txtOne = new TextBox();
    txtOne.Name = ("TextNumberOne" + count.ToString());
    txtOne.Text = textName;
    txtOne.Margin = new Thickness(10, 10, 10, 10);
    StackPanelTwoOne.Children.Add(txtOne);
    StackPanelTwoOne.RegisterName(txtOne.Name, txtOne);

    TextBox txtTwo = new TextBox();
    txtTwo.Name = ("TextNumberTwo" + count.ToString());
    txtTwo.Text = textMassa;
    txtTwo.Margin = new Thickness(10, 10, 10, 10);
    StackPanelTwoTwo.Children.Add(txtTwo);
    StackPanelTwoTwo.RegisterName(txtTwo.Name, txtTwo);
    count += 1;
}
else if (register.Count < 39){

```

```

register.Add(count);
TextBox txtOne = new TextBox();
txtOne.Name = ("TextNumberOne" + count.ToString());
txtOne.Text = textName;
txtOne.Margin = new Thickness(10, 10, 10, 10);
StackPanelThreeOne.Children.Add(txtOne);
StackPanelThreeOne.RegisterName(txtOne.Name, txtOne);

TextBox txtTwo = new TextBox();
txtTwo.Name = ("TextNumberTwo" + count.ToString());
txtTwo.Text = textMassa;
txtTwo.Margin = new Thickness(10, 10, 10, 10);
StackPanelThreeTwo.Children.Add(txtTwo);
StackPanelThreeTwo.RegisterName(txtTwo.Name, txtTwo);
count += 1;
}
else{
    MessageBox.Show("Неможливо створити позицію");
}
}
}
private void Button_Remove(object sender, RoutedEventArgs e)
{
    int id = 0;
    bool status = true;
    try{
        id = Convert.ToInt32(NumberText.Text);
    }
    catch{
        status = false;
        MessageBox.Show("Неправильне значення");
    }

    if (NumberText.Text.Length > 0 && status == true) {

```

```

string TextPanelOne = "";
string TextPanelTwo = "";

if (register.Count < 14 && register.Count >= id){
    TextBox tb_one = (TextBox)this.StackPanelOneOne.FindName("TextNumberOne" +
register[id - 1].ToString());
    TextBox tb_two = (TextBox)this.StackPanelOneTwo.FindName("TextNumberTwo" +
register[id - 1].ToString());
    this.StackPanelOneOne.Children.Remove(tb_one);
    this.StackPanelOneTwo.Children.Remove(tb_two);

    for (int i = 0; i < register.Count; i++){
        if (i + 1 == id)
        {
            register.RemoveAt(i);
            break;
        }
    }
}

if (register.Count >= 14 && register.Count <= 26 && register.Count >= id)
{
    TextPanelOne = ((TextBox)this.StackPanelTwoOne.FindName("TextNumberOne" +
register[register.Count - 1].ToString())).Text;
    TextPanelTwo = ((TextBox)this.StackPanelTwoTwo.FindName("TextNumberTwo" +
register[register.Count - 1].ToString())).Text;

    TextBox txtOne = (TextBox)this.StackPanelOneOne.FindName("TextNumberOne" +
register[id - 1].ToString());
    txtOne.Text = TextPanelOne;
    TextBox txtTwo = (TextBox)this.StackPanelOneTwo.FindName("TextNumberTwo" +
register[id - 1].ToString());
    txtTwo.Text = TextPanelTwo;

    TextBox tb_one = (TextBox)this.StackPanelTwoOne.FindName("TextNumberOne" +
register[register.Count - 1].ToString());

```

```

        TextBox tb_two = (TextBox)this.StackPanelTwoTwo.FindName("TextNumberTwo" +
register[register.Count - 1].ToString());
        this.StackPanelTwoOne.Children.Remove(tb_one);
        this.StackPanelTwoTwo.Children.Remove(tb_two);

        register.RemoveAt(register.Count - 1);
    }
    if (register.Count >= 27 && register.Count >= id)
    {
        TextPanelOne = ((TextBox)this.StackPanelThreeOne.FindName("TextNumberOne" +
register[register.Count - 1].ToString())).Text;
        TextPanelTwo = ((TextBox)this.StackPanelThreeTwo.FindName("TextNumberTwo" +
register[register.Count - 1].ToString())).Text;

        TextBox txtOne = (TextBox)this.StackPanelOneOne.FindName("TextNumberOne" +
register[id - 1].ToString());
        txtOne.Text = TextPanelOne;

        TextBox txtTwo = (TextBox)this.StackPanelOneTwo.FindName("TextNumberTwo" +
register[id - 1].ToString());
        txtTwo.Text = TextPanelTwo;

        TextBox tb_one = (TextBox)this.StackPanelThreeOne.FindName("TextNumberOne" +
register[register.Count - 1].ToString());
        TextBox tb_two = (TextBox)this.StackPanelThreeTwo.FindName("TextNumberTwo"
+ register[register.Count - 1].ToString());
        this.StackPanelThreeOne.Children.Remove(tb_one);
        this.StackPanelThreeTwo.Children.Remove(tb_two);

        register.RemoveAt(register.Count - 1);
    }
}
}
}

```

```

public class Data

```

```

{
    static public List<string> GetData()
    {
        List<string> data = new List<string>();

        Product prod = new Product();

        data = prod.NameProduct;

        return data;
    }
}

private void ProductText_KeyUp(object sender, System.Windows.Input.KeyEventArgs e)
{
    bool isFound = false;
    var border = (resultStack.Parent as ScrollViewer).Parent as Border;
    var date = Data.GetData();

    string find = (sender as TextBox).Text;

    if (find.Length == 0)
    {
        resultStack.Children.Clear();
        border.Visibility = Visibility.Collapsed;
    }
    else
    {
        border.Visibility = Visibility.Visible;
    }
    resultStack.Children.Clear();
}

```

```

foreach (var obj in date)
{
    if (obj.ToLower().StartsWith(find.ToLower()))
    {
        AddItems(obj);
        isFound = true;
    }

}

if (!isFound)
{
    TextBlock tb1 = new TextBlock();
    tb1.Text = "He має в бази";
    tb1.FontSize = 14;
    tb1.Foreground = new SolidColorBrush(Colors.Black);
    resultStack.Children.Add(tb1);

}
}

public void AddItems(string obj)
{
    Label block = new Label();
    block.Content = obj;
    block.FontSize = 14;
    block.Height = 30;
    block.Foreground = new SolidColorBrush(Colors.Black);
    block.Background = new SolidColorBrush(Colors.White);
    block.Margin = new Thickness(1, 1, 1, 1);
    block.VerticalContentAlignment = VerticalAlignment.Center;
    block.HorizontalContentAlignment = HorizontalAlignment.Left;

    Border tb_one = (Border)this.StackBack.FindName("Back");

```

```

tb_one.Background = new SolidColorBrush(Colors.Gray);

block.MouseLeftButtonDown += (sender, e) =>
{
    ProductText.Text = (sender as Label).Content.ToString();
};
block.MouseEnter += (sender, e) =>
{
    Label lb = sender as Label;
    lb.Background = new SolidColorBrush(Colors.White);
    lb.Foreground = new SolidColorBrush(Colors.Blue);

};
block.MouseLeave += (sender, e) =>
{
    Label lb = sender as Label;
    lb.Background = new SolidColorBrush(Colors.White);
};
resultStack.Children.Add(block);
}
}
}

```

Б.9 Файл ObjectTwo.cs

```

using System;
using System.ComponentModel;
using System.Runtime.CompilerServices;

namespace Nutritionist.Core
{
    class ObjectTwo : INotifyPropertyChanged
    {
        public event PropertyChangedEventHandler PropertyChanged;
    }
}

```

```

protected void OnPropertyChanged([CallerMemberName] string name = null)
{
    PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(name));
}
}
}

```

Б.10 Файл Recomands.cs

```

using System;
using System.Windows.Input;

namespace Nutritionist.Core
{
    class ReComands : ICommand
    {
        private Action<object> execute_;
        private Func<object, bool> canExecute_;

        public event EventHandler CanExecuteChanged
        {
            add { CommandManager.RequerySuggested += value; }
            remove { CommandManager.RequerySuggested -= value; }
        }

        public ReComands(Action<object> execute, Func<object, bool> canExecute = null)
        {
            execute_ = execute;
            canExecute_ = canExecute;
        }

        public bool CanExecute(object parameter)
        {

```

```

        return canExecute_ == null || canExecute_(parameter);
    }
    public void Execute(object parametr)
    {
        execute_(parametr);
    }
}
}

```

Б.11 Файл DiagnostickPage.cs

```

using System;
using System.Collections.Generic;
using System.Data.SQLite;
using System.Globalization;
using System.IO;
using System.Threading;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Media;

namespace Nutritionist.MVVM.View
{
    public partial class DiagnostickPage : UserControl
    {
        public DiagnostickPage()
        {
            InitializeComponent();
            Metod(1);
        }
        private void Button_One(object sender, RoutedEventArgs e)
        {
            Metod(1);
        }
    }
}

```

```

private void Button_Seven(object sender, RoutedEventArgs e)
{
    Metod(7);
}
private void Button_3Elelen(object sender, RoutedEventArgs e)
{
    Metod(30);
}
private void Metod(int count)
{
    CultureInfo temp_culture = Thread.CurrentThread.CurrentCulture;
    Thread.CurrentThread.CurrentCulture = CultureInfo.CreateSpecificCulture("en-US");
    SQLiteConnection sql_con;
    string command_load = "select * from productUsers";
    sql_con = new SQLiteConnection
        ("Data
Source=C:\\Users\\zenja\\source\\repos\\Nutritionist\\Nutritionist\\BDproduct\\ProdustUser.db;" +
        "Version=3;" +
        "New=False;" +
        "Compress=True;"
        );
    sql_con.Open();

    SQLiteCommand command = new SQLiteCommand(command_load, sql_con);
    SQLiteDataReader reader = command.ExecuteReader();
    string id = File.ReadAllText("Sessia.txt");

    List<Double> procent = new List<Double>();
    for(int j=0; j<36; j++)
    {
        procent.Add(0);
    }
    int number = 0;
    int dostup = 0;
    while (reader.Read())

```

```

{
    if (reader.GetInt32(1) == int.Parse(id))
    {
        number += 1;
    }
}
reader.Close();
reader = command.ExecuteReader();
while (reader.Read())
{
    if (reader.GetInt32(1) == int.Parse(id))
    {
        if (number - count <= dostup)
        {
            for (int i = 0; i < procent.Count; i++)
            {
                procent[i] += Convert.ToDouble(reader.GetString(i + 2).Split(' ')[0]);
            }
        }
        dostup += 1;
    }
}
reader.Close();

for(int t = 0; t < procent.Count; t++)
{
    procent[t] = procent[t] / count;
}

List<string> StackPanelList = new List<string> {
    "One", "Two", "Three", "Fou", "Five", "Six", "Seven", "Eight", "Nine", "Ten", "Eleven",
    "Twelve", "Thirteen",
    "Fourteen", "Fifteen", "Sixteen", "Seventeen", "Eighteen", "Nineteen", "Twenty",
    "TwentyOne", "TwentyTwo",
    "TwentyThree", "TwentyFour", "TwentyFive", "TwentySix", "TwentySeven",
    "TwentyEight", "TwentyNine", "Thirty",

```

```

    "ThirtyOne", "ThirtyTwo", "ThirtyThree", "ThirtyFour", "ThirtyFive", "ThirtySix"
};
for(int k=0; k<procent.Count; k++)
{
    if (procent[k] >= 100)
    {
        Border TextPanelOne = ((Border)this.Graphic.FindName(StackPanelList[k]));
        TextPanelOne.Background = new SolidColorBrush(Colors.Green);
        TextPanelOne.Margin = new Thickness(8, 0, 8, 10);
    }
    else
    {
        Border TextPanelOne = ((Border)this.Graphic.FindName(StackPanelList[k]));
        TextPanelOne.Margin = new Thickness(8, 354 - (354 * (procent[k] * 0.01)), 8, 10);
    }
    if (procent[k] <= 99 && procent[k] >= 73)
    {
        Border TextPanelOne = ((Border)this.Graphic.FindName(StackPanelList[k]));
        TextPanelOne.Background = new SolidColorBrush(Colors.Yellow);
    } else if (procent[k] <= 72 && procent[k] >= 47)
    {
        Border TextPanelOne = ((Border)this.Graphic.FindName(StackPanelList[k]));
        TextPanelOne.Background = new SolidColorBrush(Colors.Orange);
    } else if (procent[k] <= 46 && procent[k] >= 21)
    {
        Border TextPanelOne = ((Border)this.Graphic.FindName(StackPanelList[k]));
        TextPanelOne.Background = new SolidColorBrush(Colors.Red);
    }
    else if (procent[k] <= 20 && procent[k] >= 0.11)
    {
        Border TextPanelOne = ((Border)this.Graphic.FindName(StackPanelList[k]));
        TextPanelOne.Background = new SolidColorBrush(Colors.DarkRed);
    } else if (procent[k] < 0.1)
    {
        Border TextPanelOne = ((Border)this.Graphic.FindName(StackPanelList[k]));

```

```

        TextPanelOne.Margin = new Thickness(8, 354, 8, 10);
    }
}
}
}
}

```

Б.12 Файл DiagnostickPage.cs

```

using Microsoft.Win32;
using Prism.Services.Dialogs;
using System;
using System.Collections.Generic;
using System.Data.SQLite;
using System.Globalization;
using System.IO;
using System.Threading;
using System.Windows;
using System.Windows.Controls;

```

```

namespace Nutritionist.MVVM.View
{

```

```

    public partial class InfoHeal : UserControl
    {

```

```

        private int CountPage;
        private int CountDay = 1;
        private int buttonCount = 1;

```

```

        private List<string> AttributeName = new List<string> { };
        private List<string> AttributeContext = new List<string> { };
        private List<string> AttributeFullName = new List<string> { };

```

```

private SQLiteConnection sql_con;
private SQLiteCommand sql_cmd;

public InfoHeal()
{
    InitializeComponent();
    Start();
}

private void Button_Check(object sender, RoutedEventArgs e)
{
    if (Days.Text.Length > 0)
    {
        try
        {
            CountDay = int.Parse(Days.Text);
            Start();
        }
        catch
        {
            MessageBox.Show("Помилкове значення");
        }
    }
    else
    {
        MessageBox.Show("Введіть кількість днів");
    }
}

private void Button_Tomorrow(object sender, RoutedEventArgs e)
{
    List<string> Panell = new List<string> {
        "One", "Two", "Three", "Fou", "Five", "Six", "Seven", "Eigmt", "Nine", "Ten", "Evelen",
        "Twoneen", "Threeteen", "Foutteen"
    };
};

```

```

if (CountPage > buttonCount)
{
    for (int k = 1; k < Panell.Count; k++)
    {
        TextBlock txt = (TextBlock)this.StackPanelOne.FindName(Panell[k]);
        txt.Visibility = Visibility.Hidden;
    }

    buttonCount += 1;

    TextBlock txtOne = (TextBlock)this.StackPanelOne.FindName("One");
    txtOne.Text = AttributeFullName[buttonCount - 1].ToString();

    for (int g = 1; g < AttributeContext[buttonCount - 1].Split(",").Length; g++)
    {
        if (g <= 13)
        {
            TextBlock txtTwo = (TextBlock)this.StackPanelOne.FindName(Panell[g]);
            txtTwo.Text = AttributeContext[buttonCount - 1].Split(",")[g];
            txtTwo.Visibility = Visibility.Visible;
        }
    }
}

private void Start()
{
    List<Double> date = new List<double>();
    date = ParseDateOne();
    CountPageMethot(date);
}

```

```

private void CountPageMethot(List<Double> date)
{
    AtributeName.Clear();
    AtributeContext.Clear();
    AtributeFullName.Clear();
    buttonCount = 1;

    List<string> nameComponet = new List<string> {
        "Call", "Prote", "Fat", "Carb", "Alim", "Wate",
        "A", "B1", "B2", "B4", "B5", "B6", "B9", "C", "D", "E", "H", "K", "PP", "Kaliy",
        "CA", "SI", "MG", "NA", "P", "CL", "FE", "I", "CO", "MN", "CU", "MO", "SE", "F",
        "CR", "ZN"};

    List<string> Panell = new List<string> {
        "One", "Two", "Three", "Fou", "Five", "Six", "Seven", "Eigmt", "Nine", "Ten", "Evelen",
        "Twoneen", "Threeteen", "Fouteen"
    };

    SQLiteConnection sql_con;

    string command_load = "select * from components";

    sql_con = new SQLiteConnection
        ("Data
Source=C:\\Users\\zenja\\source\\repos\\Nutritionist\\Nutritionist\\BDproduct\\InfoComponents.db;
" +
        "Version=3;" +
        "New=False;" +
        "Compress=True;"
    );

    sql_con.Open();

    SQLiteCommand command = new SQLiteCommand(command_load, sql_con);

```

```
SQLiteDataReader reader = command.ExecuteReader();
```

```
List<string> AttributeOne = new List<string> { };
```

```
List<string> AttributeTwo = new List<string> { };
```

```
List<string> AttributeThree = new List<string> { };
```

```
while (reader.Read())
```

```
{
```

```
    AttributeOne.Add(reader.GetString(0));
```

```
    AttributeTwo.Add(reader.GetString(1));
```

```
    AttributeThree.Add(reader.GetString(3));
```

```
}
```

```
int id = 0;
```

```
for (int i = 6; i < date.Count; i++)
```

```
{
```

```
    if (date[i] < 50)
```

```
    {
```

```
        for (int j = 0; j < AttributeOne.Count; j++)
```

```
        {
```

```
            if (nameComponet[i].ToString() == AttributeOne[j].ToString())
```

```
            {
```

```
                AttributeName.Add(" ");
```

```
                AttributeName[AttributeName.Count - 1] = AttributeOne[j];
```

```
                AttributeContext.Add(" ");
```

```
                AttributeContext[AttributeContext.Count - 1] = AttributeTwo[j];
```

```
                AttributeFullName.Add(" ");
```

```
                AttributeFullName[AttributeFullName.Count - 1] = AttributeThree[j];
```

```
            id += 1;
```

```
            if (id == 1)
```

```
            {
```

```

        TextBlock txtOne = (TextBlock)this.StackPanelOne.FindName("One");
        txtOne.Text = AttributeThree[j].ToString();

        for (int g = 1; g < AttributeTwo[j].Split(",").Length; g++)
        {
            if (g <= 13)
            {
                TextBlock txtTwo =
(TextBlock)this.StackPanelOne.FindName(Panell[g]);

                txtTwo.Text = AttributeTwo[j].Split(",")[g];
                txtTwo.Visibility = Visibility.Visible;
            }
        }
    }
}

CountPage = id;
reader.Close();
AttributeOne.Clear();
AttributeTwo.Clear();
AttributeThree.Clear();
}

```

```

("Data Source= ProdstUser.db;" +
"Version=3;" +
"New=False;" +
"Compress=True;"
);

sql_con.Open();

SQLiteCommand command = new SQLiteCommand(command_load, sql_con);
SQLiteDataReader reader = command.ExecuteReader();

string id = File.ReadAllText("Sessia.txt");

List<Double> procent = new List<Double>();
List<string> nameComponent = new List<string>();

for (int j = 0; j < 36; j++)
{
    procent.Add(0);
    nameComponent.Add(" ");
}

int number = 0;
int dostup = 0;

while (reader.Read())
{
    if (reader.GetInt32(1) == int.Parse(id))
    {
        number += 1;
    }
}

reader.Close();
reader = command.ExecuteReader();

while (reader.Read())

```

```

{
    if (reader.GetInt32(1) == int.Parse(id))
    {
        if (number - CountDay <= dostup)
        {
            for (int i = 0; i < procent.Count; i++)
            {
                procent[i] += Convert.ToDouble(reader.GetString(i + 2).Split(' ')[0]);
            }
        }
        dostup += 1;
    }
}
reader.Close();

for (int t = 0; t < procent.Count; t++)
{
    procent[t] = procent[t] / CountDay;
}
return procent;
}

private void Button_Download(object sender, RoutedEventArgs e)
{
    List<Double> date = new List<double>();
    List<String> dateString = new List<string>();
    bool status = true;

    try
    { CountDay = int.Parse(DownloadDays.Text); }
    catch
    {
        status = false;
        MessageBox.Show("Помилкове значення");
    }
}

```

```

if (DownloadDays.Text.Length > 0 && status == true)
{
    date = ParseDateOne();

    for (int i = 0; i < date.Count; i++)
    {
        dateString.Add(date[i].ToString());
    }

    SaveFileDialog saveFileDialog = new SaveFileDialog();
    saveFileDialog.Filter = "Text file (*.txt)|*.txt";

    if (saveFileDialog.ShowDialog() == true)
    {
        using (StreamWriter sw = new StreamWriter(saveFileDialog.FileName, true,
System.Text.Encoding.Default))
        {
            for (int j = 0; j < date.Count; j++)
            {
                sw.WriteLine(date[j].ToString());
            }
            sw.Close();
        }
    }
}

private void Button_Input(object sender, RoutedEventArgs e)
{
    List<Double> date = new List<double>();
    OpenFileDialog openFileDialog = new OpenFileDialog();
    openFileDialog.Filter = "Text file (*.txt)|*.txt";

```

```
if (openFileDialog.ShowDialog() == true)
{
    using (StreamReader sr = new StreamReader(openFileDialog.FileName,
System.Text.Encoding.Default))
    {
        string line;
        while ((line = sr.ReadLine()) != null)
        {
            date.Add(Convert.ToDouble(line));
        }
    }
    CountPageMethot(date);
}
}
```

ДОДАТОК В
Слайди презентації

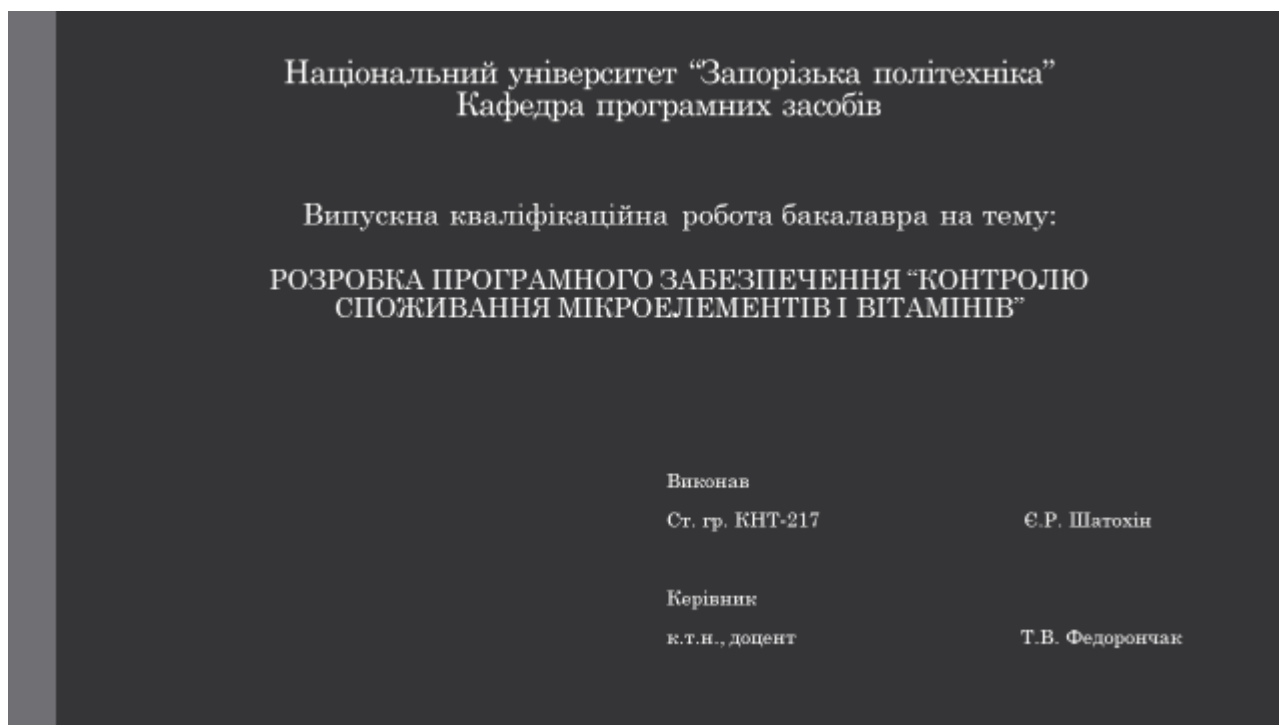


Рисунок В.1 – Слайд 1

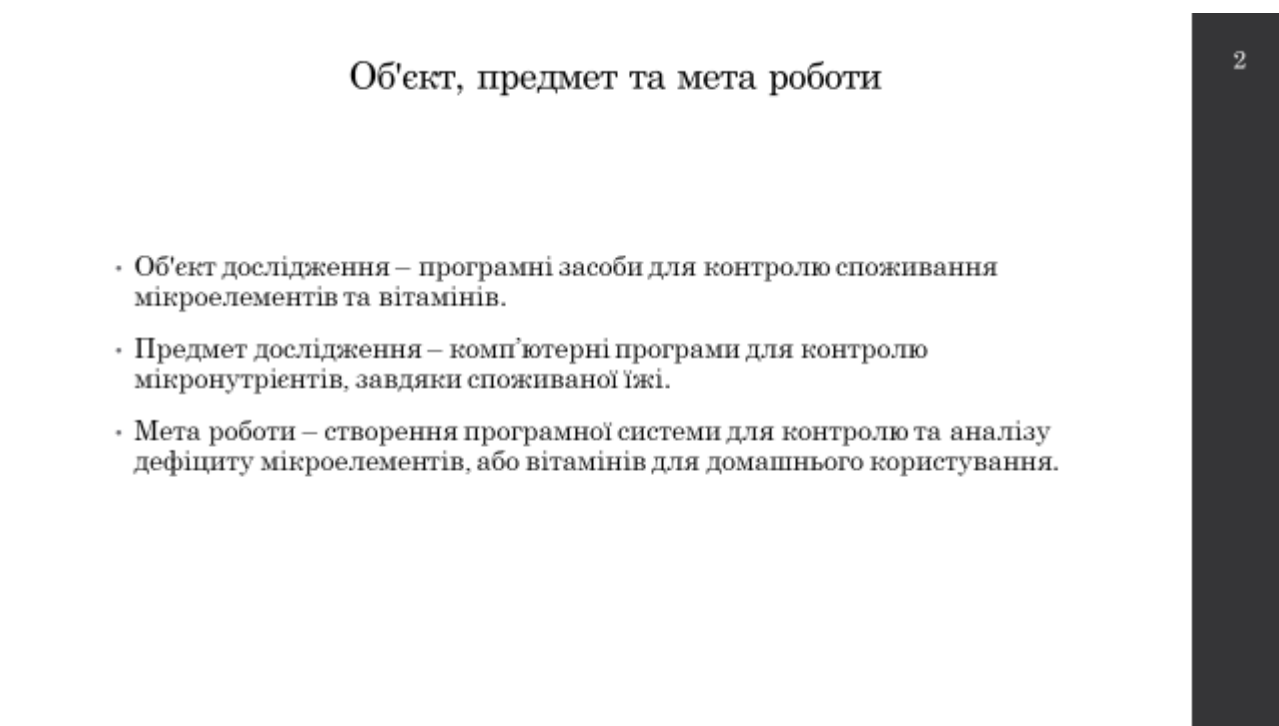


Рисунок В.2 – Слайд 2

Аналоги продуктів в мережі



– «Щоденник харчування»



– «Fit Audit»



– «Cronometer»

Порівняльна таблиця

Характеристика	«Щоденник харчування»	«Cronometer»	«Fit Audit»
Зрозумілий дизайн	+/-	+/-	+
Зрозумілий функціонал	+	-	+
Динамічний перегляд змін	+	+	+
Присутність алгоритмів аналізу	+	+	-
Фільтри	+	+	+/-
Особистий кабінет	+	+	-
Пошук	+	+	+

Рисунок В.3 – Слайд 3

Функціональні можливості розроблюваної системи

- Реєстрація користувача в системі
- Авторизація користувача в системі
- Занесення продуктів для аналізу
- Аналіз продуктів на склад вітамінів та мінералів
- Будування графіку
- Будування стислого списку можливих захворювань
- Експорт та імпорт стислого списку

Рисунок В.4 – Слайд 4

Атрибутна структура таблиць

```

1 CREATE TABLE "Products" (
2   "ID" INTEGER NOT NULL UNIQUE,
3   "NAME" TEXT NOT NULL,
4   "CALORIES" TEXT,
5   "PROTEIN" TEXT,
6   "FATS" TEXT,
7   "CARBOHYDRATES" TEXT,
8   "ALIMENTARY" TEXT,
9   "WATER" TEXT,
10  "ZOLA" TEXT,
11  "A" TEXT,
12  "B1" TEXT,
13  "B2" TEXT,
14  "BOLIN" TEXT,
15  "B5" TEXT,
16  "B6" TEXT,
17  "B9" TEXT,
18  "C" TEXT,
19  "D" TEXT,
20  "E" TEXT,
21  "H" TEXT,
22  "FF" TEXT,
23  "GALIV" TEXT,
24  "CA" TEXT,
25  "SI" TEXT,
26  "HG" TEXT,
27  "HA" TEXT,
28  "P" TEXT,
29  "CL" TEXT,
30  "FE" TEXT,
31  "I" TEXT,
32  "CO" TEXT,
33  "HS" TEXT,
34  "CU" TEXT,
35  "MO" TEXT,
36  "SE" TEXT,
37  "X" TEXT,
38  "CR" TEXT,
39  "ZB" TEXT,
40  PRIMARY KEY ("ID" AUTOINCREMENT)
41 )
42

```

– Атрибутна структура таблиці
"Products"

```

1 CREATE TABLE "Users" (
2   "Id" INTEGER NOT NULL UNIQUE,
3   "login" TEXT NOT NULL UNIQUE,
4   "email" TEXT NOT NULL,
5   "password" TEXT NOT NULL,
6   PRIMARY KEY("Id" AUTOINCREMENT)
7 )

```

Атрибутна структура таблиці
"Users"

```

1 CREATE TABLE "components" (
2   "NAME" TEXT NOT NULL UNIQUE,
3   "HYPERVITAMINOSIS" TEXT,
4   "HYPOVITAMINOSIS" TEXT,
5   "FULL" TEXT,
6   PRIMARY KEY("NAME")
7 )

```

Атрибутна структура таблиці "components"

Рисунок В.7 – Слайд 7

Аналіз мов програмування

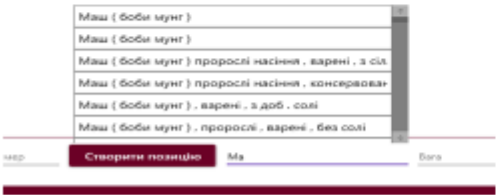
Характеристика	C#	Java	Python
Рік створення	1998	1995	1991
Тип виконання	напівкомпільовані (компілюється C # у CIL)	напівкомпільована	інтерпретований, компільований в байт-код Java
Система типизації	статична, динамічна та явна типизація	статична типизація та явна типизація	динамічна типизація та неявна типизація
Підтримка ООП	+	+	+
Стандарти	ISO ECMA	–	PEP 8
Парадигми програмування:	Мульти- парадигмальний	Мульти- парадигмальний	Мульти- парадигмальний
Управління пам'яттю	Створення об'єктів на стеку, векеровані показники, ручне управління та прибирання сміття	Прибирання сміття	Прибирання сміття

Рисунок В.8 – Слайд 8

Основний вигляд застосунку



Інтерфейс – авторизація



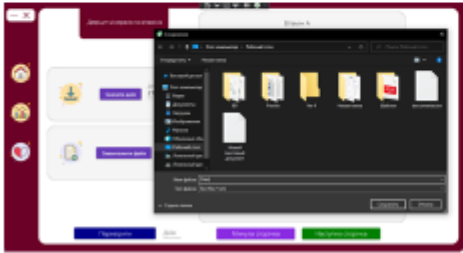
Інтерфейс – введення продуктів для аналізу

Рисунок В.9 – Слайд 9

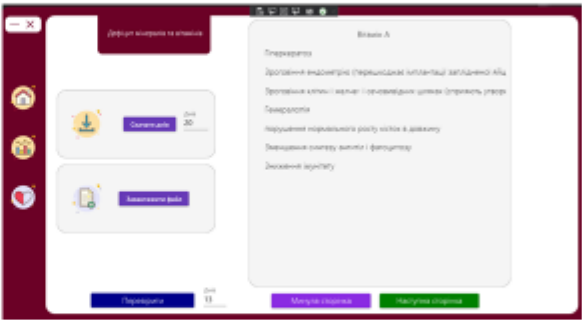
Вигляд застосунку для аналізу



Інтерфейс – формування графіку



Експорт стислого списку



Інтерфейс – формування стислого списку
можливої хвороби

Рисунок В.10 – Слайд 10

Тестування застосунку в критичних середовищах

11

Системні характеристики	Windows 7	Windows 8.1	Windows 10
Оперативна пам'ять 256 мб Процесор 1 ядра	2.4 секунд	3.0 секунд	3.1 секунд
Оперативна пам'ять 512 мб Процесор 1 ядра	2 секунд	2.5 секунд	2.4 секунд
Оперативна пам'ять 256 мб Процесор 2 ядра	2.3 секунд	2.8 секунд	2.7 секунд
Оперативна пам'ять 512 мб Процесор 2 ядра	1.9 секунд	2.3 секунд	2.0 секунд

Тест процесу авторизації

Рисунок В.11 – Слайд 11

Висновок

12

- Під час виконання бакалаврської роботи було розроблено програму з графічним інтерфейсом "Nutritionist".
- Під час дослідження було виявлено три проблеми: мала кількість людей, які контролюють свій рівень нутрієнтів, важко згрупувати інформацію цілком, наприклад щоб, це зробити потрібно шукати інформацію на різних ресурсах та остання проблема це те, що аналоги ідеї існують в загалом у web-технологіях та на мобільні програми.
- Для розв'язання проблем було створено програму, вона поєднала розв'язання трьох проблем, а саме, групування інформації в одному місці, створення програми на Windows та завдяки розв'язанню двох проблем зацікавлених користувачів стане більше.
- Програму "Nutritionist" можна використовувати в медичних закладах, а саме, для лікарів дієтологів, педіатрів та сімейних лікарів. Результат цієї програми це контроль та можливість заздалегідь уникнути можливої хвороби.
- Проаналізовано сучасні технології у створенні програмного забезпечення. Найпопулярнішою тенденцією останніх років є мінімалістичний дизайн, який виключає нагромадження об'ємних об'єктів.

Рисунок В.12 – Слайд 12