

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій
(повне найменування факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему РОЗРОБКА ТА ДОСЛІДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
ОСОБИСТОГО ПОМІЧНИКА ДЛЯ ФОРМУВАННЯ ЩОДЕННОГО
РАЦІОНУ ХАРЧУВАННЯ
DEVELOPMENT AND RESEARCH OF SOFTWARE FOR A PERSONAL
ASSISTANT IN DAILY MEAL PLANNING

Виконав(ла): студент(ка) 2 курсу, групи КНТ-124м
Спеціальності 121 Інженерія програмного
Забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

МІХАЙЛОВА М.С.

(ПРИЗВИЩЕ та ініціали)

Керівник СТЕПАНЕНКО О.О.

(ПРИЗВИЩЕ та ініціали)

Рецензент ДЬЯЧУК Т.С.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ

Кафедра програмних засобів

Ступінь вищої освіти магістр

Спеціальність 121 Інженерія програмного забезпечення

(код і найменування)

Освітня програма (спеціалізація) Інженерія програмного забезпечення

(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.

Сергій СУББОТІН

“ ” 2025 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)
МІХАЙЛОВОЇ Марії Сергіївни

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка та дослідження програмного забезпечення особистого помічника для формування щоденного раціону харчування.

Development and Research of Software for a Personal Assistant in Daily Meal Planning

керівник проєкту (роботи) к.т.н., доцент, СТЕПАНЕНКО Олександр
Олексійович

(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 30 ” вересня 2025 року № 447

2. Строк подання студентом проєкту (роботи) 08 грудня 2025 року

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

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Реферат. Перелік скорочень та умовних познач. Вступ. 1. Аналіз предметної області. 2. Вибір та обґрунтування структури системи, яка проєктується. 3. Основні рішення щодо реалізації компонентів системи. 4. Керівництво програміста. 5. Керівництво оператора. 6. Експерименти та результати. Висновки. Перелік джерел посилань. Додатки.

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

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-6 Основна частина	СТЕПАНЕНКО О.О., доцент		
Нормоконтролер	ДЕЙНЕГА Л.Ю., ст. викладач		

7. Дата видачі завдання “30” вересня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	2-3 тижні	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	4-5 тижні	Розділ 2
4	Розробка архітектури програми.	6 тиждень	Розділ 3
5	Розробка програми.	7-8 тижні	Розділ 3
6	Тестування та експериментальне дослідження програмного забезпечення.	9 тиждень	Розділ 4,5,6
7	Оформлення пояснювальної записки та документів до неї.	10-11 тижні	Додатки
8	Нормоконтроль та рецензування.	12 тиждень	
9	Захист роботи.	12 тиждень	

Студент(ка)

_____ Марія МІХАЙЛОВА
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи)

_____ Олександр СТЕПАНЕНКО
(підпис) (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
206 с., 33 табл., 48 рис., 3 дод., 61 джерел.

РАЦІОН, ПЕРСОНАЛІЗАЦІЯ, ГЕНЕТИЧНИЙ АЛГОРИТМ,
НЕЙРОННА МЕРЕЖА, PYTHON, C#, VISUAL STUDIO 2019, WPF, MLP.

Об'єкт дослідження – процеси програмної інженерії, що пов'язані з розробленням, проєктуванням, забезпеченням якості та експлуатацією програмного забезпечення для автоматизації формування персоналізованих планів харчування, що враховують індивідуальні параметри користувачів.

Предмет дослідження – методи, алгоритми та технології створення програмного забезпечення для формування персоналізованого раціону, зокрема застосування генетичних алгоритмів та багатошарових нейронних мереж (MLP) для оптимізації та прогнозування у задачах персоналізації харчування з урахуванням таких параметрів, як вік, стать, рівень фізичної активності, харчові вподобання та обмеження.

Мета роботи – створення програмної системи, яка автоматизує процес формування збалансованих індивідуальних планів харчування шляхом поєднання методів штучного інтелекту та сучасних технологій розроблення програмного забезпечення для оптимізації добової калорійності відповідно до параметрів користувача.

Наукова новизна полягає у створенні модифікованого генетичного алгоритму (ГА) для оптимізації та модифікованої багатошарової нейронної мережі (MLP) для прогнозування у задачі формування персоналізованого раціону харчування.

Матеріали, методи та технічні засоби: програмування здійснювалося мовами C# та Python, із використанням середовища Microsoft Visual Studio

2019, графічної підсистеми WPF, бази даних Microsoft SQL Server, файлів форматів JSON і CSV.

Результати. Створено програмний продукт, який забезпечує користувачам інтуїтивно зрозумілий інтерфейс для внесення особистих даних, автоматичного створення індивідуальних раціонів харчування та збереження інформації. У застосунку інтегровані передові алгоритми та технології обробки даних. Модифікований ГА дозволив отримати більше різноманітних варіантів рішень при збереженні стабільної збіжності, а модифікована MLP – швидше навчання та кращу адаптацію до специфіки даних користувачів.

Висновки. Створений застосунок «EatMe» вирізняється простотою у використанні, гнучкістю налаштувань, високою функціональністю та достатнім рівнем персоналізації, що робить його зручним як для професійного використання, так і для особистих потреб.

Сфера застосування – організація здорового способу життя, покращення харчових звичок у професійному середовищі та для індивідуального користування.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 206 pages, 33 tables, 48 figures, 3 appendixes, 61 sources.

RATION, PERSONALIZATION, GENETIC ALGORITHM, NEURAL NETWORK, PYTHON, C#, VISUAL STUDIO 2019, WPF, MLP.

Research Object – the processes of software engineering related to the development, design, quality assurance, and operation of software intended to automate the creation of personalized meal plans that take into account users individual parameters.

Subject of Research – the methods, algorithms, and technologies for developing software that generates personalized nutrition plans, in particular the use of genetic algorithms and multilayer perceptron neural networks (MLP) for optimization and prediction in nutrition-personalization tasks, considering such parameters as age, sex, physical activity level, dietary preferences, and restrictions.

The aim of the work – to develop a software system that automates the creation of balanced individual meal plans by combining artificial intelligence methods with modern software development technologies to optimize daily caloric intake according to the user's parameters.

The scientific novelty lies in the development of a modified Genetic Algorithm (GA) for optimization and a modified Multilayer Perceptron (MLP) neural network for prediction in the task of personalized meal plan generation.

Materials, Methods, and Technical Means: programming was carried out using C# and Python, utilizing the Microsoft Visual Studio 2019 environment, WPF graphical subsystem, Microsoft SQL Server database, and JSON and CSV file formats.

The results. A software product has been created that provides users with an intuitive interface for entering personal data, automatically generating

individualized meal plans, and storing information. Advanced algorithms and data processing technologies have been integrated into the application. The modified GA made it possible to obtain a greater diversity of solutions while maintaining stable convergence, whereas the modified MLP provided faster training and better adaptation to the specific characteristics of user data.

Conclusions. The «EatMe» application stands out for its ease of use, flexible settings, high functionality, and sufficient level of personalization, making it convenient for both professional use and personal needs.

Application Area. Promoting a healthy lifestyle, improving eating habits in a professional environment, and for individual use.

ЗМІСТ

	С.
Перелік умовних позначень, символів, одиниць, скорочень і термінів	12
Вступ.....	13
1 Аналіз предметної області.....	14
1.1 Огляд існуючих методів вирішення завдання	18
1.2 Огляд існуючих програмних засобів, що вирішують аналогічні завдання	21
1.2.1 Мобільний застосунок «Samsung Food»	22
1.2.2 Мобільний застосунок «MyFitnessPal»	26
1.2.3 Сервіс «Eat This Much»	29
1.2.4 Порівняльна таблиця існуючих та розроблюваного програмних продуктів.....	31
1.3 Порівняння алгоритмів для персонального планування раціону харчування	33
1.4 Аналіз підходів до оптимізації персоналізованого раціону харчування	39
1.5 Опис обраних формул для розрахунків індивідуальних показників.....	45
1.6 Постановка завдання роботи.....	49
1.7 Висновки за розділом 1	50
2 Вибір та обґрунтування структури системи, яка проектується.....	52
2.1 Вибір інструментарію розробки програмного забезпечення	52
2.1.1 Огляд особливостей мови програмування	52
2.1.2 Огляд особливостей обраного середовища розробки	62
2.1.3 Огляд особливостей обраної системи управління базами даних	67
2.1.4 Вибір бібліотеки для машинного навчання.....	70
2.2 Структурна схема програми	72
2.3 Опис модулів розроблюваної системи.....	74
2.4 Діаграма прецедентів.....	76
2.5 Висновки за розділом 2	77
3 Основні рішення щодо реалізації компонентів системи.....	78

	9
3.1 Алгоритм функціонування розробленої програми.....	78
3.1.1 Алгоритм генерації варіантів раціонів на день.....	79
3.1.2 Генетичний алгоритм	80
3.1.3 Нейромережевий алгоритм	85
3.1.4 Алгоритм зміни даних користувача.....	88
3.1.5 Алгоритм додавання алергій	88
3.1.6 Алгоритм додавання вподобань	89
3.1.7 Алгоритм вибору раціону	90
3.1.8 Алгоритм відхилення раціону	90
3.1.9 Алгоритм зчитування страв з бази даних у файли	91
3.2 Опис архітектури створеної моделі нейронної мережі.....	92
3.3 Опис модулів розроблюваної системи.....	96
3.3.1 Клас AboutWindow.....	99
3.3.2 Клас AllergyAndPreferencesWindow.....	99
3.3.3 Клас DailyMealPlanWindow	100
3.3.4 Клас InstructionsWindow.....	102
3.3.5 Клас MainWindow	102
3.3.6 Клас MessageWindow	103
3.3.7 Клас ProfileWindow.....	104
3.3.8 Клас DataManager.....	105
3.3.9 Клас Dish.....	106
3.3.10 Клас MealPlan	107
3.3.11 Клас User.....	108
3.3.12 Клас GeneticAlgorithm	110
3.3.13 Клас NeuralNetwork	111
3.3.14 Клас DataManager.....	112
3.3.15 Клас Logger.....	113
3.3.16 Клас Dish.....	114
3.3.17 Клас MealPlan	115
3.3.18 Клас User.....	115

	10
3.4 Проектування бази даних.....	116
3.5 Основні рішення щодо розробки інтерфейсу.....	120
4 Керівництво програміста.....	136
4.1 Призначення та умови застосування програми	136
4.1.1 Призначення програми.....	136
4.1.2 Функції програми.....	137
4.1.3 Умови застосування програми	138
4.2 Характеристика програми	138
4.2.1 Структура програми	139
4.3 Звертання до програми	140
4.4 Вхідні та вихідні дані.....	140
4.4.1 Вхідні дані	140
4.4.2 Вихідні дані	140
4.5 Повідомлення	141
5 Керівництво оператора	143
5.1 Призначення програми	143
5.2 Умови виконання програми	143
5.2.1 Апаратні вимоги програми	143
5.2.2 Вимоги до операторів.....	144
5.3 Виконання програми.....	144
5.3.1 Запуск програми.....	144
5.3.2 Виконання роботи з програмою	144
5.4 Повідомлення оператору.....	151
6 Тестування і результати.....	153
6.2 Результати використання розробленого програмного забезпечення ...	154
6.2.1 Налаштування профілю користувача	154
6.2.2 Генерація раціону на день.....	155
6.2.3 Головне вікно	155
6.3 Результати тестування генетичного й нейромережевого алгоритмів ..	156
Висновки	159

	11
Перелік джерел посилання	162
Додаток А Технічне завдання	168
Додаток Б Текст програми	177
Додаток В Слайди презентації.....	195

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД – бази даних;

ГА – Генетичний алгоритм

ООП – об’єктно-орієнтоване програмування;

ОС – операційна система;

ПЗ – програмне забезпечення;

ПК – персональний комп’ютер

СКБД – системи керування базами даних;

ТЗ – технічне завдання;

ШІ – штучний інтелект;

BMI – Body Mass Index;

BMR – Basal Metabolic Rate;

DNN – Dense Neural Network;

MLP – Multi-Layer Perceptron;

WPF – Windows Presentation Foundation;

UX – User Experience.

ВСТУП

У сучасному світі програмні застосунки відіграють важливу роль у повсякденному житті людей. Вони використовуються для планування часу, контролю за здоров'ям, покупок та багатьох інших завдань. З розвитком технологій попит на інструменти, що спрощують рутину та допомагають приймати обґрунтовані рішення, зростає.

Раціон харчування є важливим аспектом здорового способу життя. Правильно складений раціон впливає на фізичне та психічне здоров'я, продуктивність та якість життя. Однак складання збалансованого меню часто є викликом через необхідність врахування калорійності, дієтичних обмежень, алергій та інших параметрів. Саме тому створення програмного помічника для складання щоденного раціону є зручним і актуальним рішенням.

Актуальність обумовлена зростанням інтересу до здорового харчування та необхідністю автоматизації процесу планування раціону. Такий інструмент не лише допоможе зекономити час, але й сприятиме формуванню корисних звичок у користувачів. Використання штучного інтелекту в цьому напрямку дозволяє зробити рекомендації максимально персоналізованими.

Завданням дипломної роботи є розробка програмного застосунку «EatMe», який автоматизує процес створення щоденного раціону харчування. Застосунок дозволяє користувачам отримувати персоналізовані рекомендації, враховувати дієтичні обмеження, розраховувати калорійність страв та створювати оптимальний раціон харчування на основі заданих критеріїв.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Раціональне харчування є одним із ключових аспектів здорового способу життя. Складання щоденного раціону вимагає врахування багатьох чинників, зокрема калорійності, балансу білків, жирів і вуглеводів, дієтичних обмежень та особистих уподобань. У цьому розділі буде розглянуто основні аспекти предметної області, які стосуються здорового харчування, автоматизації процесу складання раціону та використання сучасних технологій у цій сфері.

Аналіз предметної області включає визначення ключових термінів і понять, що використовуються в контексті розробки застосунку для створення щоденного раціону. Ці визначення є важливими для правильного розуміння завдань, які вирішує застосунок.

Існує кілька основних типів харчування, які класифікуються за їх біологічною дією на організм, кожен з яких має своє призначення і вплив.

Раціональне харчування – це фізіологічно обґрунтований спосіб забезпечення організму здорової людини необхідною кількістю поживних речовин і енергії. Воно враховує норми фізіологічних потреб залежно від віку, статі, рівня фізичної активності, стану здоров'я та інших індивідуальних особливостей, включаючи перебіг захворювань і супутні фактори [1].

Превентивне харчування – це вид харчування, який спрямований на попередження захворювань, які можуть виникнути в результаті багатьох факторів, таких як атеросклероз, гіпертонія, діабет чи хвороби серця. Превентивне харчування є вдосконаленим варіантом раціонального, що враховує фактори ризику для здоров'я і покликане зменшити ймовірність розвитку захворювань [2].

Лікувально-профілактичне харчування – це харчування, яке дуже близьке до раціонального, але з акцентом на підвищення захисних властивостей їжі для запобігання впливу шкідливих факторів на організм. Воно включає певні зміни в раціоні з метою покращення стійкості організму

та попередження розвитку різноманітних захворювань, пов'язаних з впливом негативних факторів.

Дієтичне харчування — цей тип харчування використовується для людей з хронічними захворюваннями з метою запобігання загостренням або розвитку нових ускладнень. Дієтичне харчування включає обмеження або коригування вживання певних продуктів, щоб підтримувати стабільний стан хворих та уникнути прогресування захворювань [2].

Раціон харчування – набір продуктів та страв, які споживає людина протягом дня або іншого періоду, складений для забезпечення організму необхідними поживними речовинами та енергією.

Концепція індивідуального харчування полягає в тому, що кожна людина має унікальні потреби щодо поживних речовин і енергії, залежно від її здоров'я та фізіологічних і генетичних факторів. Хоча існують загальні нормативи харчування, що враховують вікові та статеві відмінності, ці норми є досить загальними і не завжди відповідають індивідуальним потребам конкретної людини. Тому все частіше звертаються до індивідуалізації харчування, що дозволяє врахувати генетичні особливості, рівень фізичної активності, стан здоров'я та інші фактори, що можуть впливати на харчові потреби [2].

У розвинених країнах все більше людей орієнтуються на індивідуальні рекомендації, підбираючи свій раціон відповідно до антропометричних показників, таких як вага, зріст і співвідношення цих параметрів. Це дозволяє не лише підтримувати здорову вагу, але й сприяє профілактиці хронічних захворювань, покращує загальне самопочуття і навіть може впливати на тривалість життя. Індивідуалізоване харчування, що враховує всі ці аспекти, набуває популярності завдяки розвиткові новітніх технологій і досліджень у галузі генетики та харчових наук.

Наукові дослідження свідчать, що для підтримання здоров'я організму людина щоденно потребує близько 600 різних поживних елементів, з яких 66

є життєво необхідними. Крім того, протягом тижня раціон повинен включати щонайменше 30 різноманітних страв [3].

Калорійність – енергетична цінність їжі, що вимірюється у кілокалоріях (ккал) і відображає кількість енергії, яку отримує організм під час споживання продукту [1].

Хоча існують деякі винятки, більшість нездорових продуктів мають високу калорійну щільність, в той час як фрукти та овочі є поживними (або мають низьку калорійну щільність). Наприклад, середній пончик (приблизно 69 грамів) містить 270 калорій. Для порівняння, чотири великі полуниці (69 грамів) містять лише 22 калорії. Тому пончик буде висококалорійним продуктом, а полуниця – поживним. Авокадо є одним з основних винятків, оскільки одне середнє авокадо містить 322 калорії. Попри високу калорійність, авокадо багате на клітковину (13 грамів), калій, фолат, вітаміни B6, E, C, K та мідь [4].

У харчуванні важливими складовими є не лише білки, жири та вуглеводи, але й вітаміни та мінерали, які необхідні для правильного функціонування організму.

Білки є основою для багатьох життєвих процесів в організмі. Вони складаються з амінокислот, деякі з яких є незамінними для людського організму, оскільки не можуть бути синтезовані і повинні надходити з їжею. У харчуванні зустрічаються білки як тваринного, так і рослинного походження. Білки тваринного походження мають високу біологічну цінність, тому що містять усі незамінні амінокислоти, тоді як білки рослинного походження часто не забезпечують організм повним набором амінокислот, тому важливо комбінувати різні види білків, наприклад, молочні продукти і хліб [1].

Надлишок білків, особливо тваринних, може призводити до накопичення сечової кислоти, що є фактором ризику для таких захворювань, як подагра і нирковокам'яна хвороба. Недостатність білка може спричинити

зниження імунітету, авітаміноз, уповільнене загоєння ран і загальне ослаблення організму [1].

Жири є важливим джерелом енергії і складаються з гліцерину та жирних кислот. Вони можуть бути насиченими або ненасиченими. Насичені жири (які зазвичай зустрічаються в тваринних продуктах) сприяють підвищенню рівня холестерину в крові, що збільшує ризик розвитку атеросклерозу. Ненасичені жирні кислоти (які в основному зустрічаються в рослинних оліях) мають корисний вплив на організм, зокрема, допомагають знижувати рівень холестерину і покращувати здоров'я судин [1].

Поліненасичені жирні кислоти, що є незамінними для організму, беруть участь у побудові клітинних мембран і в обміні холестерину. Дефіцит цих жирних кислот може призвести до порушень росту, зниження імунітету та проблем з шкірою. Але важливо пам'ятати, що надлишок жирів, особливо тваринних, може призвести до ожиріння і різних захворювань, таких як серцево-судинні хвороби [1].

Вуглеводи є основним джерелом енергії для організму. Вони поділяються на прості (моносахариди) і складні (полісахариди). Прості вуглеводи, такі як глюкоза та фруктоза, швидко всмоктуються і дають швидке джерело енергії. Складні вуглеводи, наприклад, крохмаль і клітковина, засвоюються повільніше, що забезпечує більш стабільне джерело енергії [1].

Обмін вуглеводів в організмі тісно пов'язаний з обміном жирів і білків. Якщо організм не отримує достатньо вуглеводів, він починає використовувати жири як джерело енергії. Однак надлишок вуглеводів може перетворюватися на жири, що накопичуються в організмі. Складні вуглеводи, такі як клітковина, не дають енергії, але важливі для нормального функціонування шлунково-кишкового тракту, стимулюючи перистальтику кишечника і запобігаючи закрепам [1].

Вітаміни є органічними сполуками, які необхідні для нормального функціонування організму. Вони регулюють метаболізм і є важливими для

нормального росту, підтримки імунної системи та здоров'я шкіри, очей, нервової системи. Вітаміни поділяються на водорозчинні (наприклад, вітаміни групи В, вітамін С) та жиророзчинні (вітаміни А, D, Е, К). Дефіцит вітамінів може призвести до серйозних захворювань, таких як цинга (дефіцит вітаміну С) або рахіт (дефіцит вітаміну D) [1].

Мінерали (макро- та мікроелементи) є важливими для підтримки електролітного балансу, функціонування клітин і органів. Макроелементи, такі як кальцій, магній, натрій, калій, фосфор, потрібні в великих кількостях і сприяють формуванню кісток, передачі нервових імпульсів і підтримці кислотно-лужного балансу. Мікроелементи, такі як залізо, йод, цинк, мідь, необхідні в малих кількостях, але вони критично важливі для метаболічних процесів. Наприклад, залізо є складовою частиною гемоглобіну, а йод — важливим елементом для щитоподібної залози [1].

Метою роботи є створення застосунку, який стане корисним інструментом для планування раціону харчування, і зможе, покращити та оптимізувати раціон харчування користувача, зважаючи на його індивідуальні потреби. Застосунок надасть користувачам можливість ефективно організувати свій щоденний раціон, враховуючи індивідуальні уподобання, дієтичні вимоги та обмеження.

1.1 Огляд існуючих методів вирішення завдання

Існує кілька підходів для створення персональних помічників, що допомагають у складанні щоденного раціону харчування. Ці методи варіюються від простих систем для підрахунку калорій до більш складних програм, які інтегрують функції планування меню, вибору рецептів, шопінгу та моніторингу здоров'я.

Системи для підрахунку калорій та макронутрієнтів – це системи, які дозволяють користувачам вводити дані про їжу та отримувати інформацію про кількість калорій, білків, жирів, вуглеводів та інших поживних елементів.

Вони можуть містити вбудовані бази даних з рецептами, що дозволяє автоматично обчислювати поживні значення страв. Такі системи також можуть враховувати індивідуальні потреби користувача, зокрема дієтичні обмеження або цілі здоров'я

Дієта на основі макронутрієнтів передбачає підрахунок споживання трьох макронутрієнтів — білків, жирів і вуглеводів. Вона розподіляє добові калорії на пропорції, такі як 10–35% білків, 20–35% жирів і 45–65% вуглеводів [5].

Користувач вводить інформацію про їжу (її тип, кількість), а система автоматично обчислює кількість калорій, білків, жирів і вуглеводів у кожній страві. Це допомагає користувачам слідкувати за їхньою добовою нормою і дотримуватися дієти. Системи можуть бути налаштовані з урахуванням індивідуальних потреб користувача. Наприклад, можна ввести спеціальні дієтичні обмеження (вегетаріанство, безглютенна дієта, кетогенна дієта) або медичні умови, як діабет, гіпертонія тощо. Це дозволяє оптимізувати харчування для конкретних цілей.

Прикладом такої системи є MyFitnessPal – одна з найбільш популярних програм для підрахунку калорій, яка має велику базу даних продуктів і рецептів. Вона також дозволяє зберігати інформацію про фізичну активність та вести статистику про спожиті макронутрієнти [6].

Інтелектуальні персональні помічники – це системи, які використовують штучний інтелект (ШІ) для аналізу індивідуальних потреб і уподобань користувачів. Вони автоматично адаптуються до поведінки користувача, надаючи персоналізовані поради, що дозволяє оптимізувати різні процеси, такі як планування раціону харчування, вибір продуктів або створення контенту. Ці помічники вивчають історію вибору та потреби користувача, на основі чого генерують рекомендації, що відповідають його особистим цілям, здоров'ю та перевагам.

Розмір ринку ШІ в харчовій індустрії оцінюється в \$9,68 мільярда у 2024 році і прогнозується, що він досягне \$48,99 мільярда до 2029 року з

темпом росту CAGR 38,30% [7]. Це свідчить про те, що ШІ — це майбутнє харчової індустрії, яке стимулює інновації, ефективність і сталий розвиток за допомогою різних застосувань ШІ. На рисунку 1.1 можна побачити зростання використання ШІ в харчовій індустрії [8].

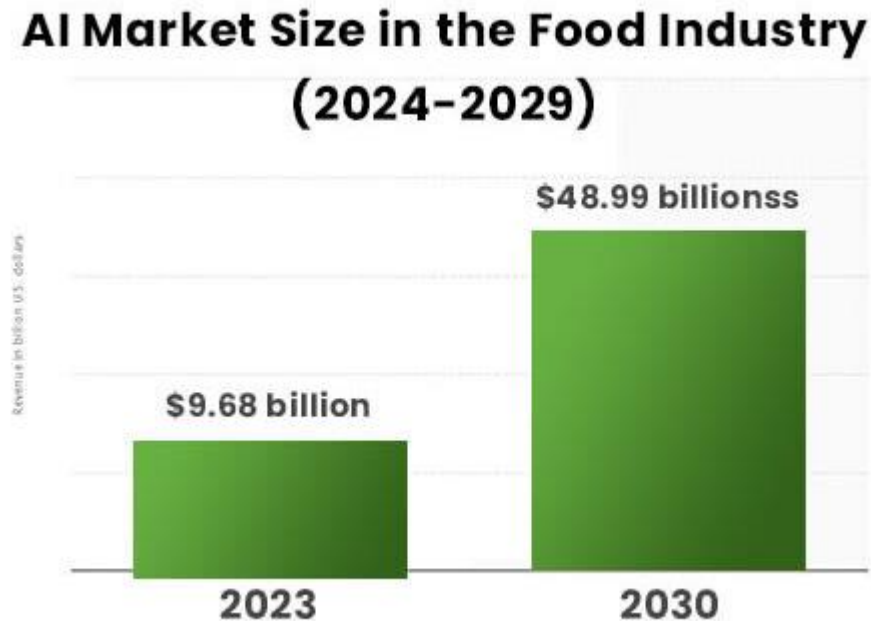


Рисунок 1.1 –Розмір ринку ШІ в харчовій індустрії [8]

ШІ у персоналізованому харчуванні включає використання передових алгоритмів і аналізу даних для створення індивідуальних дієтичних планів. Ця технологія використовує різні джерела даних, такі як генетична інформація, вибір способу життя та показники здоров'я, щоб надати поради щодо харчування, спеціально розроблені для кожної окремої особи [9].

Функції інтелектуальних персональних помічників у харчуванні:

- аналізують дані про користувача, щоб пропонувати страви та рецепти, що відповідають його харчовим потребам та вподобанням;
- допомагають створювати збалансовані плани харчування на день, тиждень або місяць, враховуючи калорійність, макронутрієнти;
- відстежують споживання їжі, надаючи зворотний зв'язок та рекомендації для досягнення здорових цілей;

– синхронізуються з фітнес-трекерами та іншими здоров'я-орієнтованими.

Наприклад, такі програми, як Noom і PlateJoy, використовують штучний інтелект для створення індивідуальних планів харчування на основі вподобань користувачів, цілей здоров'я та дієтичних обмежень [9].

Мобільні та десктопні застосунки для складання меню стали важливими інструментами для тих, хто хоче планувати своє харчування, враховуючи особисті переваги, здоров'я та дієтичні обмеження. Ці програми дозволяють користувачам вибирати страви, створювати меню на тиждень чи місяць, а також планувати харчування, враховуючи різні обмеження – від калорійних до специфічних дієт (наприклад, безглютенкових чи веганських).

Але хоча використання мобільних додатків може бути корисним, вони також мають обмеження та потенційні недоліки. Наприклад: багато додатків не завжди враховують індивідуальні особливості користувача, такі як генетичні фактори, особливі медичні умови чи харчові алергії. Це може призвести до рекомендацій, які не підходять для конкретної особи [10].

Наприклад, Paprika дозволяє користувачам зберігати рецепти, створювати меню і планувати харчування. Цей додаток також дає можливість синхронізувати рецепти між різними пристроями та інтегрувати списки покупок [11].

1.2 Огляд існуючих програмних засобів, що вирішують аналогічні завдання

Для аналізу існуючих рішень, що вирішують аналогічні завдання, було вивчено ринок мобільних та десктопних додатків, які спеціалізуються на плануванні харчування і створенні персоналізованих раціонів. В процесі аналізу були вивчені як популярні, так і нові програмні засоби, що використовуються для автоматизації складання меню на основі індивідуальних потреб користувачів, їхнього здоров'я, дієт чи цілей з

харчування. На основі вивченого ринку були обрані найбільш релевантні аналоги

1.2.1 Мобільний застосунок «Samsung Food»

Samsung Food – це мобільний застосунок, який дозволяє користувачам знаходити рецепти, створювати персоналізовані плани харчування, враховуючи дієтичні обмеження та вподобання, а також стежити за калорійністю страв. Програма доступна на смартфонах та планшетах із операційною системою Android та iOS, а також інтегрована з іншими продуктами Samsung для покращення користувацького досвіду [12].

Samsung Food — це розумний помічник, який пропонує користувачам широкий вибір з понад 160 тисяч рецептів, дає змогу створювати персоналізовані плани харчування та замовляти продукти онлайн. Сервіс також допомагає оптимізувати використання кухонної техніки, надаючи детальні покрокові інструкції для приготування страв, а також дозволяє ділитися рецептами через соціальні мережі. Платформа підтримує вісім мов і доступна в 104 країнах [13].

Завдяки впровадженню технологій штучного інтелекту, Samsung Food адаптується до індивідуальних потреб користувачів, надаючи їм можливість обирати страви відповідно до їхніх вподобань і сезонних продуктів. Платформа використовує потужну базу даних Whisk, яку Samsung придбав у 2019 році, а також інноваційну технологію Food AI, щоб рекомендувати страви на основі індивідуальних уподобань та доступних інгредієнтів [13].

Samsung Food пропонує користувачам комплексний набір функцій, що охоплює всі етапи приготування їжі. Платформа розділяється на чотири основні напрямки: пошук та налаштування рецептів, персоналізоване планування харчування, інтеграція з кухонною технікою та можливість публікації результатів у соціальних мережах. Завдяки технології штучного

інтелекту Samsung Food також пропонує можливість адаптації рецептів до індивідуальних дієтичних потреб [13].

Основні функції:

- користувачі можуть легко зберігати рецепти із різних джерел, таких як інтернет або самостійно додані рецепти. Всі рецепти зберігаються в одному місці для зручності;
- створення колекцій рецептів за категоріями, наприклад, здорове харчування, десерти, страви для вечері тощо, що дає змогу швидко знаходити потрібні рецепти;
- на основі збережених рецептів та вподобань користувача додаток може рекомендувати нові страви, що можуть сподобатися;
- додаток дозволяє створювати плани харчування на тиждень або місяць, даючи можливість планувати страви для кожного дня. Також можливе урахування дієтичних уподобань користувача, таких як веганство, безглютенова дієта, дієта для схуднення, інші харчові обмеження або алергії. Додаток пропонує рецепти, що відповідають цим вимогам;
- якщо користувач має смарт-холодильник або інші пристрої Samsung, додаток може інтегруватися з ними для надання рекомендацій щодо того, що варто приготувати з наявних продуктів [14];
- додаток використовує камеру пристрою для розпізнавання продуктів і перевірки наявності інгредієнтів у холодильнику або шафі. Застосунок може використовувати штучний інтелект для розпізнавання продуктів за допомогою камери. Це дозволяє відразу визначити, які інгредієнти у вас є вдома, і порадити страви, які можна приготувати [14];
- в деяких регіонах Samsung Food інтегрується з Instacart, що дозволяє користувачам замовити інгредієнти для своїх рецептів прямо через додаток, що зручніше та заощаджує час.

Робота застосунку продемонстрована на рисунку 1.2 [15].

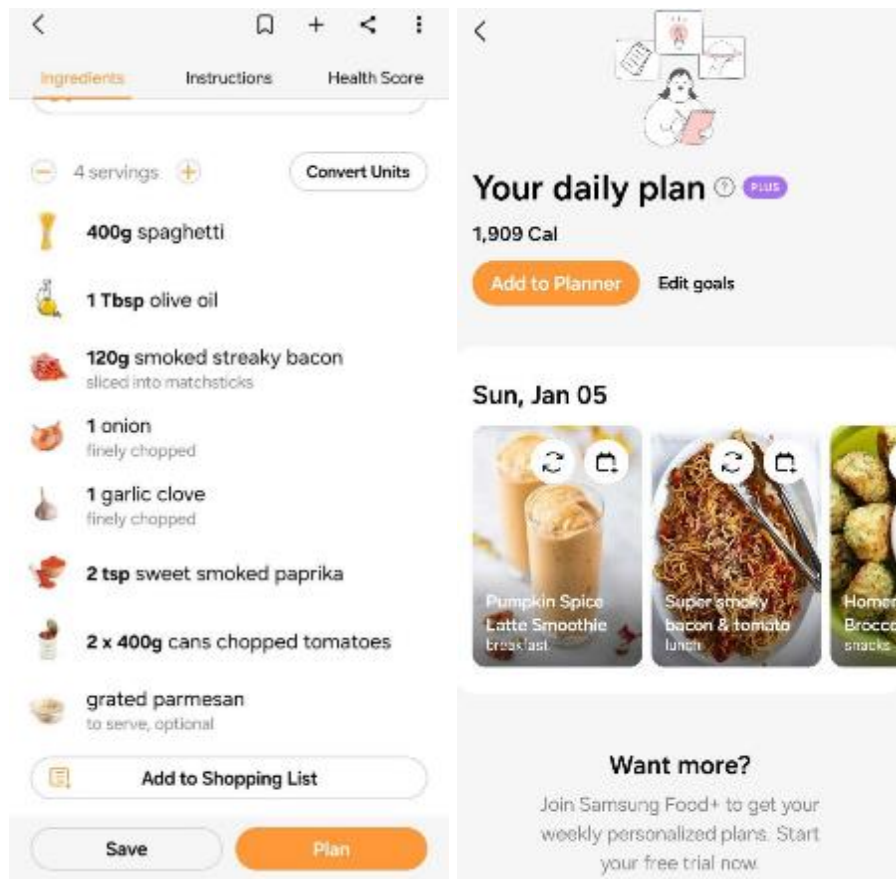


Рисунок 1.2 – Робота з мобільним застосунком «Samsung Food» [15]

Переваги:

- використання штучного інтелекту для персоналізації рецептів. Це дозволяє адаптувати рецепти відповідно до харчових уподобань користувача, його дієтичних потреб або обмежень;
- інтеграція з кухонною технікою, зокрема з холодильниками Bespoke Family Hub™. Це дозволяє користувачам отримувати рекомендації по рецептах, ґрунтуючись на продуктах, що вже є в холодильнику, що знижує витрати на продукти та запобігає їхньому псуванню. Крім того, функція автоматичного складання списків покупок значно спрощує процес планування харчування;
- Samsung Food підтримує вісім мов і доступний у 104 країнах, що дозволяє користуватися сервісом на глобальному рівні;
- надає користувачам покрокові інструкції з приготування їжі, що є корисним для тих, хто не має досвіду в кулінарії або хоче полегшити процес

готування. Також є можливість публікувати власні рецепти та досягнення в соціальних мережах, що додає елемент соціальної взаємодії та мотивації.

Недоліки:

- більшість функцій платформи найкраще працює при використанні техніки Samsung. Це обмежує можливість користуватися всіма перевагами платформи для тих, хто не має обладнання цієї марки. Оскільки не всі користувачі можуть дозволити собі техніку Samsung, це може обмежити доступність деяких функцій;

- інтерфейс програми може бути складним для новачків або тих, хто не знайомий з технологіями ІІІ. Деякі користувачі можуть знайти адаптацію до персоналізованих рекомендацій важкою, оскільки потрібно заповнити багато даних про свої вподобання та дієтичні потреби;

- ІІІ, що використовується в Samsung Food, може іноді видавати рекомендації, які не зовсім точні або відповідають уподобанням користувачів;

- багато з функцій доступні тільки в рамках платних підписок. Наприклад, персоналізація рецептів, інтеграція з деякими пристроями та доступ до більш розширених можливостей, таких як спеціалізовані плани харчування або доступ до ексклюзивних рецептів, можуть потребувати додаткових витрат;

- відсутній україномовний інтерфейс.

Samsung Food – це потужний інструмент для користувачів, які хочуть покращити своє харчування за допомогою персоналізованих рекомендацій, інтеграції з кухонною технікою та зручного планування раціону. Завдяки функціям на основі штучного інтелекту та доступу до широкої бази даних рецептів, додаток пропонує значну цінність для багатьох користувачів, особливо тих, хто має дієтичні обмеження або бажає полегшити процес приготування їжі.

Однак застосунок також має обмеження, пов'язані з залежністю від техніки Samsung, потенційною складністю інтерфейсу для новачків і

точністю персоналізованих рекомендацій. У підсумку, Samsung Food є чудовим вибором для користувачів, які вже використовують техніку Samsung і хочуть покращити своє харчування за допомогою технологій, але може бути менш корисним для тих, хто не знайомий з цією екосистемою.

1.2.2 Мобільний застосунок «MyFitnessPal»

MyFitnessPal – це найбільш популярний у світі додаток для відстежування харчування та їжі, заснований у 2005 році з метою допомогти людям по всьому світу досягати своїх здоров'яних цілей через кращий вибір їжі, надаючи знання, мотивацію та відчуття прогресу. Щороку MyFitnessPal допомагає майже мільйону людей досягати їхніх цілей в харчуванні та фітнесі, а його спільнота налічує понад 250 мільйонів користувачів у 120 країнах. MyFitnessPal пропонує своїм користувачам одну з найбільш комплексних платформ для відстеження харчування та їжі у світі, дозволяючи відстежувати їжу, записувати фізичну активність та вести облік ваги. Завдяки одній з найбільших баз даних продуктів у світі, яка містить понад 19 мільйонів продуктів, доступу до понад 2000 рецептів, більше 40 тренувальних програм і понад 40 партнерів для підключених фітнес-пристроїв, MyFitnessPal надає своїм користувачам інструменти для позитивних змін у здоров'ї [16].

Програму можна завантажити безкоштовно, і базова версія надає основні функції, такі як обчислення добової норми калорій, а також моніторинг білків, вуглеводів і жирів. Для отримання розширених можливостей необхідно підписатися на платну версію. Платна версія надає додаткові функції для більш детального відслідковування харчування та здоров'я [17].

MyFitnessPal використовує алгоритми, які допомагають створювати персоналізовані плани харчування, відслідковувати макронутрієнти та калорії, але сам по собі застосунок не використовує ІІІ для глибшої

персоналізації або автоматизації створення рецептів. Проте в майбутньому, з розвитком технологій, можливо, будуть додані більш інтегровані рішення, зокрема за допомогою ШІ для автоматичного налаштування планів харчування.

Робота застосунку продемонстрована на рисунку 1.3 [18].

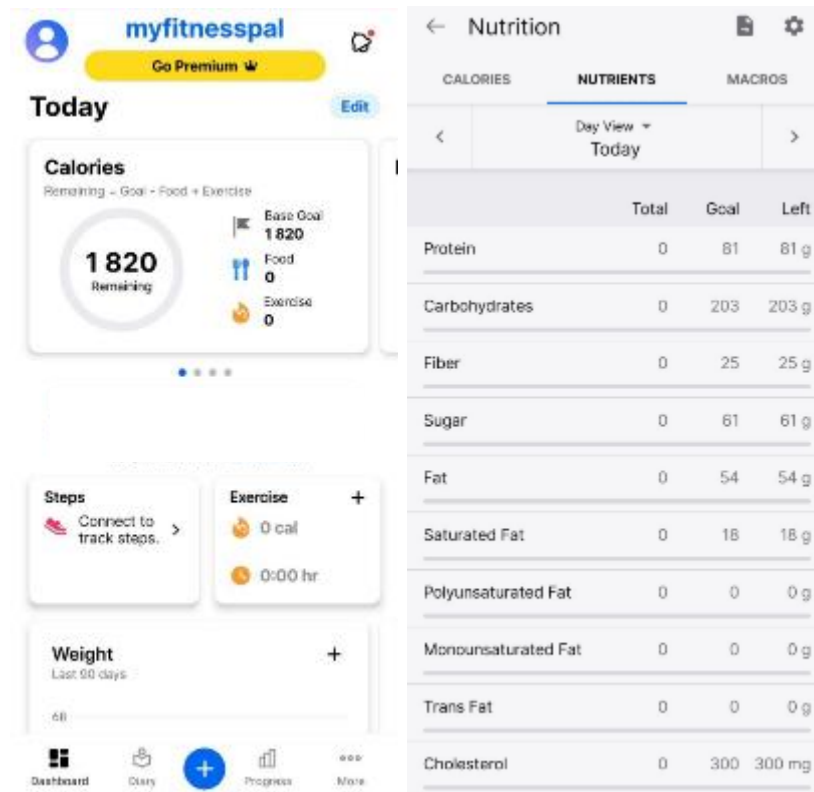


Рисунок 1.3 – Робота з мобільним застосунком «MyFitnessPal» [18]

Основні функції [17]:

- дозволяє користувачам вводити дані про їжу та напої для розрахунку калорій, білків, вуглеводів, жирів та інших макронутрієнтів. База даних містить мільйони продуктів, що дозволяє користувачам швидко знаходити і додавати продукти до свого щоденного раціону;
- підтримує інтеграцію з численними фітнес-апаратами та додатками, такими як Fitbit, Garmin, Apple Health та іншими, що дозволяє синхронізувати дані про фізичну активність та калорійний баланс;

- надає персоналізовані рекомендації, враховуючи фізичні характеристики користувача, його цілі щодо здоров'я та фітнесу. Це допомагає користувачам оптимізувати харчування відповідно до їхніх потреб (наприклад, для схуднення або набору м'язової маси);

- містить велику кількість продуктів, включаючи брендовані продукти, ресторани, рецепти, що дозволяє точно обчислювати кількість калорій і макронутрієнтів у раціоні.

Переваги:

- широка база даних продуктів. З величезною базою даних користувачі можуть швидко знаходити продукти та отримувати точну інформацію про їхній склад. Це дозволяє забезпечити точність у підрахунку калорій та макронутрієнтів;

- інтерфейс користувача дуже інтуїтивно зрозумілий. Користувачі можуть швидко додавати страви та слідкувати за своїм раціоном;

- пропонує потужні функції безкоштовно, що робить його доступним для широкого кола користувачів;

- підтримує інтеграцію з популярними фітнес-девайсами, що дозволяє відслідковувати активність та споживання калорій одночасно;

- користувачі можуть налаштувати свій план харчування в залежності від своїх цілей (схуднення, набір маси, підтримка здоров'я), що дозволяє оптимізувати дієту.

Недоліки:

- багато корисних можливостей доступні лише у Premium-версії. Це може бути обмеженням для тих, хто не хоче платити за додаткові функції;

- база даних містить величезну кількість продуктів, але деякі записи можуть бути неточними або некоректними, що може призвести до неточного підрахунку калорій або нутрієнтів;

- додаток фокусується на відслідковуванні їжі та калорій, але не має функції для автоматичного складання меню чи планування харчування на довгий термін;

- відсутній україномовний інтерфейс.

1.2.3 Сервіс «Eat This Much»

Eat This Much — це автоматизований планувальник харчування, який створює персоналізовані плани харчування, адаптовані до ваших вподобань, бюджету та розкладу [19].

Основні функції:

- автоматично генерує меню на основі введених даних (кількість калорій, яку потрібно споживати на день, бажаний макроелементний баланс (білки, жири, вуглеводи), особисті вподобання, спеціальні обмеження;
- система генерує меню на основі ваших потреб і харчових вподобань. Вона дозволяє користувачам змінювати страви або додавати нові рецепти. Меню може бути складене на тиждень або на місяць;
- має велику базу даних рецептів, що включає сотні різних варіантів страв. Користувачі можуть вибирати страви на свій смак або додавати власні рецепти;
- дозволяє створити список продуктів на основі меню і замовити їх через служби доставки, такі як Instacart чи Amazon Fresh (залежно від регіону);
- дозволяє користувачам відслідковувати свій прогрес у досягненні цілей харчування та здоров'я. Це включає у себе моніторинг ваги, рівня фізичної активності та споживаних калорій;
- у платній версії сервіс пропонує можливість професіоналам (дієтологам, тренерам) створювати персоналізовані плани харчування для своїх клієнтів, що є корисним інструментом для спеціалістів;
- використовує алгоритми для персоналізації планів харчування, що дозволяє автоматично адаптувати меню під потреби кожного користувача на основі введених даних.

Робота сервісу продемонстрована на рисунку 1.4 [19].

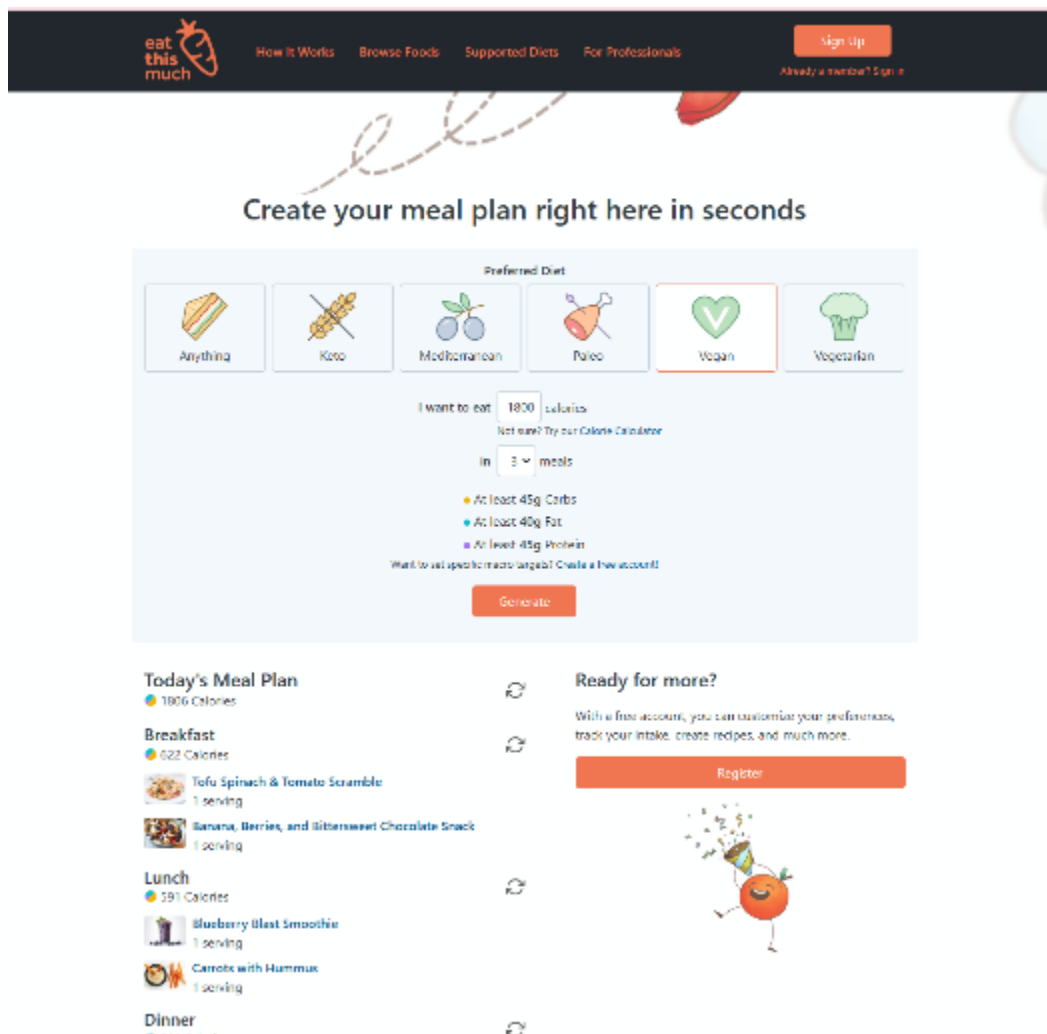


Рисунок 1.4 – Робота з сервісом «Eat This Much» [19]

Переваги:

- здатність створювати індивідуальні плани харчування, які відповідають потребам користувачів;
- дозволяє створювати списки покупок на основі меню і здійснювати замовлення продуктів через онлайн-сервіси доставки;
- крім мобільних додатків для iOS та Android, Eat This Much також доступний через веб-версію. Це дозволяє користувачам використовувати сервіс на різних пристроях і мати доступ до планів харчування у будь-який час;
- містить велику базу даних з рецептами, що дозволяє користувачам вибирати страви на свій смак або додавати власні рецепти.

Недоліки:

- більшість корисних функцій доступні лише у платній версії. Це може стати обмеженням для тих, хто не готовий платити за повний доступ до сервісу;
- для повноцінного використання сервісу необхідний постійний доступ до Інтернету. Це може бути незручним для користувачів, які хочуть використовувати додаток без доступу до мережі або під час подорожей;
- для нових користувачів, які не мають досвіду у плануванні харчування, користування Eat This Much може бути трохи складним. Є безліч параметрів для налаштування, і спочатку може бути важко зрозуміти, як правильно використовувати програму;
- відсутній україномовний інтерфейс.

1.2.4 Порівняльна таблиця існуючих та розроблюваного програмних продуктів

В результаті аналізу існуючих аналогів було вирішено детально порівняти їх між собою.

В таблиці 1.1 проведено порівняння вже існуючих програмних продуктів та програми, яка буде розроблена в ході виконання роботи.

Таблиця 1.1 – Порівняння програмних продуктів

Характеристика	EatMe	Samsung Food	MyFitnessPal	Eat This Much
1	2	3	4	5
Налаштування профілю користувача	+	+	+	+
Персоналізація харчування	+	+	+	+

Продовження таблиці 1.1

1	2	3	4	5
Створення плану харчування	+	+	+	+
Розрахунок норми калорій на день	+	+	+	+
Алергії та вподобання	+	+	+	+
Десктопна версія	+	-	-	+
Платний контент	-	+	+	+
Україномовний інтерфейс	+	-	-	-
Зручний інтерфейс	+	Частково	Частково	-
Використання ІІІ	+	+	-	-

У підсумку аналізу таблиці 1.1 стає очевидним, що «EatMe» вирізняється серед представлених варіантів завдяки унікальному поєднанню функцій, доступності, локалізації та зручності використання. Усі порівняні сервіси та застосунки забезпечують базове налаштування профілю користувача (зокрема параметри віку, статі та активності), створення планів харчування та розрахунок добової норми калорій. «Samsung Food», як і «EatMe», використовують ІІІ для персоналізації раціону харчування.

Інтерфейс «EatMe» виділяється своєю простотою та орієнтованістю на користувача, забезпечуючи максимальну зручність. У той час «Samsung Food» і «MyFitnessPal» пропонують лише частково інтуїтивний інтерфейс, а «Eat This Much» може здатися складним для новачків.

Десктопна версія доступна лише у «EatMe» та «Eat This Much» (веб-версія), що робить ці сервіси зручними для використання на комп'ютерах. Однак «Samsung Food», «MyFitnessPal» та «Eat This Much» мають платний контент, у той час як «EatMe» пропонує всі свої функції безкоштовно.

Ще однією важливою перевагою «EatMe» є наявність україномовного інтерфейсу, що робить його більш доступним для українських користувачів, тоді як інші сервіси такої функції не мають.

Загалом, «EatMe» демонструє найкраще поєднання функціоналу, доступності та локалізації, що робить його особливо привабливим для користувачів.

1.3 Порівняння алгоритмів для персонального планування раціону харчування

При вирішенні задачі персонального планування раціону харчування необхідно використовувати підхід, який дозволяє врахувати індивідуальні особливості, обмеження та вподобання користувача. Алгоритм повинен забезпечувати баланс між точністю, ефективністю та швидкістю роботи. У сучасній практиці існують кілька популярних методів, таких як жадібний алгоритм, методи лінійного програмування, метод Монте-Карло та генетичні алгоритми.

Жадібний алгоритм – це один із простих і швидких методів для розв’язання оптимізаційних задач. Його суть полягає в тому, щоб послідовно приймати рішення, які здаються найкращими на поточному кроці, без урахування глобального контексту. Це один із найпростіших підходів, який не потребує складної обчислювальної техніки [20].

Жадібні алгоритми – це клас алгоритмів, які на кожному кроці приймають локально оптимальні рішення з надією знайти глобальне оптимальне рішення. Прикладами популярних жадібних алгоритмів є задача про дробний рюкзак, алгоритм Дейкстри, алгоритм Крускала, кодування Хаффмана та алгоритм Пріма [21].

Переваги:

- жадібний алгоритм має низьку обчислювальну складність і підходить для задач, які потрібно розв'язати швидко. Він не потребує багато ресурсів для виконання;
- легко програмується та зрозумілий для впровадження. Його логіка базується на послідовному виборі найкращого локального рішення;
- не потрібно налаштовувати складні параметри або створювати додаткові моделі, як у випадку з нейромережами або генетичними алгоритмами;
- добре працює для задач, де локальне рішення збігається з глобальним оптимумом.

Недоліки:

- алгоритм може не знайти оптимального рішення, оскільки кожен крок базується на локально оптимальному виборі, без урахування глобальної картини;
- жадібний алгоритм схильний генерувати одноманітні результати;
- якщо завдання включає обмеження, алгоритм може не враховувати їх належним чином;
- залежність від впорядкування даних.

Жадібний алгоритм є хорошим вибором для швидкого і простого планування раціону харчування, якщо головна мета – досягти калорійної цілі. Однак для врахування індивідуальних потреб користувачів, таких як алергії, вподобання або різноманітність раціону, краще використовувати складніші методи, наприклад, генетичні алгоритми чи нейромережі.

Лінійне програмування – це математичний метод оптимізації, що використовується для знаходження найкращого можливого результату (максимуму або мінімуму) лінійної функції за певних лінійних обмежень. Цей підхід має широкий спектр застосувань, зокрема в плануванні харчування, логістиці, фінансовій оптимізації та управлінні ресурсами [22].

Переваги:

- завдяки строгій математичній основі лінійне програмування завжди знаходить оптимальне рішення, якщо воно існує, або повідомляє про відсутність рішення;

- дозволяє легко адаптувати модель шляхом додавання або зміни обмежень;

- забезпечує інструменти для аналізу чутливості, що дозволяє оцінювати, як зміни в параметрах впливають на оптимальне рішення.

Недоліки:

- для побудови моделі потрібно чітко визначити цільову функцію та всі обмеження у вигляді лінійних рівнянь. Це може бути складним завданням, особливо для задач із великою кількістю параметрів;

- лінійне програмування працює лише з лінійними залежностями між змінними. Нелінійні взаємозв'язки (які часто зустрічаються у реальних задачах) потребують інших методів, таких як нелінійне програмування;

- підходить для задач із неперервними змінними, але для дискретних задач;

- ефективність залежить від точності та повноти вхідних даних. Помилки або невизначеність у даних можуть призвести до неправильного рішення.

У контексті персонального планування раціону харчування лінійне програмування допомагає визначити набір страв, які задовольняють енергетичні та харчові потреби користувача при дотриманні обмежень, таких як алергії, вподобання чи бюджет. Попри свої переваги, підхід може бути менш ефективним у випадках, коли потрібно врахувати складні взаємозалежності, наприклад, варіативність смакових переваг.

Метод Монте-Карло – це стохастичний (імовірнісний) підхід, який використовує випадкове генерування даних для моделювання процесів або знаходження оптимальних рішень у складних системах. Цей метод застосовується в задачах із високою складністю або з багатьма невизначеностями [23].

Метод Монте-Карло є стохастичним підходом до оптимізації, який ґрунтується на багаторазовій генерації випадкових кандидатів і оцінці їх якості за допомогою цільової функції [24]. У контексті побудови раціону харчування кожен кандидатський раціон складається з обов'язкових прийомів їжі – сніданку, обіду та вечері – та додаткових перекусів.

Переваги:

- може бути використаний для розв'язання задач у будь-якій галузі, де є складні системи чи процеси;
- простий у реалізації, особливо у випадках, коли інші аналітичні методи є занадто складними;
- легко адаптується до задач із нелінійними функціями, складними обмеженнями або випадковими змінними;
- завдяки випадковості можна знайти глобальні оптимальні рішення, уникаючи локальних екстремумів.

Недоліки:

- для досягнення точного результату потрібно генерувати велику кількість випадкових варіантів, що може вимагати значних ресурсів;
- якщо простір рішень надто великий, метод може потребувати дуже багато ітерацій, що робить його повільним;
- метод не гарантує знаходження найкращого рішення. Він лише дає ймовірнісний результат, який залежить від обсягу симуляцій
- для точних результатів необхідно коректно налаштувати вхідні параметри.

Метод генерує випадкові варіанти наборів страв і оцінює їх відповідність критеріям, таким як калорійність, харчова цінність, вартість чи уподобання користувача. Він є корисним для задач, де точні параметри відомі частково або коли потрібно забезпечити максимальну варіативність у виборі рішень.

Хоча цей метод є гнучким і простим, він може бути менш ефективним у задачах із великим числом можливих варіантів, де потрібна висока точність і швидкість.

Генетичний алгоритм (ГА) — це метод оптимізації, який наслідує принципи природного відбору та генетики, такі як селекція, схрещення та мутація. Цей алгоритм особливо ефективний для задач, у яких простір можливих рішень великий, складний або нелінійний [25].

Генетичний алгоритм, використаний для оптимізації раціону харчування, наслідує принципи біологічної еволюції, моделюючи процес природного відбору [26]. У цьому підході кожен можливий план харчування інтерпретується як індивід популяції, де його «генами» виступають окремі страви та кількість їхніх порцій. Первинна популяція формується випадково, однак із врахуванням індивідуальних харчових уподобань користувача, що дозволяє від самого початку зменшити простір пошуку до більш релевантних комбінацій.

Якість кожного індивіда оцінюється за допомогою функції пристосованості [26].

Основні етапи роботи генетичного алгоритму:

- ініціалізація популяції. Початковий набір рішень (індивідів) генерується випадковим чином або з урахуванням певних правил;
- оцінка фітнес-функції. Кожен індивід оцінюється за фітнес-функцією, яка визначає, наскільки добре він відповідає поставленій задачі;
- селекція. Найкращі індивіди обираються для створення нових поколінь;
- схрещення (кросовер). Інформація між обраними індивідами комбінується для створення нащадків;
- мутація. Випадкові зміни в індивідах додають різноманітності до популяції;
- завершення. Алгоритм завершує роботу після досягнення заданого критерію (наприклад, кількість поколінь або рівень оптимізації).

Переваги:

- може працювати із завданнями, де кількість можливих рішень надто велика для перебору;
- завдяки різноманітності рішень у популяції, ГА має високу ймовірність уникнути локальних мінімумів і знайти глобальне оптимальне рішення;
- алгоритм можна адаптувати для широкого спектра задач, включаючи ті, що мають складні обмеження;
- імітує процеси природного відбору, що робить його легким для розуміння й впровадження;
- може обробляти кілька рішень одночасно, що дозволяє використовувати паралельні обчислення для прискорення процесу.

Недоліки:

- процес оптимізації може бути ресурсозатратним, особливо для задач із великою кількістю змінних;
- може бути повільним порівняно з іншими методами, якщо потрібно отримати дуже точний результат;
- ефективність ГА залежить від правильного налаштування параметрів, таких як розмір популяції, ймовірності мутації та кросовера

Генетичний алгоритм вирізняється своєю універсальністю та здатністю враховувати різноманітні фактори одночасно, що робить його сильним конкурентом жадібних алгоритмів, лінійного програмування та методу Монте-Карло.

Для системного порівняння ефективності алгоритмів оптимізації було сформовано таблицю 1.2, яка демонструє ключові показники роботи чотирьох методів: лінійного програмування, жадібного алгоритму, методу Монте-Карло та звичайного генетичного алгоритму. Вибір критеріїв порівняння включає час виконання пошуку раціонів, кількість згенерованих варіантів, продуктивність, а також середнє та найкраще відхилення від цільової калорійності. Таке комплексне представлення дозволяє оцінити

переваги та обмеження кожного алгоритму у задачах персоналізованого підбору раціонів.

Таблиця 1.2 – Порівняння алгоритмів

Критерій	LP	Жадібний алгоритм	Метод Монте-Карло	ГА
Час виконання (сек)	69,6	1,9	0,9	7,2
Кількість згенерованих варіантів	5	11	14	850
Продуктивність (варіанти/сек)	0,071	5,79	15,55	118,06
Середнє відхилення від цільової калорійності	1,75	11,9	1,75	7,4
Відхилення найкращого раціону від цільової калорійності	1,65	8,9	1,65	1,65

Аналіз даних таблиці 1.2 дозволяє зробити низку важливих висновків щодо ефективності розглянутих алгоритмів оптимізації. По-перше, за показником часу виконання пошуку раціонів, лінійний алгоритм демонструє найбільше значення, що зумовлено детальним перебором усіх можливих комбінацій, тоді як метод Монте-Карло характеризується мінімальним часом виконання. Жадібний алгоритм і звичайний генетичний алгоритм показують

проміжні значення, при цьому час виконання ГА залишається досить прийнятним для практичного застосування.

По-друге, аналіз кількості згенерованих варіантів свідчить, що генетичний алгоритм забезпечує значно ширшу популяцію рішень (850 варіантів), що підвищує ймовірність знаходження оптимального або близького до оптимального раціону. Лінійний алгоритм та метод Монте-Карло формують мінімальну кількість варіантів (5 та 14 відповідно), що обмежує їхню здатність до гнучкої персоналізації раціону. Жадібний алгоритм, хоча і швидкий, також формує невелику кількість варіантів (11), що робить його менш придатним для завдань комплексної оптимізації.

Що стосується продуктивності, виміряної як кількість варіантів на секунду, генетичний алгоритм значно перевищує всі інші методи (118,06 варіантів/сек), що свідчить про його високу ефективність при обробці великої кількості комбінацій. Метод Монте-Карло та жадібний алгоритм демонструють середню продуктивність, а лінійний алгоритм залишається найменш продуктивним.

За критерієм середнього відхилення від цільової калорійності, лінійний алгоритм і метод Монте-Карло показують найкращий результат (1,75), тоді як ГА демонструє трохи більші значення (7,4), що пояснюється більшою варіативністю сформованих раціонів. Жадібний алгоритм показує найгірший результат (11,9), що підтверджує його обмеженість для задач персоналізації.

Нарешті, відхилення найкращого раціону від цільової калорійності практично однакове для методів Монте-Карло, лінійного алгоритму та ГА (1,65), що свідчить про те, що навіть при великій кількості сформованих варіантів генетичний алгоритм здатний забезпечити високоточне наближення до бажаної добової калорійності. Жадібний алгоритм у цьому випадку відстає, демонструючи значно більше відхилення.

Отже, на основі проведеного порівняння алгоритмів оптимізації як проміжне рішення визначено, що генетичний алгоритм є найбільш

ефективним для подальшої роботи. Саме його доцільно використовувати у поєднанні з методами прогнозування для підвищення точності та гнучкості підбору персоналізованого раціону харчування.

1.4 Аналіз підходів до оптимізації персоналізованого раціону харчування

Для підвищення рівня персоналізації та точності рекомендацій доцільним є поєднання обраного методу оптимізації з методами прогнозування. Це забезпечує врахування індивідуальних особливостей користувача, прогнозування потреб організму та адаптацію раціону до динамічних змін у харчових вподобаннях і фізіологічних параметрах.

У сучасному світі персоналізація раціону харчування стає дедалі популярнішою, зокрема завдяки росту попиту на здоровий спосіб життя та індивідуальні потреби. Для розробки таких систем використовуються різні методи оптимізації. У цьому підпункті розглянуто та порівняно чотири основні підходи: рандомний ліс (Random Forest), градієнтний бустинг (XGBoost), та багатошарову повнозв'язну нейронну мережу (Multilayer Perceptron).

Рандомний ліс (Random Forest) – це ансамблевий метод машинного навчання, який використовує кілька дерев рішень для підвищення точності прогнозу. Кожне дерево будується на випадковому підмножині даних і характеристик, що дозволяє зменшити ризик перенавчання [27].

Переваги:

- простота в реалізації, особливо якщо кількість характеристик для аналізу невелика;
- висока стійкість до перенавчання завдяки використанню ансамблю дерев рішень;
- добре працює на невеликих наборах даних, що є корисним на початкових етапах аналізу.

Недоліки:

- не завжди забезпечує високу точність для задач прогнозування або оптимізації;
- погано масштабується для роботи з великими наборами даних чи високою кількістю характеристик;
- не враховує часову залежність, що може бути важливим у завданнях персоналізації раціону.

Градiєнтний бустинг – це техніка машинного навчання, яка використовує ансамбль слабких моделей, зазвичай дерев рішень, для створення сильної моделі. Вона послідовно покращує кожен модель шляхом корекції помилок попередньої, застосовуючи градiєнтний спуск для оптимізації функції втрат [28].

Переваги:

- висока точність і ефективність для задач сортування та класифікації;
- гнучкість у налаштуванні параметрів для досягнення оптимальних результатів;
- підтримка роботи з великими наборами даних.

Недоліки:

- вимагає більше обчислювальних ресурсів, ніж рандомний ліс;
- менш гнучкий порівняно з нейромережами, особливо у завданнях із сильною залежністю від користувацьких уподобань;
- труднощі з інтерпретацією моделей у випадках складних залежностей.

Нейронна мережа із повнозв'язними шарами (Dense Neural Network, DNN) є типом нейронної мережі, в якій кожен нейрон на кожному шарі з'єднаний з усіма нейронами наступного шару. Це дозволяє створювати моделі, здатні ефективно аналізувати складні взаємозв'язки між великими наборами вхідних даних [29].

Переваги:

- гнучкість у моделюванні складних взаємозв'язків між характеристиками користувача (алергії, уподобання, калорійна мета);
- висока точність у прогнозуванні завдяки можливості налаштування кількості шарів та нейронів;
- простота інтеграції з іншими методами, такими як підбір оптимальних параметрів через генетичні алгоритми;
- легкість масштабування на великих наборах даних.

Недоліки:

- висока потреба в обчислювальних ресурсах для навчання та обробки великих обсягів даних;
- може вимагати великих наборів даних для запобігання перенавчанню;
- складність у налаштуванні архітектури мережі та підборі гіперпараметрів.

Для порівняння методів прогнозування було обрано один із видів глибоких нейронних мереж – DNN, а саме багат шарову повнозв'язну нейронну мережу. Цей тип мережі характеризується повним з'єднанням між сусідніми шарами та відсутністю зворотних зв'язків, що робить її ефективною для задач, у яких кожен вхідний приклад обробляється незалежно. Використання MLP забезпечує високу гнучкість у моделюванні нелінійних залежностей між вхідними параметрами користувачів (вік, стать, рівень активності тощо) та вихідними показниками, що визначають оптимальний добовий раціон.

Аналіз даних таблиці 1.3 дозволяє детально оцінити переваги та недоліки кожного з методів прогнозування.

Таблиця 1.3 – Порівняння методів прогнозування

Критерій	MLP	Random Forest	XGBoost
1	2	3	4
Час навчання, секунди	3,1	0,275	0,187
Час сортування, секунди	0,335	0,304	0,291
Використання оперативної пам'яті, Мб	0,012	0.0552	0.1355

Аналіз таблиці 1.3 показує, по-перше, за критерієм часу навчання найшвидшим є алгоритм XGBoost, який забезпечує високу швидкість завдяки використанню бустингової стратегії та ефективних механізмів оптимізації. Random Forest також показує хороші результати з низьким часом навчання. У свою чергу, базовий багатошаровий перцептрон (MLP) демонструє найбільший час навчання (3,1 с), що пов'язано з більш складною структурою моделі та потребою у численних ітераціях для досягнення збіжності. Проте навіть цей час залишається прийнятним для практичних застосувань, оскільки вимірюється в секундах.

По-друге, за показником часу сортування всі три методи демонструють близькі результати. Це свідчить про те, що на етапі класифікації та прогнозування різниця між ними практично не впливає на швидкість роботи системи.

Найбільш суттєві відмінності проявляються у використанні оперативної пам'яті. Тут MLP показує найкращий результат (0,012 Мб), що є значно нижчим у порівнянні з Random Forest (0,0552 Мб) та XGBoost (0,1355 Мб). Цей критерій є ключовим у контексті даного дослідження, оскільки система має обробляти великі обсяги даних при персоналізації харчових раціонів. Мінімальне використання пам'яті забезпечує можливість масштабування та стабільну роботу без перевантажень.

Основною перевагою MLP у контексті даного дослідження є мінімальне використання оперативної пам'яті, що є критично важливим для обробки великого обсягу даних у персоналізованому підборі раціону харчування.

1.5 Опис обраних формул для розрахунків індивідуальних показників

Розробка персонального помічника для раціону харчування вимагає інтеграції точних розрахунків для визначення індивідуальних потреб кожної людини. Формули, що враховують ключові фізіологічні показники є основою для створення надійного та ефективного алгоритму. Використання математичних моделей дозволяє забезпечити індивідуальний підхід, що сприяє досягненню оптимального балансу калорійності, підтримці здоров'я та досягненню цілей користувача, будь то схуднення, набір маси чи підтримання поточного стану. Такий підхід робить програму не лише більш адаптованою до потреб кожної людини, а й більш науково обґрунтованою, що особливо важливо в контексті здорового харчування та профілактики захворювань.

Індекс маси тіла (BMI, Body Mass Index) – це стандартизований метод оцінки співвідношення маси тіла до зросту, який широко використовується в медицині та дієтології для визначення загального стану ваги людини. Цей показник дозволяє оцінити ризик розвитку хронічних захворювань, пов'язаних із зайвою вагою або її дефіцитом, таких як серцево-судинні хвороби, діабет 2 типу, гіпертонія тощо [30].

BMI розраховується за формулою (1.1) [30]:

$$BMI = \frac{Weight}{Height^2}, \quad (1.1)$$

де *BMI* – індекс маси тіла;

Weight – вага людини в кілограмах;

Height – зріст у метрах.

На основі значення ВМІ людина може бути віднесена до однієї з категорій ваги, встановлених Всесвітньою організацією охорони здоров'я (ВООЗ).

Категорії ваги визначаються за наступними значеннями [31]:

- якщо ВМІ менше 18.5 – «Недостатня вага»;
- якщо ВМІ у межах 18.5-24.9 – «Нормальна вага»;
- якщо ВМІ у межах 25-29.9 – «Зайва вага»;
- якщо ВМІ ≥ 30 – «Ожиріння».

Ці межі використовуються для визначення стану здоров'я та ризику розвитку хронічних захворювань.

Переваги використання ВМІ:

- необхідні лише два параметри: вага та зріст;
- використовується як стандартний інструмент у глобальних дослідженнях, що дозволяє порівнювати дані між різними країнами та групами населення;
- не потребує спеціального обладнання чи дорогих тестів.

Хоча ВМІ є корисним інструментом, він має певні обмеження:

- ВМІ не розрізняє м'язову та жирову масу. У спортсменів із високою м'язовою масою може бути високий ВМІ, хоча вони не мають надлишкової жирової тканини;
- формула однакова для всіх вікових груп і статей, хоча потреби організму різняться залежно від цих параметрів;

ВМІ є важливим показником для оцінки загального стану тіла, він не враховує ключових індивідуальних факторів, таких як стать, вікові особливості, м'язова маса чи рівень фізичної активності. Для більш детального та персоналізованого підходу до створення раціону харчування необхідно використовувати формули, які враховують базальний метаболізм. Ці формули дозволяють розрахувати кількість енергії, яку організм споживає

у стані спокою, з урахуванням таких індивідуальних показників, як вага, зріст, вік та стать. Це забезпечує більш точну оцінку калорійних потреб і створює основу для подальшого налаштування раціону відповідно до рівня активності та мети користувача.

Базальний метаболізм (BMR, Basal Metabolic Rate) — це кількість калорій, які організм витрачає в стані спокою. Навіть коли тіло не рухається, йому потрібна певна кількість калорій для підтримки основних функцій, таких як дихання, виробництво клітин і кровообіг [32].

Фактори, що впливають на BMR:

- з віком метаболізм сповільнюється через зниження м'язової маси та гормональні зміни;
- у середньому, чоловіки мають вищий BMR через більший обсяг м'язової тканини;
- більша маса тіла, особливо м'язова, сприяє збільшенню BMR;
- рівень фізичної активності;
- генетика визначає природний рівень метаболічної активності;
- гормони щитоподібної залози, статеві гормони, а також рівень кортизолу впливають на швидкість метаболізму;
- захворювання або стресові стани можуть змінювати BMR.

Існує декілька формул для оцінки BMR, які враховують основні фізіологічні параметри. Найпоширеніш [32]:

- рівняння Міффіна–Сан-Жеора (одне з найточніших);
- рівняння Харріса-Бенедикта (застаріле, але ще використовується).

Отже, було обрано рівняння Міффіна–Сан-Жеора. BMR для чоловіків розраховується за формулою (1.2) [32]:

$$BRM = 88.36 + (13.4 \times \text{Weight}) + (4.8 \times (\text{Height} \times 100)) - (5.7 \times \text{Age}), (1.2)$$

де *BRM* – базальний метаболізм;

Weight – вага людини в кілограмах;

Height – зріст у в метрах (помножено на 100, щоб перевести в сантиметри);

Age – вік у роках.

BMR для жінок розраховується за формулою (1.3) [31]:

$$BRM = 447.6 + (9.2 \times \text{Weight}) + (3.1 \times (\text{Height} \times 100)) - (4.3 \times \text{Age}), \quad (1.3)$$

Переваги використання BMR

- точна оцінка потреб у калоріях для основного метаболізму;
- можливість адаптації дієти залежно від індивідуальних потреб;
- основний компонент для побудови моделей прогнозування енергетичних потреб.

Обмеження BMR

- не враховує м'язову та жирову масу окремо, що може вплинути на точність у людей з високим вмістом жиру чи м'язів;
- залежить від точності вихідних даних (вага, зріст, вік);
- не враховує вплив короточасних змін, таких як стрес, хвороби або інтенсивні фізичні навантаження.

Для визначення добової потреби в калоріях використовуються дві основні складові: рівень фізичної активності та (ВМІ). Такий підхід дозволяє врахувати не лише базальні потреби організму, а й особливості способу життя та поточний стан здоров'я людини.

Коефіцієнти для фізичної активності:

- сидячий спосіб життя: $BMR \times 1.2$;
- помірно активний: $BMR \times 1.55$;
- активний: $BMR \times 1.725$;
- дуже активний: $BMR \times 1.9$

Коригування калорій залежно від ВМІ:

- недостатня вага: +10%;
- нормальна вага: без змін;

- зайва вага: -10% ;
- ожиріння: -20% .

Коефіцієнти, що використовуються в цих формулах, дозволяють точно оцінити потреби людини в калоріях залежно від її фізичного стану (активності, ваги) та здоров'я (категорія BMI).

1.6 Постановка завдання роботи

Під час аналізу технічного завдання (ТЗ) були виявлені основні функції, які повинно виконувати програмне забезпечення для персоналізованого планування харчування. Основні функції включають:

- налаштування профілю користувача. Програмне забезпечення повинно надавати можливість створення та налаштування особистого профілю, де користувач може вказати такі дані, як ім'я, вік, стать, зріст, вага, а також специфічні потреби, такі як алергії, вподобання в харчуванні (вегетаріанство, безглютенова дієта тощо) та рівень фізичної активності. Ці дані допомагають адаптувати програму харчування до індивідуальних потреб користувача;

- створення персонального плану харчування на день. Після заповнення профілю, користувач отримає персоналізований план харчування, що містить від 3 до 5 прийомів їжі на день. План повинен враховувати не тільки індивідуальні вимоги користувача, але й забезпечувати різноманітність раціону. Усі страви повинні бути збалансовані за поживними властивостями та відповідати рекомендаціям щодо здорового харчування;

- персоналізація плану харчування за минулими виборами користувача. Програмне забезпечення має вміти адаптувати план харчування на основі минулих виборів користувача. Це дозволяє враховувати уподобання та звички в харчуванні, що були зроблені раніше, і таким чином робить план більш індивідуальним;

– розрахунок денного споживання калорій. Програмне забезпечення повинно мати можливість автоматично розраховувати денну норму калорій на основі особистих даних користувача, таких як вага, вік, рівень активності та інші параметри. Це включає використання формул для розрахунку базового метаболізму (BMR) та коригування потреби в калоріях залежно від рівня фізичної активності (наприклад, сидячий спосіб життя, помірно активний, активний або дуже активний). Таким чином, користувач отримає точну кількість калорій, необхідних для підтримки здоров'я, схуднення або набору ваги.

1.7 Висновки за розділом 1

Під час проведення аналізу предметної області, були розглянуто:

- огляд предметної області;
- огляд існуючих методів вирішення завдання;
- огляд та порівняння популярних на ринку аналогів;
- огляд та порівняння алгоритмів для персонального планування раціону харчування;
- огляд і порівняння підходів до оптимізації персоналізованого раціону харчування;
- опис обраних формул для розрахунків індивідуальних показників;
- опис основних функцій програмного забезпечення.

Таким чином, прототипом було обрано Samsung Food, оскільки ця платформа найбільш схожа за функціоналом та орієнтована на персоналізацію харчування.

Методом було обрано інтелектуальні персональні помічники, оскільки вони дозволяють автоматизувати процес планування раціону та адаптувати його до потреб користувача за допомогою збору та аналізу персональних даних.

Алгоритмом став генетичний алгоритм, який дозволяє ефективно знаходити оптимальні рішення серед великої кількості варіантів.

Підходом до оптимізації персоналізованого раціону харчування була вибрана нейронна мережа з повнозв'язними шарами, що дозволяє точніше налаштовувати план харчування, враховуючи зміни в стані користувача.

Формули для BMI та BMR стали основою для розрахунку індивідуальних потреб у калоріях і допомагають персоналізувати рекомендації для кожного користувача.

2 ВИБІР ТА ОБҐРУНТУВАННЯ СТРУКТУРИ СИСТЕМИ, ЯКА ПРОЕКТУЄТЬСЯ

2.1 Вибір інструментарію розробки програмного забезпечення

Вибір інструментарію для розробки програмного забезпечення включає кілька ключових компонентів, таких як мови програмування, середовище розробки та система керування базами даних. Під час визначення мови програмування враховувалися різні аспекти проекту. Для створення інтерфейсу користувача була обрана мова, яка дозволяє ефективно розробляти графічні інтерфейси, а для реалізації штучного інтелекту та обробки великих даних – мова, яка підтримує численні бібліотеки та фреймворки для машинного навчання. Окремим етапом був вибір середовища розробки, що забезпечує гнучкість та сумісність з іншими інструментами проекту, а також дозволяє розробникам працювати в інтегрованому середовищі. Крім того, була обрана система керування базами даних для ефективного зберігання та обробки даних користувачів, їхніх вподобань та історії використання додатку. Ці компоненти були підібрані з урахуванням вимог до швидкості, масштабованості, безпеки та зручності під час розробки та використання системи.

2.1.1 Огляд особливостей мови програмування

Мова програмування – це система символів і правил, яка дозволяє людині задавати інструкції для комп'ютера. Вона містить визначену структуру синтаксису та семантики, що дозволяє писати алгоритми та управляти обчисленнями, ефективно перетворюючи ідеї в робочі програми [33].

Вибір мови програмування є критично важливим етапом у розробці програмного забезпечення, оскільки він впливає на ефективність, продуктивність, масштабованість та підтримку проекту. Правильний вибір

мови може значно скоротити час розробки, знизити кількість помилок та полегшити подальше обслуговування системи. Неправильний вибір може призвести до складнощів у підтримці, обмежити можливості масштабування та збільшити витрати на розробку [34].

Категоризація мов програмування [34]:

- мови низького рівня, такі як Assembly та C, надають високий рівень контролю над апаратними ресурсами. Вони зазвичай використовуються для програмування на системному рівні, де критичними є продуктивність і взаємодія з апаратним забезпеченням;
- мови високого рівня, такі як Python, Java та C#, надають пріоритет продуктивності розробника. Вони відомі своєю читабельністю та зручністю використання, що робить їх підходящими для широкого спектра застосувань;
- скриптові мови, такі як JavaScript та Ruby, відзначаються здатністю автоматизувати завдання та розробляти веб-додатки. Вони часто використовуються для створення інтерактивних веб-додатків і автоматизації повторюваних процесів

Популярність мов програмування не є статичною. Це динамічна сфера. Мови підвищуються та знижуються за рівнем популярності залежно від тенденцій індустрії, технологічних досягнень і підтримки спільноти. Хоча основні мови зазвичай залишаються незмінними, їх порядок може змінюватися [34].

Для продовження обговорення популярності мов програмування, важливо звернути увагу на індекси та результати опитувань, що проводяться серед розробників і професіоналів у галузі технологій. Одним з таких важливих джерел є Stack Overflow Developer Survey, яке проводиться щорічно і надає дані про найбільш популярні технології, мови програмування, інструменти та методи, які використовуються програмістами по всьому світу. Це опитування є одним з найбільших і найбільш авторитетних у галузі технологій і допомагає оцінити актуальні тенденції в

програмуванні. Результат опитування стосовно популярності мов програмування можна побачити на рисунку 2.1 [35]



Рисунок 2.1 – Найпопулярніші мови програмування, скриптові та розмітки за версією Stack Overflow Developer Survey [35]

На графіку з рисунку 2.1 про популярність мов програмування, Python, як правило, займає високе місце завдяки своєму широкому використанню в таких сферах, як машинне навчання, наука про дані та веб-розробка.

C# має стабільну популярність серед розробників десктопних програм, зокрема завдяки платформі .NET, яка надає потужні інструменти для створення багатофункціональних та ефективних десктопних застосунків. Крім того, використання таких фреймворків, як WPF і Windows Forms, робить C# потужним вибором для розробки десктопних додатків.

Крім того, існують інші джерела, що аналізують популярність мов програмування, такі як TIOBE Index. Індекс TIOBE (The Importance of Being Earnest) є показником популярності мов програмування. Індекс оновлюється

раз на місяць. Рейтинги базуються на кількості кваліфікованих інженерів у всьому світі, курсах і сторонніх постачальниках. Для підрахунку рейтингів використовуються такі популярні пошукові системи, як Google, Bing, Yahoo!, Wikipedia, Amazon, YouTube і Baidu [36].

На рисунку 2.2 можна побачити графік популярності мов програмування за Індексом TIOBE [37].

Січень 2025	Січень 2024	Зміна	Мова програмування		Рейтинги	Зміна
1	1			Python	23,28%	+9,32%
2	3	▲		C++	10,29%	+0,33%
3	4	▲		Java	10,15%	+2,28%
4	2	▼		C	8,86%	-2,59%
5	5			C#	4,45%	-2,71%
6	6			JavaScript	4,20%	+1,43%

Рисунок 2.2 – Найпопулярніші мови програмування за індексом TIOBE [37]

На графіку на рисунку 2.2 TIOBE Index для січня 2025 року видно, що Python продовжує утримувати лідерство, досягнувши 23,28% і збільшивши свою популярність на 9,32% порівняно з попереднім роком. У свою чергу, C# має 4,45% і втратило 2,71%, що відображає його зниження в рейтингу. Python значно покращив свої позиції, в той час як C# залишається на 4 місці.

Мова програмування Python була названа «Мовою програмування 2024 року» за версією TIOBE завдяки значному зростанню популярності — 9,3%. Вона випередила Java, JavaScript і Go, ставши домінуючою в багатьох сферах [37].

Також варто зазначити індекс PYPL (Popularity of Programming Languages). Індекс PYPL створюється шляхом аналізу частоти пошуку мов програмування в Google. Чим частіше шукають навчальні матеріали з певної мови, тим популярнішою вважається ця мова. Дані отримуються з Google Trends. Індекс PYPL може допомогти у виборі мови для навчання або для використання в новому програмному проєкті [36].

На рисунку 2.3 представлено рейтинг популярності мов програмування за індексом PYPL [38].

Worldwide, Jan 2025 :

Rank	Change	Language	Share	1-year trend
1		Python	29.8 %	+1.7 %
2		Java	15.35 %	-0.5 %
3		JavaScript	7.96 %	-1.0 %
4		C/C++	7.17 %	+0.3 %
5		C#	6.36 %	-0.3 %
6		R	4.63 %	+0.0 %
7		PHP	3.87 %	-0.6 %

Рисунок 2.3 – Найпопулярніші мови програмування за індексом PYPL [38]

Як видно з рисунку 2.3 Python утримує стабільно високі позиції завдяки широкому використанню в таких сферах, як машинне навчання, веб-розробка та автоматизація. C# також демонструє стабільний інтерес, особливо в розробці корпоративних рішень і ігрових додатках. Хоча Python продовжує зростати, C# залишається важливою мовою для специфічних застосунків.

У таблиці 2.1 описано порівняння кількох популярних мов програмування, серед яких Python і C#, що дозволяють чітко оцінити, що кожна з них підходить для різних цілей, таких як розробка інтерфейсів або ШІ.

Таблиця 2.1 – Порівняння мов програмування

Критерій	C#	Python	Java	C++
1	2	3	4	5
Тип мови	Об'єктно-орієнтована, статично типізована	Високорівнева, динамічно типізована	Об'єктно-орієнтована, статично типізована	Об'єктно-орієнтована статично типізована
Сфера застосування	Розробка інтерфейсів, десктопних додатків, веб-розробка	Штучний інтелект, наука про дані, веб-розробка	Мобільні додатки, серверні додатки	Системне програмування, ігри, високопродуктивні додатки
Основні переваги	Потужний для розробки інтерфейсів, багатофункціональний фреймворк .NET	Легкість у вивченні, багатий набір бібліотек для ШІ	Підтримка багатозадачності	Висока швидкість, контроль над пам'яттю, ефективність
Основні недоліки	Обмежена кросплатформність	Не найкраща продуктивність, не підтримує багатозадачність	Не найкраща продуктивність, не підтримує багатозадачність	Складність синтаксису, висока ймовірність помилок у пам'яті

Продовження таблиці 2.1

1	2	3	4	5
Продуктивність	Висока (особливо для Windows)	Помірна, оптимізація через бібліотеки	Висока, добре оптимізована JVM	Дуже висока, особливо для системного програмува- ння
Популярність	Висока, особливо для бізнес- додатків	Дуже висока, особливо в ІІІ та науці про дані	Висока, особливо для великих корпоратив- них додатків	Висока серед розробників інфраструк- тури та ігор

Вибір мови програмування залежить від конкретних вимог проекту. Як можна побачити з таблиці 2.1, для розробки інтерфейсу користувача та взаємодії з ним С# є відмінним вибором завдяки потужним інструментам для розробки графічних інтерфейсів через .NET Framework та WPF, а також високій продуктивності на платформі Windows. З іншого боку, Python є кращим вибором для реалізації штучного інтелекту завдяки своєму багатому набору бібліотек, високій читабельності та гнучкості для обробки великих даних та навчання моделей ІІІ.

Тому для інтерфейсу користувача та взаємодією з ним було обрано С#, а для штучного інтелекту – Python, враховуючи переваги кожної з мов у відповідних сферах.

С# був розроблений у 2000 році під керівництвом Андерса Хейлсберга в Microsoft як частина стратегії для платформи .NET. Він був створений з метою поєднати переваги С++ і Java, ставши при цьому більш безпечним і зручним для розробників. Мова швидко здобула популярність завдяки своїй

простоті та підтримці сучасних парадигм програмування, таких як об'єктно-орієнтоване програмування, багатопоточність та асинхронне програмування [39].

Перша версія C# була випущена в 2002 році, а з розвитком .NET Framework мова набувала нових можливостей. Вона також включала функції, що спрощують інтеграцію з іншими технологіями, такими як Windows API та COM. Важливим моментом в історії розвитку C# стало впровадження технології LINQ, яка спростила роботу з базами даних, а також підтримка асинхронного програмування, що зробило мову дуже популярною для розробки високопродуктивних додатків [40].

В 2010 році C# значно оновилася, додавши нові можливості для інтеграції з мобільними платформами, такими як Android і iOS через Xamarin. Мова також стала підтримувати більше платформ, таких як Linux, що розширило її використання в корпоративних рішеннях та розробці крос-платформених програм [40].

На сьогодні C# залишається однією з найпопулярніших мов програмування, активно підтримується компанією Microsoft і продовжує розвиватися, додаючи нові функції для задоволення вимог сучасних технологій, таких як штучний інтелект та блокчейн. Одним з ключових аспектів, що сприяє її популярності, є наявність великої спільноти, яка підтримує мову та забезпечує великий вибір бібліотек та фреймворків для розробників.

Основні особливості [41]:

- C# підтримує всі принципи об'єктно-орієнтованого програмування ООП, включаючи інкапсуляцію, наслідування та поліморфізм, що дозволяє створювати масштабовані та добре організовані додатки;
- мова має строгі типи даних і підтримує примітивні типи (як `int`, `double`) і складні типи (класи, структури, інтерфейси). Це допомагає забезпечити високу безпеку коду;

- C# має вбудовану систему збору сміття (garbage collection), що автоматично відслідковує і звільняє непотрібну пам'ять, зменшуючи ймовірність помилок;
- завдяки .NET Core C# дозволяє створювати додатки, що працюють на різних платформах, таких як Windows, macOS і Linux;
- C# підтримує лямбда-вирази, анонімні методи, асинхронне програмування (async/await), що робить мову зручною для розробки високопродуктивних додатків;
- C# використовує потужну бібліотеку класів .NET, яка містить інструменти для роботи з базами даних, веб-запитами, мережею, графічними інтерфейсами та багато іншого.

Переваги C# [42]:

- підтримує ООП, що дозволяє будувати масштабовані та добре організовані програми;
- підтримується в рамках .NET Framework, що пропонує велику кількість бібліотек, інструментів і фреймворків для розробки різних типів програмного забезпечення;
- компілюється в байт-код, що дозволяє досягти хороших результатів у порівнянні з іншими мовами високого рівня;
- має чистий синтаксис, який зберігає структуру програм, роблячи їх зрозумілими та легкими для читання;
- висока продуктивність;
- завдяки своїй структурованості, C# полегшує підтримку та розширення великих проєктів.

Недоліки C#:

- C# не дозволяє так гнучко працювати з апаратними ресурсами, як C або C++;
- оскільки C# працює з автоматичним управлінням пам'яттю, це може бути неефективним у ситуаціях з обмеженими ресурсами;

– обмежену кількість бібліотек та інструментів для машинного навчання.

Python – це високорівнева мова програмування, створена для зручності читання та простоти використання. Вона підтримує множинні програмні парадигми, зокрема об'єктно-орієнтоване, імперативне і функціональне програмування.

Python був створений Гвідо ван Россумом у кінці 1980-х років в Нідерландах. Спочатку розробка велася як вдосконалення мови ABC, орієнтуючись на підвищення читабельності та зручності для програмістів. У 1991 році Python був випущений як версія 0.9.0, що містила основні функції, які збереглися до сьогодні [43].

Мова почала активно розвиватися завдяки спільноті розробників, а вже у 2000 році вийшла стабільна версія 2.0, яка включала важливі покращення. У 2008 році було випущено Python 3, що не було сумісно з попередніми версіями, але принесло нові можливості, такі як поліпшена підтримка Unicode [43].

Python активно використовувався в наукових обчисленнях, наукових дослідженнях і веб-розробці завдяки бібліотекам, таким як NumPy, Pandas, Django та Flask. Завдяки цьому мова стала популярною серед розробників, науковців і інженерів.

Деякі особливості мови програмування Python [44]:

- простота синтаксису;
- підтримка різних парадигм програмування (об'єктно-орієнтоване, функціональне);
- широка бібліотека стандартних і сторонніх модулів;
- код Python може працювати на різних платформах;
- динамічна типізація;
- підтримка інтерпретованого виконання, що спрощує налагодження;
- використовується в багатьох сферах: від веб-розробки до штучного інтелекту.

Переваги Python [43]:

- простий синтаксис;
- підтримка об'єктно-орієнтованого, функціонального і імперативного програмування;
- велика кількість бібліотек;
- можливість запуску на різних операційних системах без змін у коді.

Недоліки:

- низька продуктивність. Python повільніший у порівнянні з деякими іншими мовами, такими як C++ чи Java;
- динамічна типізація може призвести до помилок у час виконання, особливо в великих проєктах;
- у створенні графічних інтерфейсів не має єдиного стандартного інструменту для створення GUI. Бібліотеки, як Tkinter, PyQt або Kivy, є окремими варіантами, кожен з яких має свої особливості, що може ускладнювати процес розробки;
- високе споживання пам'яті. Для великих проєктів Python може вимагати значних обсягів пам'яті.

Сьогодні Python є однією з найбільш використовуваних мов у таких сферах, як штучний інтелект, машинне навчання та обробка великих даних. Відкритий код і активна спільнота підтримують безперервний розвиток мови.

2.1.2 Огляд особливостей обраного середовища розробки

Інтегроване середовище розробки (IDE) – це програмне забезпечення, яке надає програмісту всі необхідні інструменти для створення, тестування та налагодження програмного забезпечення в одному інтерфейсі. Зазвичай IDE включає редактор коду, компілятор або інтерпретатор, налагоджувач (debugger) та інші інструменти для автоматизації складання програм.

Мета такого середовища – зробити процес розробки програм більш ефективним, забезпечуючи зручний доступ до всіх інструментів через єдиний

інтерфейс. IDE дозволяє програмістам зосередитися на написанні коду, зменшуючи необхідність вручну запускати та налаштовувати різні зовнішні інструменти.

Для аналізу популярності IDE можна також використати опитування від компаній та індекси які відображають загальні тенденції до популярності на сьогоднішній день.

Відоме вже опитування від Stack Overflow також проводить опитування стосовно популярності інтегрованих середовищ розробки. На рисунку 2.4 можна побачити результат цього опитування [35].

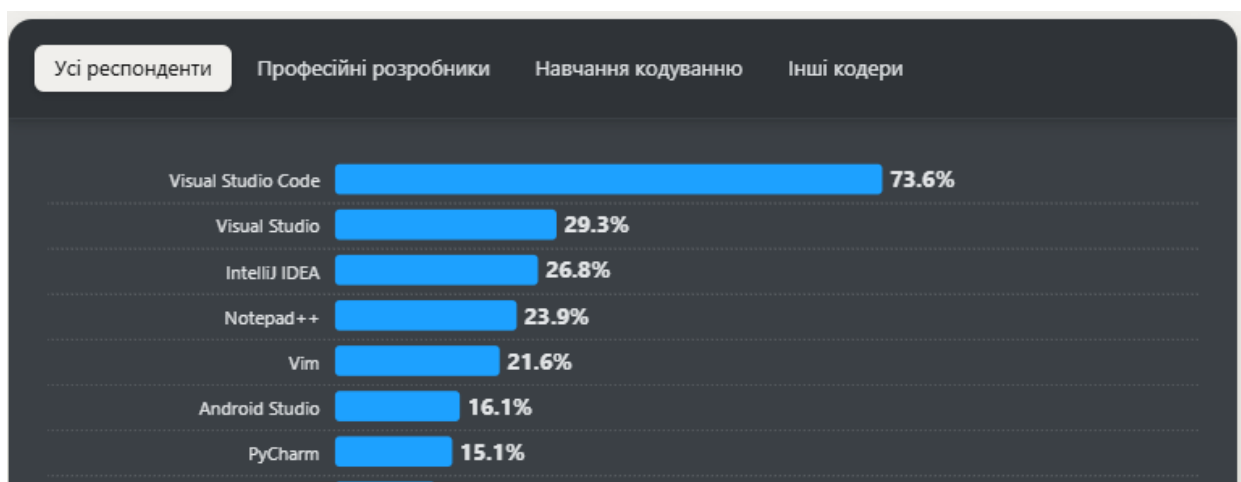


Рисунок 2.4 – Найпопулярніші інтегровані середовища розробки за версією Stack Overflow Developer Survey [35]

Як можна побачити з рисунку 2.4 Microsoft Visual Studio залишається дуже популярним IDE, але поступається Visual Studio Code.

Також ще один відомий індекс – це PYPL. Компанія також окремо надає інформацію про популярність IDE, яка створюється шляхом аналізу того, як часто сторінку завантаження IDE шукають у Google. Рейтинг за цим індексом можна побачити на рисунку 2.5 [45].

Worldwide, Jan 2025 :

Rank	Change	IDE	Share	1-year trend
1		Visual Studio	27.79 %	-0.1 %
2		Visual Studio Code	15.05 %	+1.5 %
3	↑	pyCharm	10.73 %	+0.6 %
4	↓	Eclipse	10.3 %	-0.8 %
5		Android Studio	10.18 %	+1.1 %

Рисунок 2.5 – Найпопулярніші IDE за індексом PYPL [45]

З рисунку 2.5 видно, що на січень 2025 року Microsoft Visual Studio є лідером по популярності серед розробників.

У таблиці 2.2 було порівняно декілька середовищ розробки, які дозволяють працювати як із Python, так і з C# в одному проєкті.

Таблиця 2.2 – Порівняння середовищ розробки

Характеристика	Microsoft Visual Studio	PyCharm	JetBrains Rider	VS Code
1	2	3	4	5
Підтримка Python	Вбудована через розширення	Рідна підтримка	Обмежена через плагіни	Розширення
Підтримка C#	Рідна підтримка	Обмежена через плагіни	Вбудована підтримка	Розширення

Продовження таблиці 2.2

1	2	3	4	5
Особливості	Інтеграція з .NET, чудовий налагоджувач, потужний інструментарій для великих проєктів, розширення для Python	Зручне середовище для Python, інструменти для аналізу коду, розширення для підтримки C#	Чудово підходить для C#, розширення для Python, легкий інтерфейс	Легка вага, плагіни для Python і C#, інтеграція з Git, безкоштовна
Мінуси	Великий розмір, потребує більше ресурсів	Менш підходить для великих C# проєктів	Не така зручна інтеграція для Python, платне середовище	Менше інструментів для великих проєктів

Згідно з таблицею 2.2 Microsoft Visual Studio є найкращим вибором для проєктів, які одночасно використовують Python і C#. Його потужні інструменти, чудова інтеграція з .NET та можливість розширення для роботи з Python роблять його ідеальним для великих і комплексних проєктів. Хоча Visual Studio потребує більше системних ресурсів, його функціонал повністю компенсує цей недолік, надаючи розробникам універсальний інструмент для роботи з багатьма мовами програмування. Саме тому для розробки програмного забезпечення було обрано середовище Microsoft Visual Studio.

Microsoft Visual Studio – це інтегроване середовище розробки (IDE), створене компанією Microsoft, яке підтримує широкий спектр мов програмування, таких як C#, Python, Visual Basic, C++, F#, JavaScript тощо.

Воно забезпечує інструменти для створення додатків для Windows, Android, iOS, веба, хмарних систем і багато іншого [46].

Особливості Visual Studio:

- підтримка багатьох мов програмування: C#, Python, C++, F# та інші;
- інтеграція з .NET забезпечує оптимізоване середовище для розробки .NET-додатків;
- інструменти для виявлення помилок на будь-якому етапі розробки;
- можливість додавання функціоналу через розширення для роботи з Python, Java тощо;
- вбудовані інструменти інтеграції з Azure DevOps та Git;
- підтримка веб-, мобільних, десктопних і хмарних додатків;
- бібліотека плагінів для розширення функцій IDE.

Переваги Visual Studio:

- підтримує розробку програмного забезпечення різних типів і складності;
- інтуїтивний інтерфейс – зручний і зрозумілий навіть для новачків;
- підходить як для невеликих, так і для великих команд;
- стабільність і підтримка. Постійні оновлення і велика кількість навчальних ресурсів.

Недоліки Visual Studio:

- IDE потребує значної кількості оперативної пам'яті та потужного процесора;
- займає багато місця на диску після встановлення;
- незважаючи на інтуїтивність, повноцінне використання всіх функцій вимагає часу.

Windows Presentation Foundation (WPF) – це підсистема для створення користувацьких інтерфейсів у Windows-додатках, розроблена Microsoft. Вона дозволяє створювати високоякісні графічні інтерфейси з використанням XAML (eXtensible Application Markup Language) для опису елементів і взаємодії між ними [47].

Особливості WPF:

- розмежування логіки та візуального представлення через XAML;
- забезпечує масштабованість елементів інтерфейсу;
- інструменти для створення складних ефектів і анімацій;
- повна сумісність із .NET Framework та Core;
- можливість створення тем для додатків;
- підтримка аудіо- та відео контенту.

Переваги WPF:

- висока продуктивність;
- гнучкий дизайн
- інструменти для роботи з 3D-графікою, анімацією та медіа;
- підтримка численних бібліотек і фреймворків.

Недоліки WPF:

- переважно орієнтована на Windows;
- для складних інтерфейсів на слабкому обладнанні можуть виникати затримки;
- непридатне для створення мобільних додатків;

Для Windows-додатків WPF залишається найкращим варіантом завдяки потужності, інтеграції з Visual Studio, підтримці XAML та гнучкості.

2.1.3 Огляд особливостей обраної системи управління базами даних

Система керування базами даних (СКБД) — це програмне забезпечення, яке дозволяє створювати, редагувати, керувати та взаємодіяти з базами даних. Вона забезпечує інструменти для організації, збереження, пошуку та маніпуляції даними в різних формах [48].

Основні функції СКБД [48]:

- створення та організація баз даних;
- додавання, видалення, оновлення та пошук даних;

- виконання SQL-запитів для отримання або аналізу даних;
- забезпечення безпеки даних, управління правами доступу;
- автоматизація процесів резервного копіювання та відновлення баз даних;
- оптимізація роботи та контроль навантаження.

Ці функції роблять СКБД незамінним інструментом для роботи з великими обсягами даних.

База даних – це структурований набір даних, які організовані для легкого доступу, управління та оновлення. Дані можуть бути представлені в різних формах: таблиці, документи, графи тощо [49].

Види баз даних та реляційна модель

- реляційна база даних організовує дані у вигляді таблиць, що складаються з рядків (записів) і стовпців (атрибутів). Ці таблиці пов'язані між собою через ключі, що забезпечує зручність пошуку та підтримання цілісності даних. Прикладом є MySQL, Microsoft SQL Server, PostgreSQL;

- нереляційна база даних підходить для великих даних, які важко структурувати у таблиці. Використовується для документів, графів, пар «ключ-значення». Приклади: MongoDB, Redis, Cassandra;

- об'єктно-орієнтовані бази даних інтегрують дані з об'єктами, що дозволяє зберігати і маніпулювати складними структурами. Приклад: db4o, ObjectDB;

- графові бази даних моделюють дані у вигляді вузлів і зв'язків між ними, що добре підходить для соціальних мереж чи рекомендаційних систем. Приклад: Neo4j;

- ієрархічні бази даних – дані організовані як дерево, де кожен вузол має один батьківський вузол. Приклад: IBM IMS.

На сьогоднішній день реляційна модель залишається найпопулярнішою завдяки своїй універсальності та широкому застосуванню.

Опитування від Stack Overflow також задає питання у розробників стосовно використання та популярності бази даних. На рисунку 2.6 можна побачити результат цього опитування [35].

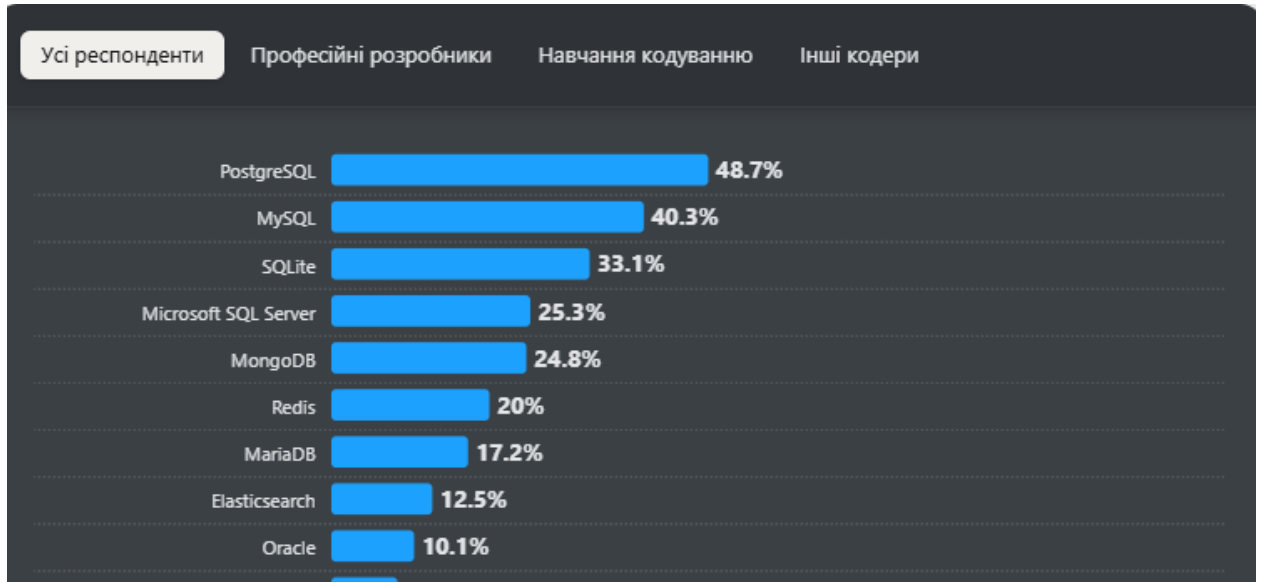


Рисунок 2.6 – Найпопулярніші бази даних за версією Stack Overflow Developer Survey [35]

Microsoft SQL Server — це реляційна система керування базами даних (RDBMS), розроблена Microsoft. Вона забезпечує масштабовані рішення для зберігання, обробки та аналізу даних з інтеграцією інструментів бізнес-аналітики та хмарних сервісів [50].

Причини, чому Microsoft SQL Server є найкращим вибором:

- підтримка великих обсягів даних і складних запитів;
- плавна інтеграція з Visual Studio;
- передові функції шифрування даних і захисту;
- зручність розробки;
- інструменти для роботи з великими даними та бізнес-аналітики.

2.1.4 Вибір бібліотеки для машинного навчання

PyTorch – це бібліотека з відкритим кодом для машинного та глибокого навчання, яка розроблена для швидкого прототипування моделей і продуктивної роботи з тензорами. Її особливістю є динамічний обчислювальний граф, який дозволяє змінювати структуру моделі «на льоту» без необхідності переписувати код. Ця бібліотека підтримує GPU-обчислення, що робить її ідеальною для роботи з великими наборами даних і складними моделями [51].

Основні причини вибору PyTorch:

- підтримка динамічного графа обчислень спрощує процес розробки моделей, особливо для задач із варіативними розмірами вхідних даних або складними логічними умовами;
- PyTorch дозволяє використовувати потужність графічних процесорів (GPU) для швидшої обробки даних і тренування моделей;
- PyTorch має велику кількість інтегрованих інструментів для машинного навчання, зокрема бібліотеку torchvision для обробки зображень;
- ком'юніті та документація: Активне ком'юніті розробників і велика кількість ресурсів (офіційна документація, курси, статті) полегшують навчання та вирішення проблем під час розробки.

NumPy та PyTorch часто використовуються разом, оскільки обидві бібліотеки мають спільні цілі — ефективну роботу з числовими даними. NumPy забезпечує швидкі обчислення на базі багатовимірних масивів і широко використовується для підготовки та обробки даних, тоді як PyTorch спеціалізується на побудові та тренуванні моделей машинного навчання, зокрема нейронних мереж. PyTorch навіть має інтеграцію з NumPy, дозволяючи легко конвертувати дані між ndarray (структурою NumPy) та тензорами, що забезпечує плавний перехід від етапу підготовки даних до побудови моделей машинного навчання.

NumPy (Numerical Python) – це базова бібліотека для наукових обчислень на Python, яка забезпечує підтримку багатовимірних масивів та матриць, а також широкий набір математичних функцій для роботи з ними. Вона служить основою для багатьох інших бібліотек, включаючи бібліотеки для машинного навчання та аналізу даних, таких як TensorFlow, PyTorch, і SciPy [52].

Основні причини вибору NumPy:

- NumPy оптимізована для виконання обчислень із великими обсягами даних, використовуючи низькорівневі оптимізації на основі C та Fortran;
- надає безліч функцій для роботи з масивами, включаючи математичні операції, лінійну алгебру, статистичний аналіз тощо;
- є стандартом для роботи з числовими даними в Python і часто використовується як основа для передачі даних між бібліотеками;
- NumPy розроблена та підтримується великою спільнотою розробників, забезпечуючи стабільну роботу та регулярні оновлення.

Після підготовки та обробки даних за допомогою NumPy, наступним логічним кроком у розробці моделей машинного навчання є вибір інструменту для побудови алгоритмів, їхнього налаштування та оцінки. Тут на допомогу приходить sklearn (Scikit-learn), яка пропонує широкий набір інструментів для моделювання даних, спрощуючи виконання складних завдань без необхідності створювати алгоритми з нуля.

Scikit-learn – це одна з найпопулярніших бібліотек Python для машинного навчання, яка забезпечує зручний інтерфейс для роботи з даними та реалізує велику кількість алгоритмів машинного навчання, включаючи регресію, класифікацію, кластеризацію та зменшення розмірності. Бібліотека побудована на основі NumPy, SciPy та matplotlib, що робить її невіддільною частиною екосистеми Python для наукових обчислень [53].

Scikit-learn є ідеальним вибором для цього проєкту завдяки своїм потужним функціям обробки даних та алгоритмів машинного навчання. Її

інтеграція з NumPy спрощує передобробку даних, а модуль `sklearn.preprocessing` дозволяє ефективно нормалізувати та масштабувати дані. Крім того, бібліотека забезпечує швидкий та простий доступ до широкого спектра методів моделювання, що значно прискорює процес розробки.

2.2 Структурна схема програми

Структурна схема – це графічне зображення, яке відображає основні компоненти програми, їх взаємодію та зв'язки між ними. Вона допомагає зрозуміти архітектуру програми на високому рівні, показуючи, як різні модулі або функції взаємодіють один з одним для досягнення кінцевої мети.

У структурній схемі повинно бути представлено основні елементи, які дозволяють зрозуміти логіку програми або системи, її компоненти, а також взаємозв'язки між ними. Вона повинна бути зрозумілою та повною, щоб на її основі можна було реалізувати програмний продукт.

Створення структурної схеми – важливий етап у життєвому циклі розробки програмного забезпечення, який забезпечує організованість, знижує ризик помилок та допомагає досягти високої якості кінцевого продукту.

У структурній схемі повинно бути представлено основні елементи, які дозволяють зрозуміти логіку програми або системи, її компоненти, а також взаємозв'язки між ними. Вона повинна бути зрозумілою та повною, щоб на її основі можна було реалізувати програмний продукт. Структурну схему зображено на рисунку 2.7.

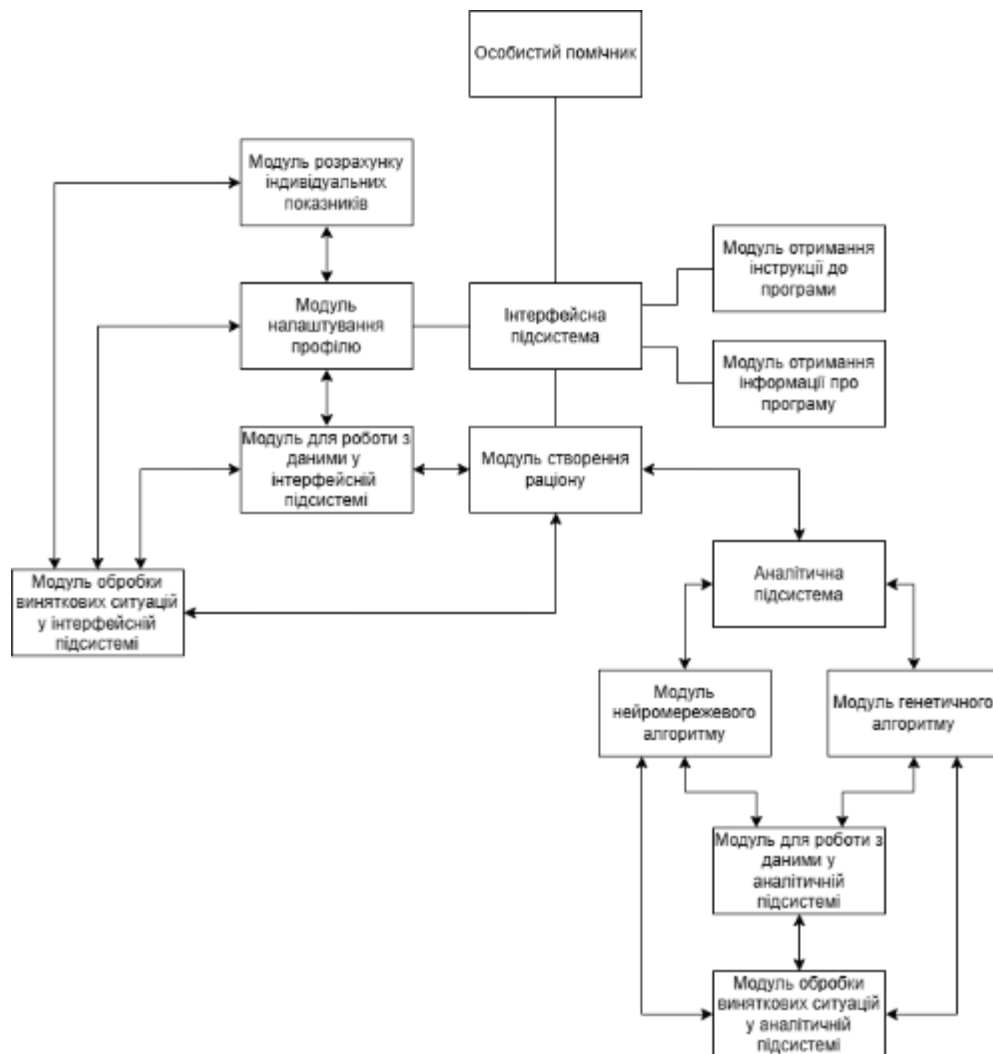


Рисунок 2.7 – Структурна схема проєкту

З наведеної структурної схеми на рисунку 2.7 видно, що застосунок складається з двох взаємопов'язаних підсистем: інтерфейсної та аналітичної.

Інтерфейсна підсистема призначена для забезпечення зручної взаємодії користувача із програмою. Її функціонал включає модулі, які відповідають за надання інструкцій з використання програми, доступ до інформації про програму, налаштування профілю та генерацію денних раціонів. Зокрема, модуль налаштування профілю та генерації раціонів інтегрується з модулем обробки даних, розташованим у графічній підсистемі. Крім того, модуль налаштування профілю здійснює взаємодію з окремим модулем, який виконує розрахунок індивідуальних параметрів користувача. Всі модулі інтерфейсної підсистеми взаємодіють із модулем обробки

виняткових ситуацій, який гарантує стабільну роботу інтерфейсу та своєчасне реагування на помилки.

Аналітична підсистема відповідає за формування раціонів на основі генетичного та нейромережового алгоритмів, використовуючи відповідні модулі для виконання цих задач. У процесі роботи генетичний та нейромережовий модулі взаємодіють із модулем обробки даних, розташований у межах аналітичної підсистеми. Аналогічно до інтерфейсної підсистеми, всі модулі аналітичної підсистеми тісно пов'язані з модулем обробки виняткових ситуацій, що забезпечує коректну роботу алгоритмів та мінімізацію впливу можливих помилок.

2.3 Опис модулів розроблюваної системи

У таблиці 2.3 представлено опис модулів програмного забезпечення і визначено їх функціонал.

Таблиця 2.3 – Опис модулів реалізованого ПЗ з визначенням основних функцій

Модуль	Опис
1	2
Розрахунку індивідуальних показників	Цей модуль відповідає за розрахунок індивідуальних показників BMI та BRM.
Налаштування профілю	Цей модуль відповідає за налаштування профілю користувача: віку, ваги, зросту, статі, рівню активності, алергій та вподобань
Створення раціону	Цей модуль відповідає за початок генерації раціонів та відображення і взаємодію з ними

Продовження таблиці 2.2

1	2
Роботи з даними у інтерфейсній підсистемі	Цей модуль відповідає у інтерфейсній частині застосунку за роботу даними: зчитування, запис та взаємодія з БД
Обробки виняткових ситуацій у інтерфейсній підсистемі	Цей модуль відповідає за обробку виняткових ситуацій у інтерфейсній частині застосунку та сповіщає користувача про проблему, якщо це необхідно.
Отримання інструкції до програми	Цей модуль відповідає за надання інструкції до програми користувачу
Отримання інформації про програму	Цей модуль відповідає за надання інформації про програму користувачу
Генетичного алгоритму калорій	Цей модуль відповідає за реалізацію генетичного алгоритму, який буде генерувати раціони які будуть приближені до оптимальної калорійності
Нейромережевого алгоритму	Цей модуль відповідає за реалізацію нейромережевого алгоритму, який буде сортувати раціони які будуть приближені до оптимальної калорійності беручи до увагу користувацьку історію вибору раціонів до цього
Роботи з даними у аналітичній підсистемі	Цей модуль відповідає у аналітичній частині застосунку за роботу даними: зчитування та запис
Обробки виняткових ситуацій у аналітичній підсистемі	Цей модуль відповідає за обробку виняткових ситуацій у аналітичній частині застосунку

2.5 Діаграма прецедентів

Діаграма прецедентів (англ. Use Case Diagram) — це графічне зображення взаємодії між зовнішніми користувачами (акторами) та системою через її функції (прецеденти). Вона є одним із ключових елементів візуального моделювання у мові UML (Unified Modeling Language) і використовується для представлення функціональних вимог до системи.

Діаграма складається з:

- акторів – ролей користувачів або зовнішніх систем, які взаємодіють із системою;
- прецедентів – функцій або послуг, які надає система акторам;
- зв'язків – відношень між акторами та прецедентами, а також між самими прецедентами.

На рисунку 2.8 представлена ця діаграма.

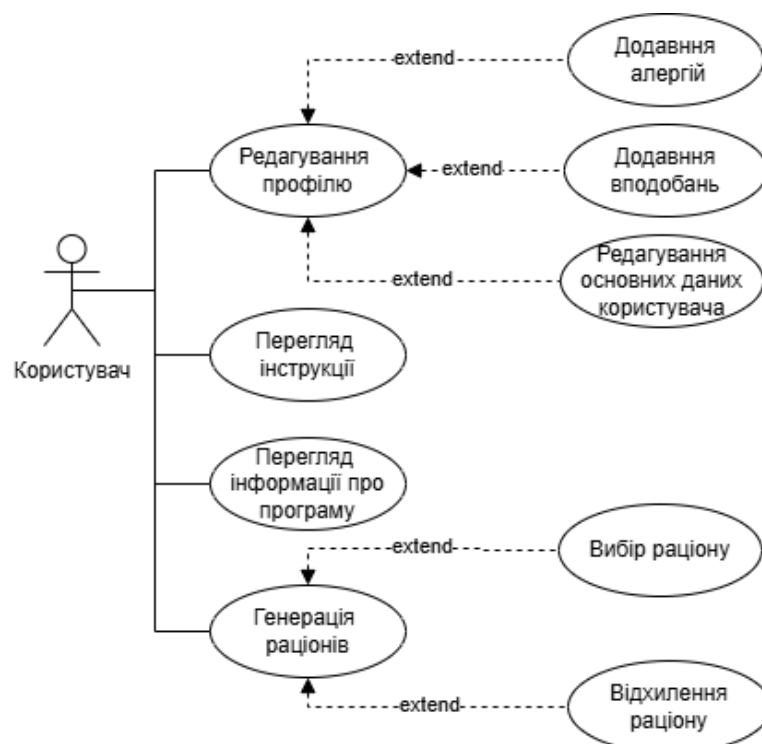


Рисунок 2.8 – Діаграма прецедентів

2.6 Висновки за розділом 2

Отже у підсумку за розділом 2 було:

- порівняно мови програмуванні, інтегровані середовища розробки за індексами, опитуванням серед розробників та особливостями;
- обрано мову програмування для інтерфейсу та взаємодії з ним – C#, мову для роботи з ШІ – Python;
- обрано інтегроване середовище розробки Microsoft Visual Studio;
- обрано СКБД Microsoft SQL Server;
- представлено структурну схему;
- у таблиці описано основні модулі програмного забезпечення.

Функціональна схема – це графічне представлення, яке демонструє зв'язки між різними елементами системи та описує їхню функціональну взаємодію [54]. Функціональну схему для ПЗ «EatMe» показано на рисунку 3.1.



Основна мета функціональної схеми – пояснити процеси, які відбуваються в системі, показати її структуру, зв'язки між компонентами і способ їх взаємодії.

З рисунку 3.1 можна побачити основні розроблені алгоритми, а саме:

- алгоритм генерації варіантів раціонів на день;
- генетичний алгоритм;
- нейромережевий алгоритм;
- алгоритм зміни даних користувача;
- алгоритм додавання алергій;
- алгоритм додавання вподобань;
- алгоритм вибору раціону;
- алгоритм відхилення раціону;
- алгоритм зчитування страв з бази даних у файли.

3.1.1 Алгоритм генерації варіантів раціонів на день

Призначення: цей алгоритм створено для того, щоб користувачі мали можливість згенерувати варіанти раціонів на день, які будуть підходити під данні які вони надали застосунку (як ім'я, вік, стать, зріст, вага, а також специфічні потреби, такі як алергії, вподобання в харчуванні) у відповідних полях у вікні «Профіль користувача». Ці дані допомагають адаптувати програму харчування до індивідуальних потреб користувача

На рисунку 3.2 можна побачити блок-схема алгоритму.

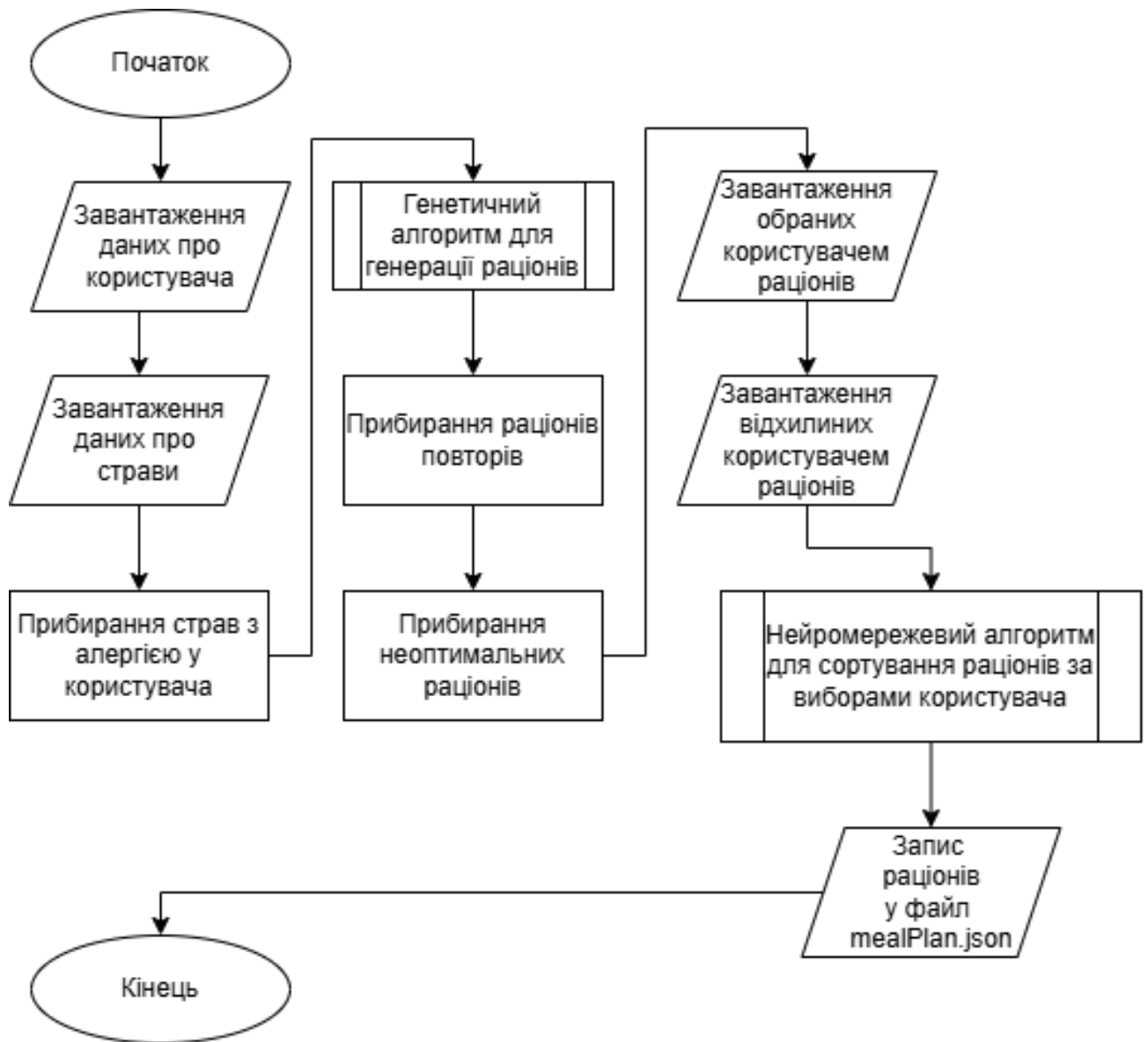


Рисунок 3.2 – Блок-схема для алгоритму генерації раціону на день

3.1.2 Генетичний алгоритм

У рамках дослідження створено модифікований генетичний алгоритм (ГА), побудований на основі елітарної моделі (elitist GA) із доменно-специфічними правилами генерації та мутації індивідів. Базовий ГА передбачає бінарне кодування хромосом, випадкову ініціалізацію популяції, класичні оператори схрещування та мутації. Запропонований підхід відходить від класичної моделі, зберігаючи загальну еволюційну парадигму,

проте застосовуючи адаптацію під предметну область – формування збалансованих харчових планів.

Кожен індивід I у популяції P репрезентується у вигляді плану харчування (MealPlan), тобто впорядкованої множини страв із відповідними порціями, що можна відобразити виразом 3.1:

$$I = \{(d_1, s_1), (d_2, s_2), \dots, (d_k, s_k)\}, k \in \{3, 4, 5\}, \quad (3.1)$$

де d_i – страва, вибрана з множини доступних (сніданки, обіди, вечері, перекуси);

$s_i \in \{1, 2, \dots, 5\}$ – кількість порцій.

Тоді загальна калорійність індивіда визначається формулою 3.2:

$$C(I) = \sum_{i=1}^k s_i * c(d_i), \quad (3.2)$$

де $c(d_i)$ – калорійність однієї порції страви d_i .

Метою оптимізації є мінімізація відхилення від заданої калорійної мети C^* . Виходячи з цього, функція пристосованості формулюється як вираз 3.3:

$$f(I) = |C(I) - C^*|, \quad (3.3)$$

Таким чином, чим менше значення $f(I)$, тим кращим вважається індивід.

Першою модифікацією була адаптована під завдання ініціалізація популяції. На відміну від класичного ГА, де початкова популяція формується випадковим чином, у даному алгоритмі застосовується доменно-специфічна

ініціалізація. При створенні індивіда з імовірністю 0.5 обираються страви, що відповідають індивідуальним уподобанням користувача, тобто формула 3.4:

$$P(d \in D_{fav}) = 0.5, P(d \in D \setminus D_{fav}) = 0.5, \quad (3.4)$$

де D_{fav} – множина страв, що містять інгредієнти з переліку преференцій користувача.

Другою модифікацією було використано елітний відбір, що передбачає формування нової популяції на основі найбільш придатних особин поточного покоління. Це можна виразити формулою 3.5:

$$P' = \operatorname{argmin}_{I \in P} f(I), |P'| = \frac{1}{2} |P|, \quad (3.5)$$

де P – поточна популяція, яка складається з набору рішень (раціонів);

$f(I)$ – цільова функція, яка оцінює якість кожного рішення I (наприклад, відхилення від цільової калорійності);

$\operatorname{argmin}_{I \in P} f(I)$ – означає вибір тих особин із популяції, для яких значення функції $f(I)$ є мінімальним, тобто найближчих до оптимального рішення;

P' – нова популяція, яка формується з найкращих особин;

$|P'| = \frac{1}{2} |P|$ – до нової популяції переходить рівно половина особин з початкової, причому саме та половина, яка має кращі показники (найменше відхилення від цільової функції).

На відміну від рулетки чи турнірної селекції, даний метод гарантує збереження найбільш пристосованих особин.

Третя модифікація стосується операції схрещення. Вона була адаптована під структуру індивідів у задачі формування раціону. Для цього застосовано одноточкове схрещення: беруться два батьківські рішення, обирається випадкова точка розділу, і новий індивід формується як

комбінація першої частини одного батька та другої частини іншого. Це дозволяє природним чином поєднувати страви та порції з різних рішень, створюючи більш різноманітні та потенційно кращі раціони.

Четверта модифікація стосується процесу мутації, який у генетичних алгоритмах відповідає за випадкове внесення змін у поточні рішення, щоб уникнути застою і дати змогу дослідити нові варіанти. Вона може відбуватися у двох варіантах:

- зміна кількості порцій – наприклад, для певної страви замінюється кількість порцій на іншу (від 1 до 5). Це дозволяє гнучко підлаштовувати калорійність під задану ціль;

- заміна страви – вибрана страва у плані харчування може бути замінена на іншу зі своєї категорії (сніданок, обід, вечеря чи перекус). При цьому алгоритм контролює, щоб у раціоні не було повторів одних і тих самих страв, тобто унікальність завжди зберігається.

Таким чином, алгоритм є елітним генетичним алгоритмом з модифікаціями на етапах ініціалізації, селекції, схрещування та мутації, які враховують предметну специфіку задачі (формування збалансованих харчових планів). Що дозволяє підвищити якість рішень за рахунок скорочення простору пошуку завдяки уподобанням користувача та збереженню найкращих індивідів на кожному кроці при адаптованих операторах мутації, що запобігають деградації рішень.

Алгоритм створено для того, щоб реалізувати генерацію варіантів раціонів на день, які максимально відповідають заданій оптимальній калорійності.

Алгоритм розпочинається зі створення початкової популяції раціонів, яка слугує першим поколінням. Це покоління формується з тисячі випадкових варіантів раціонів, при цьому виключаються страви, що містять інгредієнти, на які у користувача зазначена алергія.

Кожен раціон представлений у застосунку як набір страв із відповідною інформацією про калорійність, кількість порцій і використані

продукти. Процес еволюції реалізується через ітеративний цикл, який повторюється 100 разів.

На кожній ітерації всі раціони оцінюються за ступенем відповідності до заданої оптимальної калорійності.

Раціони, сумарна калорійність яких ближча до цільового значення, отримують вищу оцінку оптимальності. Після оцінки популяція сортується за ступенем відповідності до заданої калорійності, і найкраща половина особин популяції (500 раціонів), які демонструють найбільшу відповідність, відбирається для формування нового покоління.

Щоб забезпечити необхідний розмір популяції, рівний тисячі особин, до нової групи додаються раціони, отримані внаслідок схрещування випадково обраних особин із вже відібраної половини.

У процесі схрещування новостворені варіанти піддаються мутаціям із шансом 5%. Мутація полягає у зміні страви чи кількості порцій у раціоні, що сприяє підвищенню різноманітності популяції та забезпечує ефективнішу еволюцію.

Після завершення останньої ітерації алгоритм формує фінальну популяцію раціонів, яка представляє набір раціонів, найбільш наближених до оптимального розв'язання задачі.

На рисунку 3.3 можна побачити блок-схема алгоритму.



Рисунок 3.3 – Блок-схема для генетичного алгоритму

3.1.3 Нейромережевий алгоритм

Призначення: цей алгоритм створено для реалізації нейромережевого підходу до сортування варіантів раціонів на день на основі уподобань користувача, що визначаються з історії його вибору (обрані та відхилені раціони). Завдання алгоритму полягає у виявленні прихованих закономірностей у виборі користувача для створення персоналізованих рекомендацій.

На початковому етапі алгоритм отримує необхідні дані для роботи, включаючи набір раціонів, відповідну інформацію про інгредієнти, назви страв і попередні уподобання користувача. Після чого алгоритм для ефективної обробки даних створює два векторизатори на базі модуля `nn.Embedding`, які перетворюють текстові та категорійні дані (назви страв і список інгредієнтів) у числові векторні представлення. Ці вектори слугують вхідними даними для подальшого аналізу у нейромережевій моделі.

Після етапу векторизації відбувається ініціалізація нейронної мережі, структура якої включає кілька ключових компонентів. Вхідний шар (`nn.Linear`) відповідає за обробку об'єднаних векторних представлень, отриманих з векторизаторів. Вхідний шар забезпечує зменшення або збільшення розмірності векторів до заданого розміру, що спрощує подальшу обробку. Далі для додання нелінійності в моделі використовується функція активації `ReLU`, що дозволяє ефективно моделювати складні взаємозв'язки між характеристиками раціонів і вподобаннями користувача. Після на виході нейронної мережі використовується ще один шар `nn.Linear`, доповнений сигмоїдною функцією активації (`Sigmoid`), яка перетворює результат у ймовірність у вигляді числового значення від 0 до 1, що відображає оцінку.

Для оптимізації ваг моделі використовується алгоритм `Adam`, який забезпечує швидку та стабільну збіжність. Навчання моделі здійснюється за допомогою функції втрат `Binary Cross-Entropy Loss`, яка обчислює розбіжність між передбаченими ймовірностями та фактичними мітками (обраний чи відхилений раціон)..

Після ініціалізації відбувається етап навчання моделі на основі виборів користувача данні про які використовуються для оптимізації ваг мережі. Цей процес дозволяє алгоритму краще прогнозувати ймовірності того, що новий раціон відповідатиме вподобанням користувача.

На етапі прогнозування модель аналізує список нових варіантів раціонів. Для кожного з них модель обчислює ймовірність того, що раціон є позитивним (відповідає вподобанням користувача). Отримані значення

сортуються за спаданням, формуючи список. У цьому списку найкращі варіанти, які з найбільшою ймовірністю відповідатимуть уподобанням користувача, знаходяться на початку.

На рисунку 3.4 зображено саме блок-схема алгоритму.



Рисунок 3.4 – Блок-схема для нейромережевого алгоритму

3.1.4 Алгоритм зміни даних користувача

Призначення: цей алгоритм створено для того, щоб користувачі мали можливість змінити данні які будуть використані для генерації раціонів для них. На рисунку 3.5 зображено саме блок-схема алгоритму.



Рисунок 3.5 – Блок-схема для алгоритму зміни даних користувача

3.1.5 Алгоритм додавання алергій

Призначення: цей алгоритм створено для того, щоб користувачі мали можливість данні про алергії які будуть використані для генерації раціонів для них. На рисунку 3.6 зображено саме блок-схема алгоритму.



Рисунок 3.6 – Блок-схема для алгоритму додавання алергій

3.1.6 Алгоритм додавання вподобань

Призначення: цей алгоритм створено для того, щоб користувачі мали можливість данні про вподобання які будуть використані для генерації раціонів для них. На рисунку 3.7 зображено саме блок-схема алгоритму.



Рисунок 3.7 – Блок-схема для алгоритму додавання вподобань

3.1.7 Алгоритм вибору раціону

Призначення: цей алгоритм створено для того, щоб користувачі мали можливість обрати раціони які їм подобаються для покращення рекомендацій у майбутньому.

На рисунку 3.8 зображено саме блок-схема алгоритму.

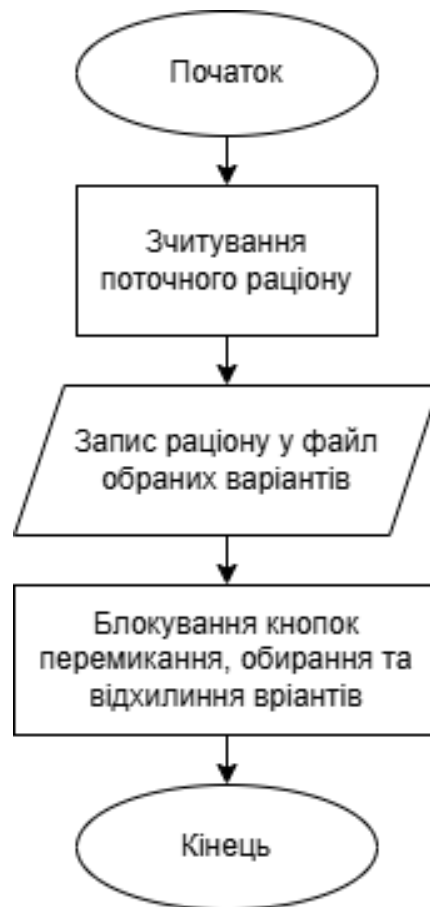


Рисунок 3.8 – Блок-схема для алгоритму вибору раціону

3.1.8 Алгоритм відхилення раціону

Призначення: цей алгоритм створено для того, щоб користувачі мали можливість відхилити раціони які їм не подобаються для покращення рекомендацій у майбутньому.

На рисунку 3.9 зображено саме блок-схема алгоритму.



Рисунок 3.9 – Блок-схема для алгоритму «Коментування рецепту»

3.1.9 Алгоритм зчитування страв з бази даних у файли

Призначення: цей алгоритм створено для того, щоб зчитати з таблиць бази даних у файли данні (списки сніданків, обідів, вечерь та перекусів), які будуть використані у подальшій роботі програми під час виконання ГА для генерації оптимальних варіантів раціону на день.

На рисунку 3.10 зображено блок-схема алгоритму.

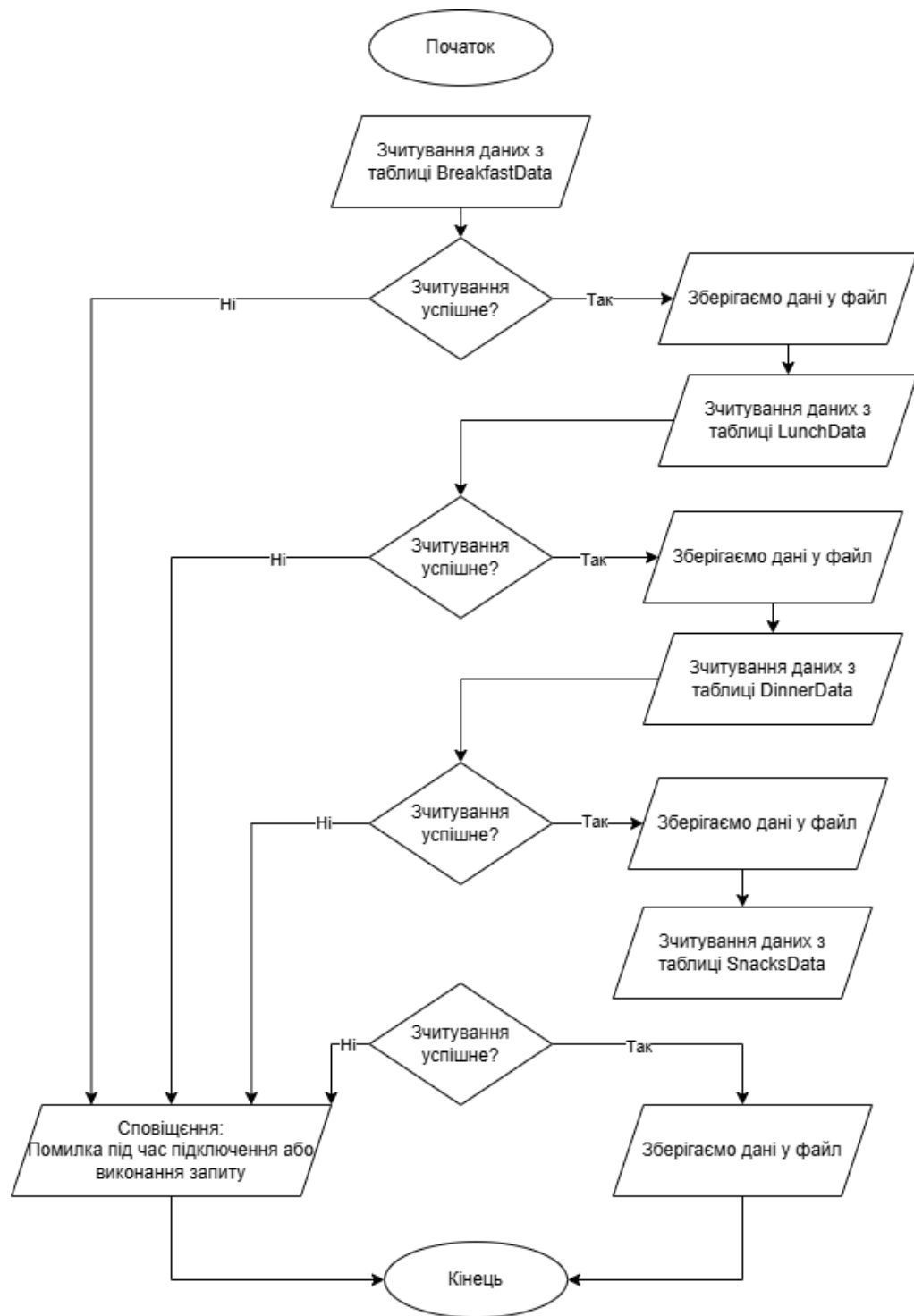


Рисунок 3.10 – Блок-схема для алгоритму «Редагування рецепту»

3.2 Опис архітектури створеної моделі нейронної мережі

Розроблена нейронна мережа реалізована у вигляді багатошарової повнозв'язної архітектури, яка вирішує задачу бінарної класифікації. Вона

побудована за принципом повного зв'язку між шарами, де кожен нейрон попереднього шару передає інформацію до кожного нейрона наступного.

Багатшарова повнозв'язна нейронна мережа (також відома як багатшаровий перцептрон, MLP) – це тип штучної нейронної мережі, у якій кожен нейрон одного шару з'єднаний з кожним нейроном наступного шару. Така архітектура дозволяє моделювати складні, нелінійні взаємозв'язки в даних [55].

MLP складається з вхідного шару, одного або кількох прихованих шарів і вихідного шару. Кожен нейрон виконує лінійне перетворення своїх вхідних даних, після чого застосовує нелінійну активаційну функцію. Це дозволяє мережі навчатися розпізнавати складні шаблони та взаємозв'язки [55].

На рисунку 3.11 представлена архітектура нейронної мережи застосунку «EatMe».

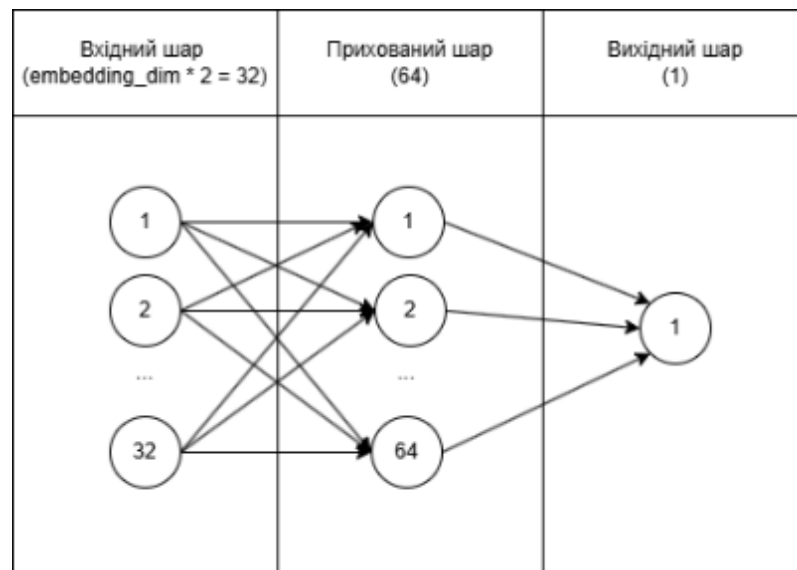


Рисунок 3.11 – Архітектура MLP

Вхідний шар складається із двох компонент: векторизації інгредієнтів та назв страв.

Векторизація інгредієнтів – шар вбудованих представлень (ingredient_embedding) перетворює індекси інгредієнтів на вектори

розмірності `embedding`. Обчислення середнього значення векторів інгредієнтів забезпечує узагальнене представлення для кожної страви.

Векторизація назв страв – аналогічно, шар `dish_name_embedding` перетворює індекси назв страв у вектори розмірності `embedding`.

Отримані вектори об'єднуються шляхом конкатенації, формуючи вхідний вектор розмірності, який передається до прихованого шару розраховується за формулою (3.6):

$$\text{embedding_dim} = 2 \times \text{embedding}, \quad (3.6)$$

де *embedding_dim* – вхідний вектор розмірності;

embedding – вектор розмірності;

Прихований шар є повнозв'язним і містить 64 нейрони. Кожен нейрон приймає вхідні дані з усіх нейронів попереднього шару. Лінійне перетворення вхідного вектора здійснюється через оператор `Linear` з наступною функцією активації `ReLU`, яка розраховується за формулою (3.7):

$$h = \text{ReLU}(W_x + b), \quad (3.7)$$

де *h* – вектор виходів прихованого шару;

b – вектор зміщення;

x – вхідний вектор;

W – матриця ваг розмірності, яка розраховується за формулою (3.8):

$$W = (64 \times 2\text{embedding_dim}), \quad (3.8)$$

Вихідний шар представлений одним нейроном, що приймає вхідні сигнали від кожного з 64 нейронів прихованого шару. Лінійне перетворення із функцією активації `Sigmoid` перетворює сумарний сигнал у значення ймовірності, яке розраховується за формулою (3.9):

$$y = \sigma(W_o h + b_o), \quad (3.9)$$

де y – значення ймовірності;

W_o – ваги вихідного шару;

b_o – зміщення;

$\sigma(x)$ – матриця ваг розмірності, яка розраховується за формулою (3.10):

$$\sigma(x) = \frac{1}{1 + e^{-x}}, \quad (3.10)$$

Вихідний нейрон прогнозує ймовірність позитивної оцінки раціону, що дозволяє класифікувати раціон як схвалений чи не схвалений.

Також для кращого відображення архітектури створено зображення у графічному вигляді на рисунку 3.12.

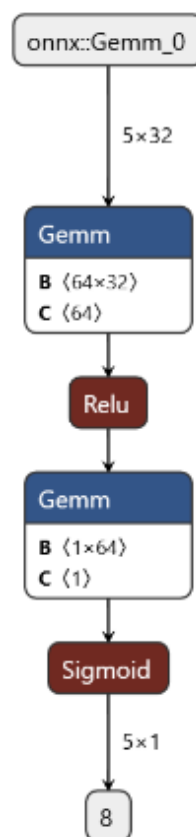


Рисунок 3.12 –Графічне відображення MLP

Особливості архітектури:

- усі нейрони кожного шару підключені до всіх нейронів наступного шару, що забезпечує врахування всіх можливих залежностей між інгредієнтами та назвами страв;
- використання ReLU у прихованому шарі та Sigmoid на виході забезпечує гнучкість моделі та можливість роботи з ймовірностями;
- під час навчання використовується функція втрат Binary Cross-Entropy (BCELoss), яка є стандартом для задач бінарної класифікації.

Дана нейронна мережа реалізує компактну та ефективну архітектуру для моделювання залежностей між інгредієнтами та назвами страв у контексті оцінки раціону харчування. Її модульна структура дозволяє адаптувати модель до різних типів даних та забезпечує високий рівень узагальнення.

3.3 Опис модулів розроблюваної системи

Для розроблених модулів були створені класи, що належать до інтерфейсної підсистеми:

- AboutWindow – реалізує вікно з інформацією про програму;
- AllergyAndPreferencesWindow – реалізує вікно вибору продуктів вподобань чи алергій;
- DailyMealPlanWindow – реалізує вікно для відображення, генерації, вибору та відхилення раціонів на день;
- InstructionsWindow – реалізує вікно з інструкцією до програми;
- MainWindow – реалізує вікно головного меню;
- MessageWindow – реалізує вікно для обробки сповіщень про помилки та успіхи;
- ProfileWindow – реалізує вікно з профілем;
- DataManager – реалізує роботу з базою даних та файлами;
- Dish – реалізує екземпляри сутності «Страва»;

Рисунок 3.13 – Діаграма класів інтерфейсної підсистеми

Для реалізації функціональності створених модулів було розроблено та впроваджено відповідні класи для аналітичної підсистеми:

- GeneticAlgorithm – реалізує генетичний алгоритм для генерації оптимальних раціонів;
- NeuralNetwork – реалізує нейромережний алгоритм для сортування раціонів залежно від минулих виборів та відхилень користувача;
- DataManager – реалізує роботу з файлами;
- Logger – реалізує обробку виняткових ситуацій у вигляді ведення логів;
- Dish – реалізує екземпляри сутності «Страва»;
- MealPlan – реалізує екземпляри сутності «Раціон»;
- User – реалізує екземпляри сутності «Користувач».

Для ілюстрації класової ієрархії аналітичної підсистеми було створено відповідну діаграму класів, яку зображено на рисунку 3.14.

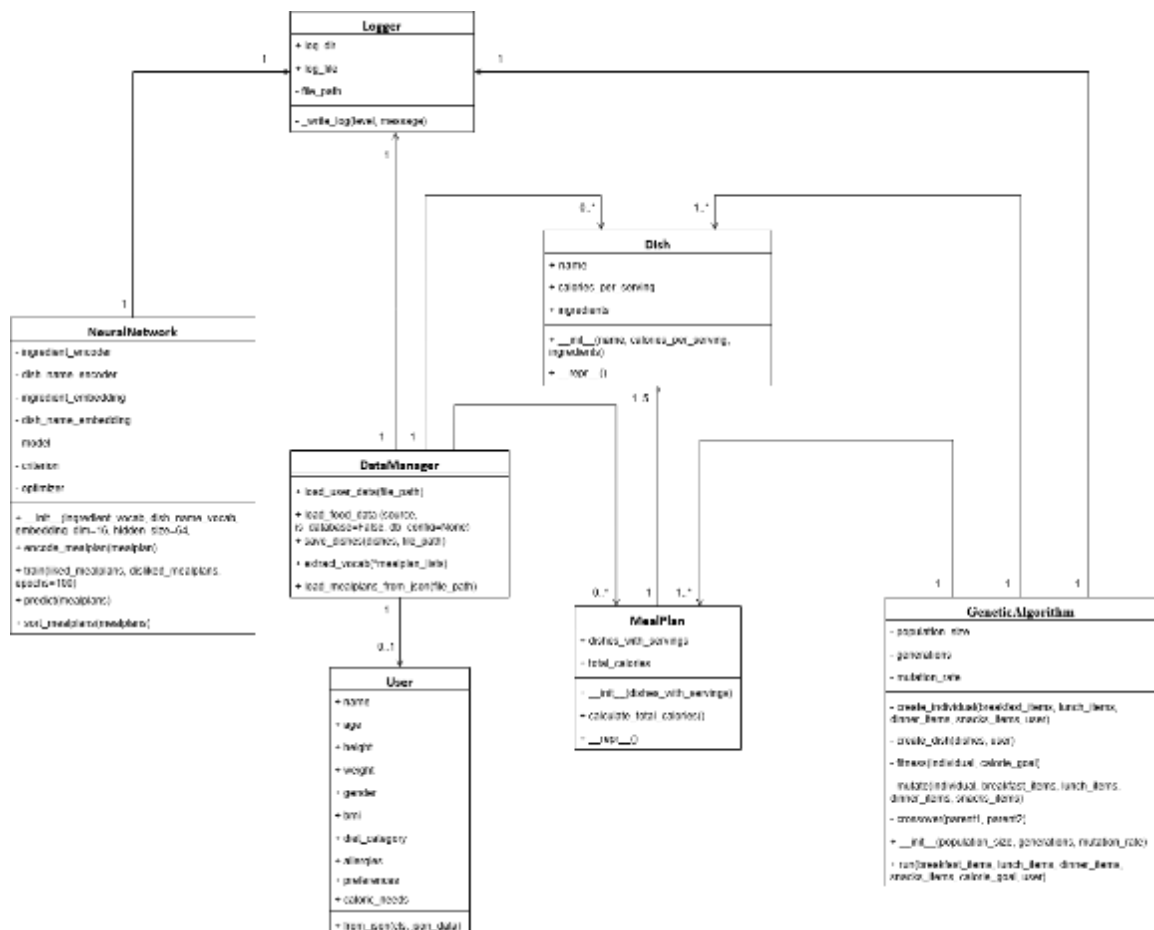


Рисунок 3.14 – Діаграма класів аналітичної підсистеми

3.3.1 Клас AboutWindow

Ціль класу полягає у реалізації вікна з інформацією про програму.

Дані та методи класу наведені у таблиці 3.1.

Таблиця 3.1 – Опис полів та методів класу AboutWindow

Поля та методи класу	Опис
<i>public:</i>	
public AboutWindow()	Конструктор, який ініціалізує вікно

3.3.2 Клас AllergyAndPreferencesWindow

Ціль класу полягає у реалізації вікна реалізує вікно вибор продуктів вподобань чи алергій. Дані та методи класу наведені у таблиці 3.2.

Таблиця 3.2 – Опис полів та методів класу AllergyAndPreferencesWindow

Поля та методи класу	Опис
1	2
<i>public:</i>	
public AllergyAndPreferencesWindow(Profile Window profileWindow, string windowType, List<string> productList)	Конструктор, який ініціалізує вікно
<i>private:</i>	
ProfileWindow	Змінна з об'єктом вікна профілю
string windowType	Змінна з типом вікна, який визначає, чи обробляти алергії або вподобання

Продовження таблиці 3.2

1	2
List<string> productList	Змінна з списком продуктів
private void SetCheckBox()	Метод, який встановлює стан кожного чекбокса на основі списку продуктів
private void Window_Closing(object sender, System.ComponentModel.CancelEventArgs e)	Метод, який викликається перед закриттям вікна. Оновлює список продуктів залежно від обраних чекбоксів і передає їх у ProfileWindow

3.3.3 Клас DailyMealPlanWindow

Ціль класу полягає у реалізації вікна відображення, генерації, вибору та відхилення раціонів на день. Дані та методи класу наведені у таблиці 3.3.

Таблиця 3.3 – Опис полів та методів класу DailyMealPlanWindow

Поля та методи класу	Опис
1	2
<i>public:</i>	
public DailyDietWindow()	Конструктор, який ініціалізує вікно
public async void GenerateMealPlanButton_Click(object sender, RoutedEventArgs e)	Метод, що обробляє подію натискання кнопки для генерації нового плану харчування. Виконує асинхронне отримання даних і оновлення вікна

Продовження таблиці 3.3

1	2
public void PastRationButton_Click(object sender, RoutedEventArgs e)	Метод, що обробляє подію натискання кнопки для перегляду попереднього плану харчування
public void NextRationButton_Click(object sender, RoutedEventArgs e)	Метод, що обробляє подію натискання кнопки для перегляду наступного плану харчування
public void ChooseMealPlanButton_Click(object sender, RoutedEventArgs e)	Метод, що додає поточний план до списку обраних у JSON-файлі та деактивує кнопки управління
public void RejectMealPlanButton_Click(object sender, RoutedEventArgs e)	Метод, що видаляє поточний план зі списку, додає його до списку відхилених у JSON-файлі та оновлює вікно
public async void GifLoad()	Метод, що завантажує та відображає GIF-анімацію під час очікування генерації плану харчування
<i>private:</i>	
private List<MealPlan> mealPlansList	Змінна з списком доступних планів харчування
private int currentMealPlanIndex	Змінна з індексом поточного плану харчування в списку
private List<MealPlan> GetDailyDietList()	Метод, що отримує список планів харчування шляхом виконання Python-скрипта та завантаження даних через менеджер даних DataManager

Продовження таблиці 3.3

1	2
private void UpdateWindow()	Метод, що оновлює елементи інтерфейсу відповідно до поточного плану харчування

3.3.4 Клас InstructionsWindow

Ціль класу полягає у реалізації вікна з інструкцією до програми. Дані та методи класу наведені у таблиці 3.4.

Таблиця 3.4 – Опис полів та методів класу InstructionsWindow

Поля та методи класу	Опис
<i>public:</i>	
public InstructionsWindow()	Конструктор, який ініціалізує вікно

3.3.5 Клас MainWindow

Ціль класу полягає у реалізації вікна головного меню. Дані та методи класу наведені у таблиці 3.5.

Таблиця 3.5 – Опис полів та методів класу MainWindow

Поля та методи класу	Опис
1	2
<i>public:</i>	
public MainWindow()	Конструктор, який ініціалізує вікно
<i>private:</i>	
private void ProfileButton_Click(object	Метод, що обробляє подію натискання кнопки відкриття профілю користувача.

sender, RoutedEventArgs e)	Ініціалізує та відображає вікно ProfileWindow
----------------------------	---

Продовження таблиці 3.5

1	2
private void DailyMealPlanButton_Click(object sender, RoutedEventArgs e)	Метод, що обробляє подію натискання кнопки для створення щоденного плану харчування. Ініціалізує та відображає вікно DailyDietWindow
private void InstructionsButton_Click(object sender, RoutedEventArgs e)	Метод, що обробляє подію натискання кнопки відкриття інструкцій. Ініціалізує та відображає вікно InstructionsWindow
private void About_Click(object sender, RoutedEventArgs e)	Метод, що обробляє подію натискання кнопки для відображення інформації про додаток. Ініціалізує та відображає вікно AboutWindow

3.3.6 Клас MessageWindow

Ціль класу полягає у реалізації вікна для обробки сповіщень про помилки та успіхи.

Дані та методи класу наведені у таблиці 3.6.

Таблиця 3.6 – Опис полів та методів класу MessageWindow

Поля та методи класу	Опис
1	2
<i>public:</i>	
public MessageWindow(string title, string mainText, string text)	Конструктор, який ініціалізує вікно

3.3.7 Клас ProfileWindow

Ціль класу полягає у реалізації вікна з профілем користувача. Дані та методи класу наведені у таблиці 3.7.

Таблиця 3.7 – Опис полів та методів класу ProfileWindow

Поля та методи класу	Опис
1	2
<i>public:</i>	
public ProfileWindow()	Конструктор, який ініціалізує вікно
public void SetAllergies(List<string> productList)	Метод, що оновлює список алергій користувача
public void SetPreferences(List<string> productList)	Метод, що оновлює список уподобань користувача
<i>private:</i>	
private User	Змінна з об'єктом користувач, що містить дані профілю, з якими користувач може взаємодіяти
private void SaveButton_Click(object sender, RoutedEventArgs e)	Метод, що обробляє подію натискання кнопки для збереження даних користувача
private bool ValidateInput()	Метод, що перевіряє правильність введених даних в полях форми
private void FillData()	Метод, що заповнює дані користувача в інтерфейс на основі отриманих даних з файлу
private void AllergiesButton_Click(object sender, RoutedEventArgs e)	Метод, що обробляє подію натискання кнопки для відкриття вікна з алергіями

Продовження таблиці 3.3

1	2
private void PreferencesButton_Click(object sender, RoutedEventArgs e)	Метод, що обробляє подію натискання кнопки для відкриття вікна з уподобаннями

3.3.8 Клас DataManager

Ціль класу полягає у реалізації роботи з базою даних та файлами. Дані та методи класу наведені у таблиці 3.8.

Таблиця 3.8 – Опис полів та методів класу DataManager

Поля та методи класу	Опис
1	2
public:	
public User LoadUserData()	Метод, що завантажує дані користувача з файлу JSON і десеріалізує їх в об'єкт User
public void SaveUserData(User user)	Метод, що зберігає дані користувача у файл JSON
public List<MealPlan> LoadMealPlanData()	Метод, що завантажує дані про план харчування з JSON файлу
public void AddMealPlanToJson(string filePath, MealPlan newMealPlan)	Метод, що додає новий план харчування у файл JSON
public void LoadFoodData()	Метод, що завантажує дані про їжу (сніданок, обід, вечерю, перекуси)
private:	
private string userDataFilePath	Змінна з шляхом до файлу JSON, де зберігаються дані користувача

Продовження таблиці 3.8

1	2
private void LoadBreakfastData()	Метод, що завантажує дані про сніданки з бази даних та записує їх у CSV
private void LoadLunchData()	Метод, що завантажує дані про обіди з бази даних та записує їх у CSV
private void LoadDinnerData()	Метод, що завантажує дані про вечері з бази даних та записує їх у CSV
private void LoadSnacksData()	Метод, що завантажує дані про перекуси з бази даних та записує їх у CSV

3.3.9 Клас Dish

Ціль класу полягає у реалізації сутності «Страва». Дані та методи класу наведені у таблиці 3.9.

Таблиця 3.9 – Опис полів та методів класу Dish

Поля та методи класу	Опис
1	2
<i>public:</i>	
public string Name { get => name; set => name = value; }	Метод для доступу до назви страви
public int CaloriesPerServing { get => calories_per_serving; set => calories_per_serving = value; }	Метод для доступу до кількості калорій на порцію
public List<string> Ingredients { get => ingredients; set => ingredients = value; }	Метод для доступу до списку інгредієнтів

Продовження таблиці 3.9

1	2
public int Servings { get => servings; set => servings = value; }	Метод для доступу до кількості порцій
private:	
private string name	Змінна з назвою страви
private int calories_per_serving	Змінна з кількістю калорій на одну порцію
private List<string> ingredients	Змінна з списком інгредієнтів страви
private int servings	Змінна з кількістю порцій у страві

3.3.10 Клас MealPlan

Ціль класу полягає у реалізації сутності «Раціон».

Дані та методи класу наведені у таблиці 3.10.

Таблиця 3.10 – Опис полів та методів класу MealPlan

Поля та методи класу	Опис
1	2
public:	
public List<Dish> DishesWithServings { get => dishes_with_servings; set => dishes_with_servings = value; }	Метод для доступу до списку страв
public int TotalCalories { get => total_calories; set => total_calories = value; }	Метод для доступу до загальної кількості калорій
private:	
private List<Dish> dishes_with_servings	Змінна з списком страв

Продовження таблиці 3.10

1	2
private int total_calories	Змінна з загальною кількістю калорій для всіх страв у плані харчування

3.3.11 Клас User

Ціль класу полягає у реалізації сутності «Користувач». Дані та методи класу наведені у таблиці 3.11.

Таблиця 3.11 – Опис полів та методів класу User

Поля та методи класу	Опис
1	2
<i>public:</i>	
public void SetTestUserData()	Метод для тестового заповнення даних користувача
public void UpdateUserData()	Метод для оновлення даних користувача і виконання розрахунків BMI, BMR і калорій
public void CalculationBMI()	Метод для обчислення індексу маси тіла
public void GetBMICategory()	Метод для визначення категорії BMI
public void CalculationBMR()	Метод для розрахунку рівня базального метаболізму в залежності від статі
public void CalculateCaloricNeeds()	Метод для розрахунку необхідної кількості калорій на основі рівня активності та категорії BMI

Продовження таблиці 3.11

1	2
<i>private:</i>	
private Name	Змінна з ім'ям користувача
private Age	Змінна з віком користувача
private Height	Змінна з зростом користувача в метрах
private Weight	Змінна з вагою користувача в кілограмах
private Gender	Змінна з статтю користувача (Чоловік / Жінка)
private BMI	Змінна з індексом маси тіла (розраховується автоматично)
private BMICategory	Змінна з категорією BMI (Недостатня вага, Нормальна вага, Зайва вага, Ожиріння)
private Allergies	Змінна з списком алергій користувача
private Preferences	Змінна з списком уподобань користувача
private ActivityLevel	Змінна з рівнем фізичної активності користувача (Сидячий спосіб життя, Помірно активний, Активний, Дуже активний)
private BRM	Змінна з рівнем базального метаболізму
private Calories	Змінна з кількістю калорій, яку користувач потребує на день

3.3.12 Клас GeneticAlgorithm

Ціль класу полягає у реалізації генетичного алгоритму для генерації оптимальних раціонів. Дані та методи класу наведені у таблиці 3.12.

Таблиця 3.12 – Опис полів та методів класу GeneticAlgorithm

Поля та методи класу	Опис
1	2
<i>public:</i>	
<code>__init__(population_size, generations, mutation_rate)</code>	Метод конструктор класу. Ініціалізує параметри генетичного алгоритму.
<code>run(breakfast_items, lunch_items, dinner_items, snacks_items, calorie_goal, user)</code>	Метод, що виконує генетичний алгоритм, створюючи початкову популяцію, виконуючи відбір, мутацію та схрещування для еволюції раціонів до досягнення калорійної мети
<i>private:</i>	
<code>population_size</code>	Змінна з розміром популяції
<code>generations</code>	Змінна з кількістю поколінь для еволюції
<code>mutation_rate</code>	Змінна з ймовірністю мутації
<code>create_individual(breakfast_items, lunch_items, dinner_items, snacks_items, user)</code>	Метод, що створює індивіда (раціон) з випадково обраними стравами. Розмір раціону варіюється від 3 до 5 страв
<code>create_dish(dishes, user)</code>	Метод, що створює страву для індивіда, з можливістю вибору улюблених страв користувача

Продовження таблиці 3.12

1	2
fitness(individual, calorie_goal)	Метод, що оцінює індивіда за відповідністю до калорійної мети
mutate(individual, breakfast_items, lunch_items, dinner_items, snacks_items)	Метод, що здійснює мутацію індивіда, змінюючи страви або кількість порцій
crossover(parent1, parent2)	Метод, що виконує схрещування двох батьків для створення нового нащадка

3.3.13 Клас NeuralNetwork

Ціль класу полягає у реалізації нейромережного алгоритму для сортування раціонів залежно від минулих виборів та відхилень користувача. Дані та методи класу наведені у таблиці 3.13.

Таблиця 3.13 – Опис полів та методів класу NeuralNetwork

Поля та методи класу	Опис
1	2
<i>public:</i>	
ingredient_encoder	Змінна з кодувальником для інгредієнтів
dish_name_encoder	Змінна з кодувальником для страв
ingredient_embedding	Змінна з шаром векторизації для інгредієнтів
dish_name_embedding	Змінна з шаром векторизації для назв страв
model	Змінна з нейронною мережею для прогнозування

Продовження таблиці 3.13

1	2
criterion	Змінна з функцією втрат для бінарної класифікації
optimizer	Змінна з оптимізатором для навчання моделі
<i>private:</i>	
<code>__init__(ingredient_vocab, dish_name_vocab, embedding_dim=16, hidden_size=64, learning_rate=0.001)</code>	Метод конструктор класу. Ініціалізує векторизацію, нейронну мережу та параметри навчання
<code>encode_mealplan(mealplan)</code>	Метод, що кодує раціон харчування у числовий формат для подачі на вхід моделі
<code>train(liked_mealplans, disliked_mealplans, epochs=100)</code>	Метод, що тренує модель на основі списків раціонів, які подобаються або не подобаються
<code>predict(mealplans)</code>	Метод, що оцінює список раціонів за їх ймовірною привабливістю
<code>sort_mealplans(mealplans)</code>	Метод, що сортує раціони за їх оцінками від найвищої до найнижчої

3.3.14 Клас DataManager

Ціль класу полягає у реалізації роботи з файлами. Дані та методи класу наведені у таблиці 3.14.

Таблиця 3.14 – Опис полів та методів класу DataManager

Поля та методи класу	Опис
1	2
<i>public:</i>	
load_user_data(file_path)	Метод для завантаження даних користувача з JSON-файлу
load_food_data(source, is_database=False, db_config=None)	Метод для завантаження даних про страви з CSV-файлу
save_dishes(dishes, file_path)	Метод для збереження списку страв у JSON-файл
load_mealplans_from_json(file_path)	Метод для завантаження
extract_vocab(*mealplan_lists)	Метод для екстракції лексикон інгредієнтів і назв страв з одного або кількох списків

3.3.15 Клас Logger

Ціль класу полягає у реалізації обробки виняткових ситуацій у вигляді ведення логів. Дані та методи класу наведені у таблиці 3.15.

Таблиця 3.15 – Опис полів та методів класу Logger

Поля та методи класу	Опис
1	2
<i>public:</i>	
log_dir	Змінна з директорією, в якій зберігається файл логів
log_file	Змінна з ім'ям файлу, куди записуються логи
info(message)	Метод, що записує інформаційне повідомлення в лог-файл

Продовження таблиці 3.15

1	2
warning(message)	Метод, що записує повідомлення з рівнем «Попередження»
error(message)	Метод, що записує повідомлення про помилку в лог-файл
<i>private:</i>	
file_path	Змінна зі шляхом до файлу, в якому зберігатимуться логи
_write_log(level, message)	Метод для запису повідомлення у файл

3.3.16 Клас Dish

Ціль класу полягає у реалізації екземплярів сутності «Страва». Дані та методи класу наведені у таблиці 3.16.

Таблиця 3.16 – Опис полів та методів класу Dish

Поля та методи класу	Опис
1	2
<i>public:</i>	
name	Змінна з назвою страви
calories_per_serving	Змінна з кількістю калорій на порцію
ingredients	Змінна з списком інгредієнтів страви
__init__(name, calories_per_serving, ingredients)	Метод конструктор класу. Ініціалізує об'єкт страви з переданими параметрами
__repr__()	Метод для повернення текстового представлення об'єкта страви у вигляді строки

3.3.17 Клас MealPlan

Ціль класу полягає у реалізації екземплярів сутності «Раціон». Це створено для зручної передачі усіх рецептів до клієнта.

Дані та методи класу наведені у таблиці 3.17.

Таблиця 3.17 – Опис полів та методів класу MealPlan

Поля та методи класу	Опис
1	2
<i>public:</i>	
dishes_with_servings	Змінна зі списком пар, де кожна пара складається з об'єкта класу Dish та кількості порцій для цієї страви
total_calories	Змінна з загальною кількістю калорій для всього плану харчування
__init__(dishes_with_servings)	Метод конструктор класу. Ініціалізує план харчування з переданими стравами та кількістю порцій, а також підраховує загальну кількість калорій
calculate_total_calories()	Метод для підрахування та повертає загальну кількість калорій у плані харчування
__repr__()	Метод для повернення текстового представлення об'єкта плану харчування у вигляді строки

3.3.18 Клас User

Ціль класу полягає у реалізації екземплярів сутності «Користувач». Дані та методи класу наведені у таблиці 3.18.

Таблиця 3.18 – Опис полів та методів класу User

Поля та методи класу	Опис
1	2
<i>public:</i>	
name	Змінна з ім'ям користувача
age	Змінна з віком користувача
height	Змінна з зростом користувача
weight	Змінна з вагою користувача
gender	Змінна з статтю користувача
bmi	Змінна з BMI
diet_category	Змінна з категорією дієти, до якої належить користувач
allergies	Змінна з алергіями користувача
preferences	Змінна з уподобаннями користувача
caloric_needs	Змінна з добовою потребою в калоріях
from_json(cls, json_data)	Метод для створення екземпляра користувача на основі даних у форматі JSON

3.4 Проектування бази даних

Концептуальна схема БД – це високорівневий опис інформаційних потреб, що лежать в основі дизайну бази даних. Вона визначає структуру всієї бази даних для спільноти користувачів, приховуючи деталі фізичного зберігання та зосереджуючись на описі типів даних, сутностей і зв'язків між ними [56].

Концептуальна схема бази даних виконує важливу роль у процесі проектування та управління базою даних. Вона потрібна для:

- забезпечення абстрактного уявлення бази даних, зосереджуючись на описі сутностей, атрибутів і зв'язків. Це дозволяє уникнути деталей

фізичного зберігання, роблячи дизайн більш зрозумілим для розробників, бізнес-аналітиків і кінцевих користувачів;

- створення єдиного уявлення про дані, яке об'єднує різні підрозділи або користувачів організації. Вона визначає чітку структуру даних, що сприяє ефективній співпраці між командами;

- завдяки концептуальній схемі легше обговорювати структуру даних між фахівцями різних рівнів, від технічних експертів до керівників, які не мають технічного досвіду;

- є базою для створення логічної та фізичної схем бази даних. Вона слугує першим етапом у деталізації структури бази даних та забезпечує правильність розробки;

- завдяки визначенню сутностей і зв'язків між ними концептуальна схема сприяє створенню бази даних, яка відповідає правилам цілісності, уникаючи дублювання або втрати даних;

- чітке визначення структури даних на концептуальному рівні дозволяє легше адаптувати базу даних до змін бізнес-потреб або зростання організації.

На рисунку 3.15 представлено створену концептуальну схему БД.

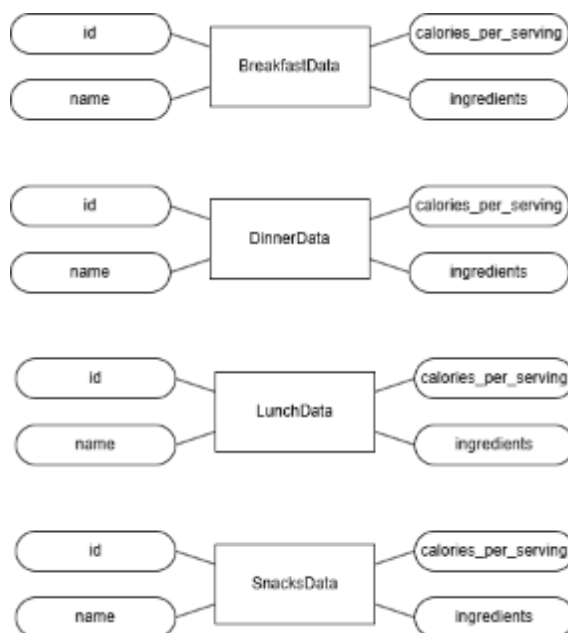


Рисунок 3.15 – Концептуальна схема БД

Сутність «BreakfastData» (id, name, calories_per_serving, ingredients) – опис сутності, а саме: назва поля, тип даних, розмір, представлено у таблиці 3.19

Таблиця 3.19 – Опис полів сутності «BreakfastData»

№	Назва поля	Тип даних	Розмір
1	id	int	4 байти
2	name	nvarchar (255)	510 байт
3	calories_per_serving	int	4 байти
4	ingredients	nvarchar (MAX)	2 * кількість символів байтів

Сутність «DinnerData» (id, name, calories_per_serving, ingredients) – опис сутності, а саме: назва поля, тип даних, розмір, представлено у таблиці 3.20

Таблиця 3.20 – Опис полів сутності «DinnerData»

№	Назва поля	Тип даних	Розмір
1	id	int	4 байти
2	name	nvarchar (255)	510 байт
3	calories_per_serving	int	4 байти
4	ingredients	nvarchar (MAX)	2 * кількість символів байтів

Сутність «LunchData» (id, name, calories_per_serving, ingredients) – опис сутності, а саме: назва поля, тип даних, розмір, представлено у таблиці 3. 21

Таблиця 3.21 – Опис полів сутності «LunchData»

№	Назва поля	Тип даних	Розмір
1	id	int	4 байти
2	name	nvarchar (255)	510 байт
3	calories_per_serving	int	4 байти
4	ingredients	nvarchar (MAX)	2 * кількість символів байтів

Сутність «SnacksData» (id, name, calories_per_serving, ingredients) – опис сутності, а саме: назва поля, тип даних, розмір, представлено у таблиці 3.22

Таблиця 3.22 – Опис полів сутності «SnacksData»

№	Назва поля	Тип даних	Розмір
1	id	int	4 байти
2	name	nvarchar (255)	510 байт
3	calories_per_serving	int	4 байти
4	ingredients	nvarchar (MAX)	2 * кількість символів байтів

Не всі дані зберігаються в базі даних. Інформація про користувача зберігається в окремому JSON файлі, так само як і дані щодо вибору та виключення раціонів. Окрім того, списки страв для сніданків, обідів, вечерь та перекусів зберігаються у CSV файлах, що використовуються для подальшої обробки у Python скрипті. Результати виконання цього скрипту потім зберігаються в JSON файлі для подальшої передачі в інтерфейс застосунку.

3.5 Основні рішення щодо розробки інтерфейсу

При розробці інтерфейсу користувача необхідно враховувати цілий ряд факторів, які забезпечують зручність, ефективність та привабливість взаємодії користувачів із програмою. Одним із ключових аспектів дизайну є правильне використання кольорів, яке не лише впливає на естетичне сприйняття, а й допомагає підвищити функціональність програми. Вдалий вибір рішень у дизайні інтерфейсу може зробити його зрозумілим і зручним для користувача, сприяючи задоволеності від використання програми.

Колір – це властивість об'єкта, яка визначається візуальним відчуттям, спричиненим дією електромагнітного випромінювання на сітківку ока в діапазоні видимого світла. Колір має значення в дизайні, оскільки він здатен впливати на емоційний стан і сприйняття користувачів [57].

Декілька прав, які допоможуть зробити інтерфейс привабливим [58]:

- гармонійна комбінація кольорів сприяє створенню візуально привабливого та професійного вигляду інтерфейсу. Інструменти, такі як Coolors або Adobe Color, можуть допомогти у створенні кольорових схем;
- мінімалістичний дизайн забезпечує зрозумілість інтерфейсу, зосереджуючи увагу користувача на важливих елементах. Треба уникати перенавантаження інтерфейсу текстом чи декоративними елементами;
- використовувати читабельні шрифти, що гармоніюють з кольоровою схемою. Добре підібраний шрифт покращує сприйняття тексту;
- розміщення елементів у порядку їхньої важливості допомагає користувачу швидко знаходити потрібну інформацію. Це досягається за допомогою кольору, розміру та розташування елементів;
- треба використовувати інтерактивні елементи, такі як анімації або візуальні підказки, щоб інформувати користувача про стан програми, наприклад, активність кнопки або стан завантаження.

Пріоритизація задоволеності користувачів у дизайні інтерфейсу не лише допомагає утримати клієнтів, а й залучає нових.

Ось як поведуться інтернет-користувачі, за даними Інституту онлайн-маркетингу [59]:

- 85% можуть покинути сайт, якщо їм не сподобається дизайн інтерфейсу;
- 83% залишать сайт, якщо їм доведеться робити занадто багато кліків, щоб знайти потрібну інформацію;
- 40% ніколи не повернуться на сайт, якщо їм було складно користуватися ним з першого разу.

Принципи хорошого інтерфейсу є однаковими для веб-сайтів, програм і сервісів.

Головна мета зручного інтерфейсу – забезпечити ідеальний користувацький досвід, щоб після встановлення програми чи використання сайту залишалися лише позитивні емоції. Якщо вдасться досягти такого ефекту, рівень конверсії буде високим [59].

Дані численних досліджень веб-аналітиків показують, що майже половина аудиторії не дає інтерфейсу другого шансу, якщо він не виправдав очікувань або виявився занадто складним для розуміння. Замість подолання перешкод для вирішення проблеми більшість людей намагатимуться знайти альтернативу [59].

Прийнято вважати, що дружній інтерфейс має бути лише простим. Насправді однієї цієї характеристики недостатньо для досягнення комерційних цілей компанії. Продукт повинен бути корисним для аудиторії, інакше ніхто не буде ним користуватися, особливо якщо на ринку існують інші рішення. Зручний інтерфейс має встановлювати міцний зв'язок із користувачами та показувати, як він допомагає вирішувати повсякденні завдання. Наприклад, замовлення продукту, відправлення листа чи читання книги [59].

Створити інтерфейс, ідеальний для вирішення конкретних завдань і водночас відповідний корпоративним цілям компанії, непросто. Його

потрібно спроектувати так, щоб користувач міг легко зрозуміти функціонал і досягти кінцевої мети без тривалого навчання.

Вибір правильного кольору для інтерфейсу має вирішальне значення, оскільки кольори впливають не лише на естетичну привабливість, але й на сприйняття, настрій і зручність використання. Кольори здатні викликати емоційні реакції, допомагати користувачам орієнтуватися в інтерфейсі та привертати увагу до важливих елементів. Наприклад, теплі відтінки можуть стимулювати активність і привертати увагу, тоді як холодні кольори сприяють відчуттю спокою та професійності.

Крім того, кольорова гама повинна відповідати брендовій ідентичності, щоб забезпечити впізнаваність і послідовність. Погано підібрані кольори можуть створити враження хаосу, ускладнити взаємодію користувачів із системою або навіть викликати дискомфорт. Наприклад, недостатній контраст може зробити текст важким для читання, особливо для людей із порушеннями зору.

Таким чином, грамотний підбір кольорів є важливою частиною дизайну, яка забезпечує зручність, комфорт і ефективність взаємодії користувачів з інтерфейсом

Психологія кольору в User Experience (UX) пов'язує зелений із природою, травою, деревами та землею, що викликає почуття спокою, умиротворення та зв'язок із природою та екологією. У різних культурах зелений має різні значення [59]. Наприклад, у фінансових додатках зелений часто символізує прибуток або позитивні зміни, тоді як у програмах, пов'язаних із здоров'ям, він асоціюється зі свіжістю, природністю та гармонією. Однак варто враховувати культурні особливості аудиторії, щоб уникнути непорозумінь.

У західних культурах зелений символізує удачу, весну, багатство, недосвідченість і заздрість (звідси й вираз «позеленіти від заздрості»). У США зелений іноді асоціюється з жадібністю та хворобами. У Мексиці зелений вважається національним кольором, що означає незалежність. У той

же час в Індонезії використання зеленого кольору заборонено. Багато південноамериканських культур використовують зелений для позначення смерті [60].

Якщо глибше розглянути психологію кольору в UX, можна зробити висновок, що зелений зазвичай використовується для підтвердження дій, таких як відповідь на дзвінок, натискання кнопки «купити» або вказівка на успіх [60].

Зелений колір може позитивно впливати на емоційний стан користувача, особливо в програмах, які потребують заспокоєння чи зосередженості. Наприклад, у медитаційних чи екологічних додатках зелений допомагає створити атмосферу довіри та релаксу.

Серед брендів, які використовують зелений у своєму брендингу та логотипах, можна назвати Animal Planet, Starbucks, Spotify, John Deere, Android і Land Rover [60].

Окрім логотипів відомих брендів, зелений часто використовується у програмах для екологічного моніторингу, фітнес-додатках або платформах, орієнтованих на здоров'я та благополуччя. Це підсилює їх асоціацію із природою та гармонією.

Зелена гама інтерфейсу була обрана для застосунку особистого помічника з раціону харчування через її асоціацію з природою, свіжістю та здоровим способом життя. Цей колір викликає почуття спокою й гармонії, що допомагає користувачам налаштуватися на позитивні зміни в харчуванні та зменшити стрес від адаптації до нового раціону.

Окрім цього, зелений підкреслює успішність дій, наприклад, виконання рекомендацій чи досягнення поставлених цілей. Він також асоціюється з екологічністю та свідомим вибором, що резонує з основними цінностями застосунку. Завдяки візуальній гармонії зелена палітра сприяє створенню привабливого та інтуїтивно зрозумілого інтерфейсу.

Саме кольорову гаму інтерфейсу можна побачити на рисунку 3.16 [61].

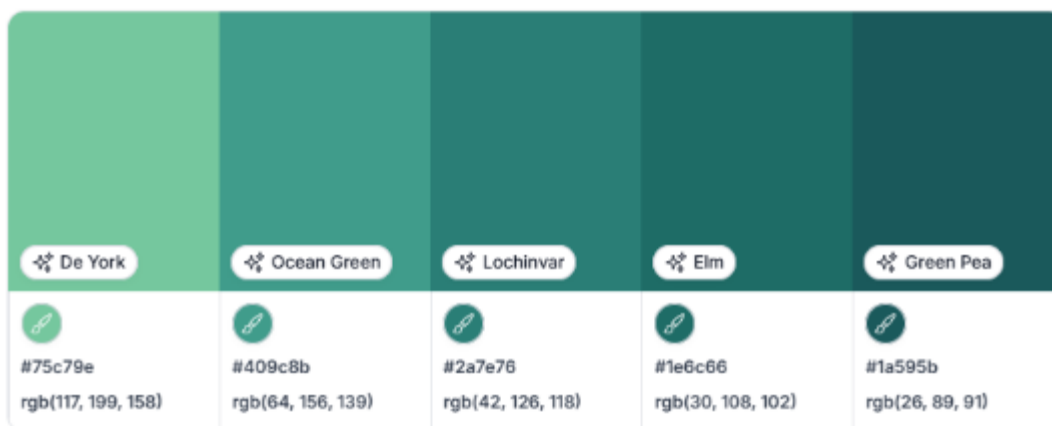


Рисунок 3.16 – Кольорова гама [61]

Ця гама кольорів на рисунку 3.16 є гарним вибором для інтерфейсу, особливо для теми здоров'я та харчування, оскільки вона має відтінки зеленого, які асоціюються з природою, свіжістю та гармонією.

- #75c79e (Color-1) є світлим, свіжим відтінком зеленого, що надає відчуття спокою та балансу;
- #409c8b (Color-2) – трохи темніший зелений, додає глибини та серйозності без втрати природного вигляду;
- #2a7e76 (Color-3) і #1e6c66 (Color-4) додають більш темних тонів, що створює баланс і контраст, роблячи інтерфейс більш професійним;
- #1a595b (Color-5) – найтемніший відтінок, який може використовуватися для акцентів або фону, щоб додати стійкості та надійності.

Такі кольори створюють спокійну та природну атмосферу, що важливо для застосунку, орієнтованого на здоров'я. Вони підкреслюють екологічність і сприяють позитивному сприйняттю контенту.

На рисунку 3.17 представлено вже саме реалізований інтерфейс головного. Фон для цього нього був розроблений окремо з використанням платформи «Canva», а стокові картинки взяти з сервісу «Freepik».

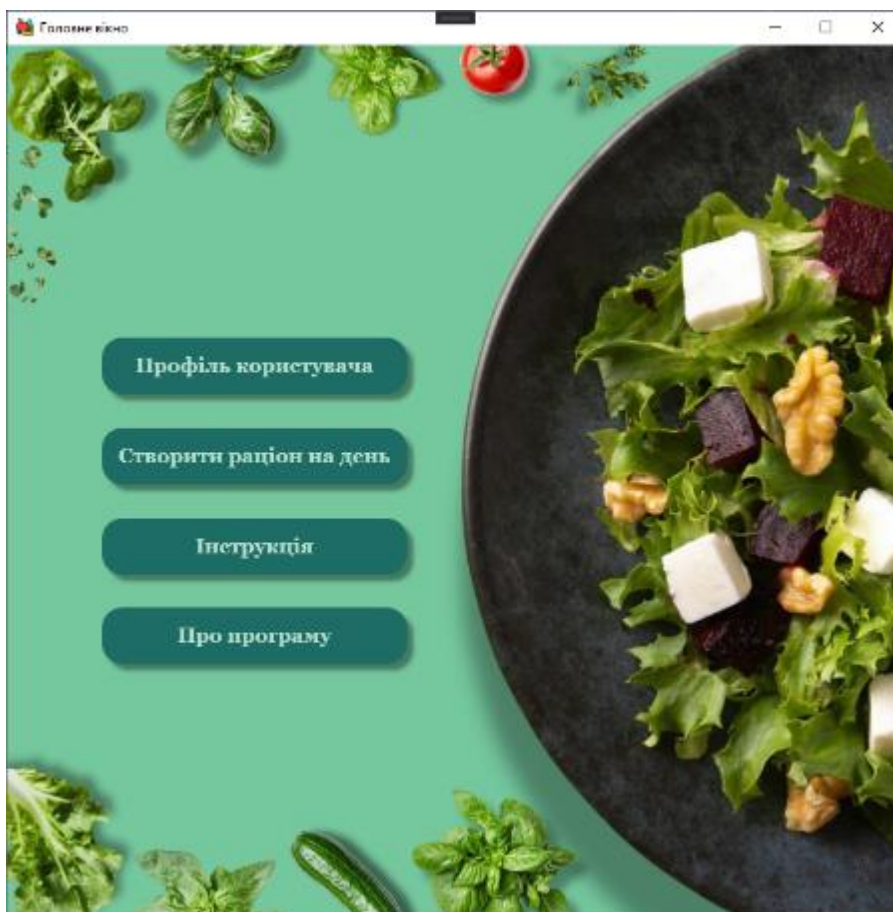


Рисунок 3.17 – Головне вікно

Для фону кнопки використано колір #1e6c66, який є насиченим темно-зеленим. Цей колір створює враження стабільності, надійності та природності, що є особливо доречним для застосунків, що стосуються здоров'я чи екології. Колір тексту #BAE2CE є світлим та контрастним, що покращує читабельність і додає свіжості.

Кути кнопки мають плавне округлення, що робить елемент менш агресивним і більш привабливим для користувача. Закруглення додає елементам дизайну м'якості та сучасного вигляду, покращуючи загальне сприйняття інтерфейсу.

Тінь під кнопкою створює ефект підняття та об'єму, що робить кнопку більш видимою та інтерактивною. Вона також допомагає відокремити елемент від фону і привертає увагу користувача.

Коли користувач наводить курсор на кнопку, колір фону змінюється на #1a595b, що є темнішим відтінком зеленого. Ця зміна сигналізує

користувачеві про можливу взаємодію, додаючи інтерактивності та зручності.

Приклад коду з встановленням стилю для елементів Button

```
<Style TargetType="Button">
    <Setter Property="FontFamily" Value="Georgia" />
    <Setter Property="Background" Value="#1e6c66" />
    <Setter Property="Foreground" Value="#BAE2CE" />
    <Setter Property="FontSize" Value="18" />
    <Setter Property="FontWeight" Value="Bold" />

    <!-- Закруглення -->
    <Setter Property="Template">
        <Setter.Value>
            <ControlTemplate TargetType="Button">
                <Border Background="{TemplateBinding Background}"
                    CornerRadius="18">
                    <Border.Effect>
                        <DropShadowEffect Color="#0F3632" Direction="315"
ShadowDepth="8" Opacity="0.4"/>
                    </Border.Effect>
                    <ContentPresenter HorizontalAlignment="Center"
VerticalAlignment="Center"/>
                </Border>
            <ControlTemplate.Triggers>
                <Trigger Property="IsMouseOver" Value="True">
                    <Setter Property="Background" Value="#1a595b"/>
                </Trigger>
            </ControlTemplate.Triggers>
        </ControlTemplate>
    </Setter.Value>
</Setter>
</Style>
```

Вже в свою чергу на рисунку 3.18 представлено інтерфейс з профілем користувачі. Фон також було створено в платформі «Canva», а стокові картинки взяти з сервісу «Freerik».

Профіль користувача

Ім'я	Вага у кілограмах
Джон	60,0
Вік	Зріст у метрах
20	2,01
Стать	Рівень активності
Чоловік	Активний
Алергії	Зберегти
Вподобання	

Рисунок 3.18 – Вікно «Профіль користувача»

Як і в інших елементах, для `ComboBox` встановлено темно-зелений колір тексту (`#0F3632`), що забезпечує хороший контраст на світлому фоні і покращує читабельність. Шрифт вибрано жирним (`Bold`) та розміром 18, що робить текст чітким і помітним.

Для кожного елемента списку `ComboBox` (`ComboBoxItem`) встановлено фон у кольорі `#409c8b`, що гармоніює з іншими елементами дизайну та підтримує свіжий і спокійний вигляд.

Як і в інших елементах, для ComboBox застосовується закруглення куточків, що надає елементу м'який і приємний вигляд, роблячи його менш агресивним і більш сучасним.

Відповідно до вимог дизайну, фон для кнопки «ToggleButton» (відкриття випадаючого списку) заповнений лінійним градієнтом з двома відтінками зеленого (#75c79e і #409c8b). Це додає динамічності та глибини елементу, а також підтримує загальну кольорову схему інтерфейсу

Коли користувач відкриває випадаючий список, фон і обрамлення елемента отримують колір #75c79e, що зберігає єдність із дизайном. Окрім того, для списку застосовано закруглення, що зберігає плавність і гармонійність елементів інтерфейсу.

Приклад коду для кнопки «ToggleButton» (відкриття випадаючого списку) для елементів ComboBox

```
<Border Background>
  <LinearGradientBrush StartPoint="0,0" EndPoint="1,1">
    <GradientStop Offset="0" Color="#75c79e"/>
    <GradientStop Offset="1" Color="#409c8b"/>
  </LinearGradientBrush>
</Border.Background>
```

Для вікон з алергіями та вподобаннями були використанні зображення для відображення різних продуктів. Ці зображення представлені на рисунку 3.19



Рисунок 3.19 – Створені зображення для продуктів

Приклад коду, що реалізує завантаження зображення у CheckBox

```
<CheckBox x:Name="CheckBox4" HorizontalAlignment="Left" Margin="7,42,0,0"
VerticalAlignment="Top" ToolTip="картопля">
    <WrapPanel>
        <Image Source="/Image/Icon/icons8-картошка-60.png"
Width="16" Height="16" Margin="5,0" />
    </WrapPanel>
</CheckBox>
```

На рисунку 3.20 можна побачити інтерфейс основного вікна застосунку «EatMe», що відображає раціон на день.



Рисунок 3.20 – Основне вікно

Колір тексту для підпису встановлено на #BAE2CE, м'який зелений відтінок, що гармоніює з іншими елементами інтерфейсу і створює відчуття свіжості та спокою, відповідно до обраної зеленої гами.

В шаблоні використовується TextBlock з властивістю TextWrapping= «Wrap», що дозволяє тексту автоматично переноситись на нові рядки, коли це необхідно, що покращує зручність читання. Також встановлено вирівнювання тексту TextAlignment= «Justify», щоб текст виглядав акуратно та рівномірно по ширині.

Приклад коду з встановленням стилю для елементів Label

```

<!-- Налаштування підпису -->
<Style TargetType="Label">
    <Setter Property="FontFamily" Value="Georgia" />
    <Setter Property="Foreground" Value="#BAE2CE" />
    <Setter Property="FontSize" Value="18" />
    <Setter Property="FontWeight" Value="Bold" />

    <Setter Property="Template">
        <Setter.Value>
            <ControlTemplate TargetType="Label">
                <TextBlock Text="{TemplateBinding Content}"
TextWrapping="Wrap" TextAlignment="Justify"/>
            </ControlTemplate>
        </Setter.Value>
    </Setter>
</Style>

```

Також для реалізації очікування виконана анімація в платформі «Canva», яка зображена на рисунку 3.21

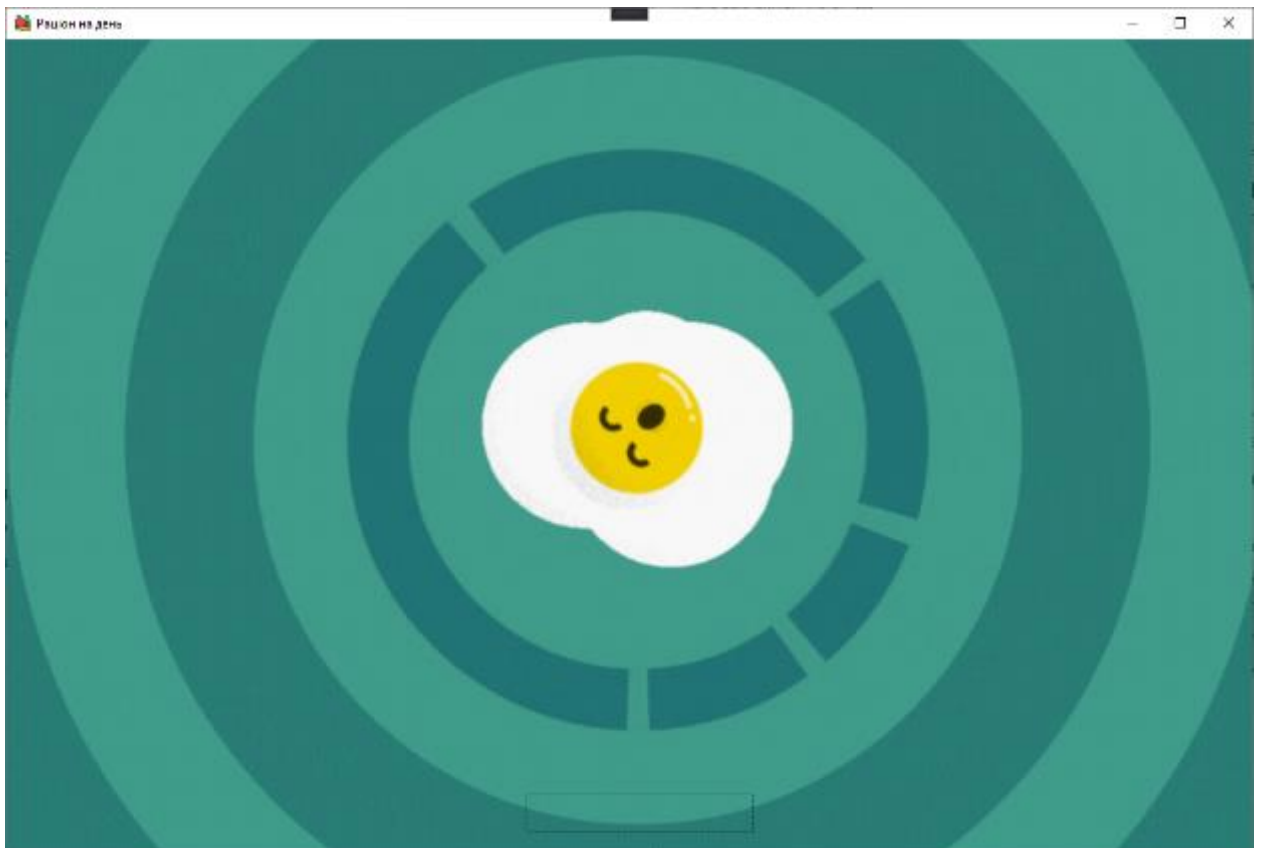


Рисунок 3.21 – Вікно з анімацією

На рисунку 3.22 є представлення інтерфейсу вікна з інструкцією. Фон також було створено в платформі «Canva», а стокові картинки взяти з сервісу «Freepik».

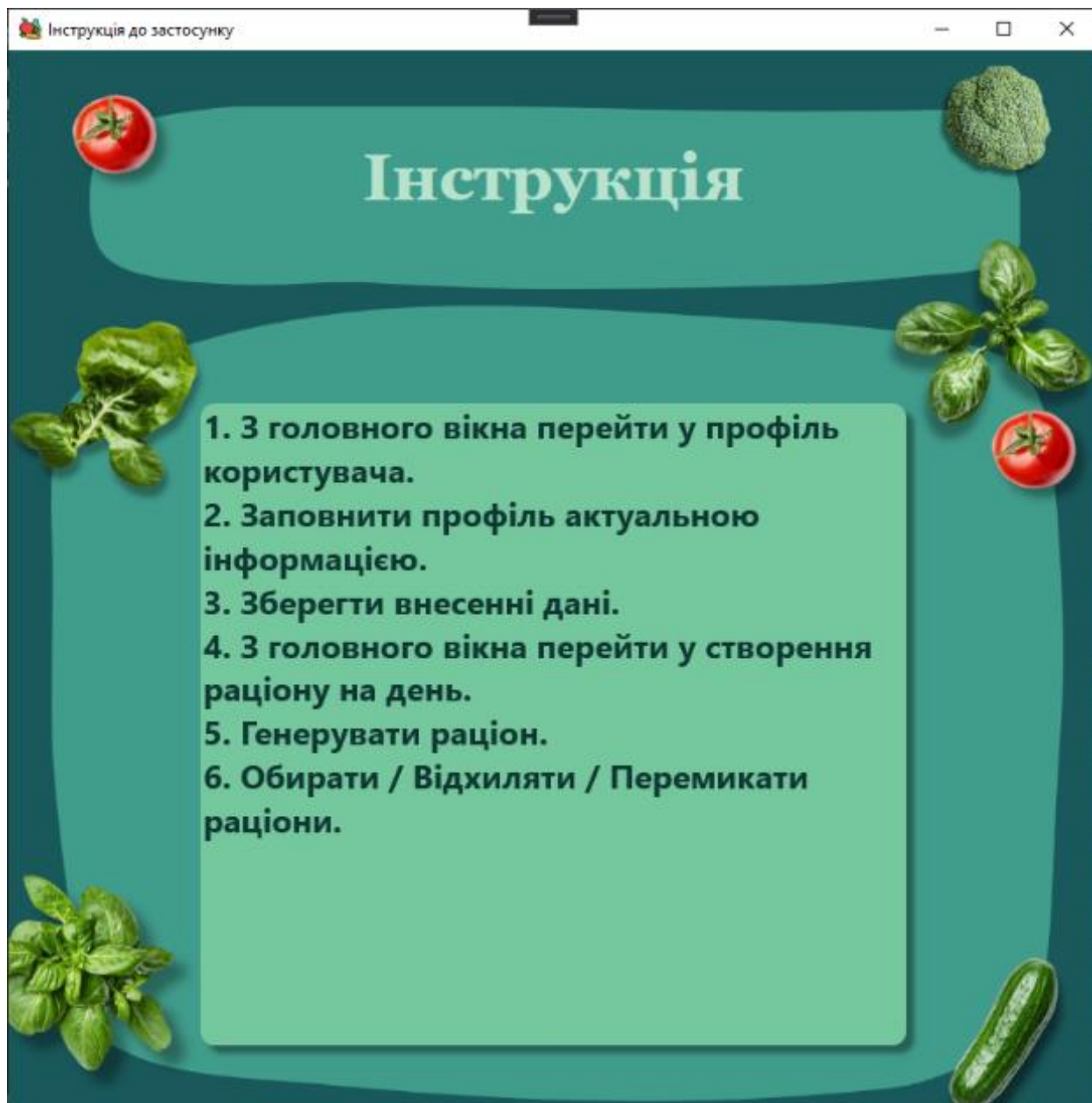


Рисунок 3.22 – Вікно з інструкцією

Колір тексту встановлений на темно-зелений (#0F3632), що забезпечує гарний контраст з фоном і покращує читабельність. Вибраний шрифт має розмір 18 та жирне (Bold) накреслення, що робить текст чітким і помітним.

Фон текстового поля має світло-зелений відтінок (#75c79e), що додає елементу свіжості та природності. Цей колір гармонійно поєднується з іншими елементами інтерфейсу, надаючи йому м'якість і теплоту.

Текстове поле має помірно округлені кути, що надає елементу м'якший, більш дружелюбний вигляд, зменшуючи його агресивність і підвищуючи привабливість. Закруглення куточків створює сучасний, гладкий дизайн.

Тінь, що додається до текстового поля, створює враження об'єму та допомагає візуально відокремити поле від фону. Вона робить текстове поле більш інтерактивним і помітним, підвищуючи його видимість.

Приклад коду з встановленням стилю для елементів TextBox

```
<!-- Налаштування текстових полів -->
<Style TargetType="TextBox">
    <Setter Property="Foreground" Value="#0F3632" />
    <Setter Property="FontSize" Value="18" />
    <Setter Property="FontWeight" Value="Bold" />
    <Setter Property="Background" Value="#75c79e" />
    <Setter Property="BorderThickness" Value="0" />
    <!-- Закруглення -->
    <Setter Property="Template">
        <Setter.Value>
            <ControlTemplate TargetType="TextBox">
                <Border Background="{TemplateBinding Background}"
                    BorderBrush="{TemplateBinding BorderBrush}"
                    BorderThickness="{TemplateBinding BorderThickness}"
                    CornerRadius="10">
                    <Border.Effect>
                        <DropShadowEffect Color="#0F3632" Direction="315"
ShadowDepth="8" Opacity="0.4"/>
                    </Border.Effect>
                    <ScrollViewer x:Name="PART_ContentHost" />
                </Border>
            </ControlTemplate>
        </Setter.Value>
    </Setter>
</Style>
```

Ще застосунок має одне довідкове вікно – це вікно з інформацією про розробників та програму. Його представлено на рисунку 3.23. Фон також було створено в платформі «Canva», а стокові картинки взяти з сервісу «Freepik».

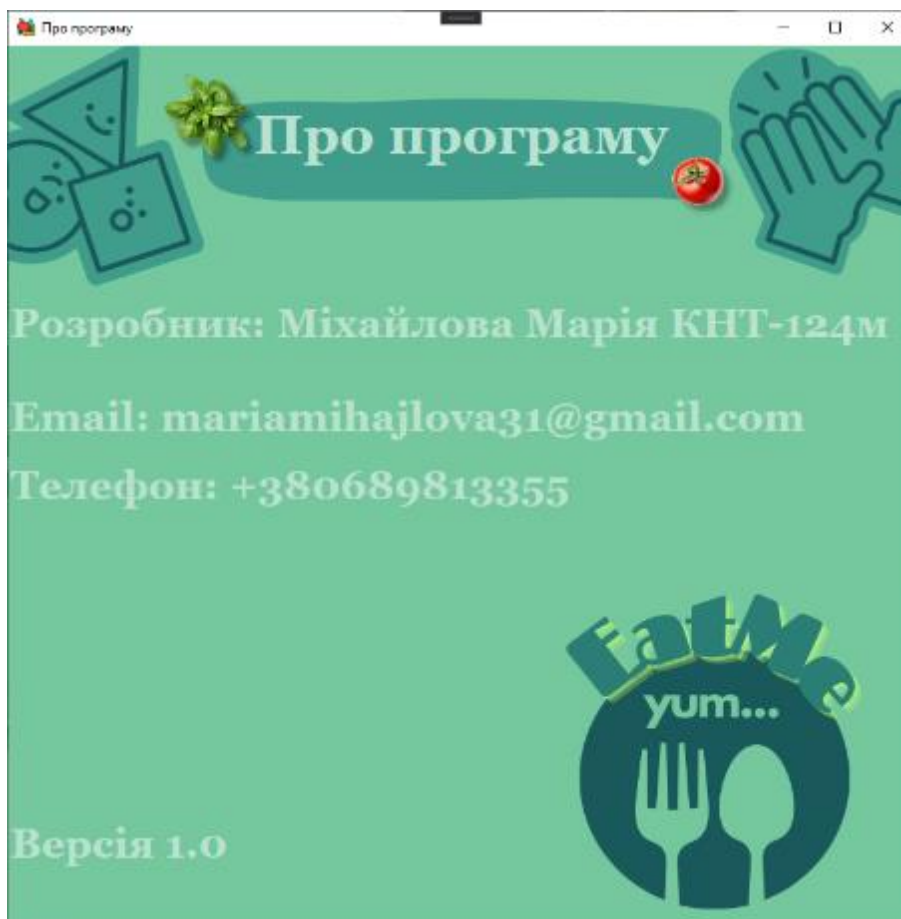


Рисунок 3.23 – Вікно з інформацією про розробників та програму

Також при виниканні виключних ситуації відкривається вікно на рисунку 3. 24.

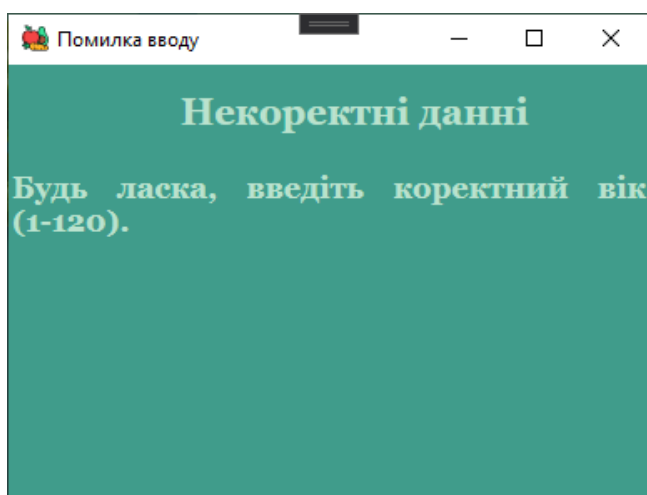


Рисунок 3.24 – Сповіщення про помилку

Також слід приділити увагу на власну іконку, яке представлено на рисунку на рисунку 3. 25



Рисунок 3. 25 – Іконка застосунку «EatMe»

Ще для застосунку було створено логотип за допомогою платформи «Canva». Його можна побачити на рисунку 3.26



Рисунок 3.26 –Логотип застосунку «EatMe»

3.6 Висновки за розділом 3

У результаті виконано такі кроки:

- створено функціональну схему програмного забезпечення, що забезпечує чітке уявлення про структуру та взаємодію основних компонентів системи, а також визначає основні етапи роботи програми;
- розроблено та детально описано ключові алгоритми, що визначають логіку роботи програмного забезпечення, включаючи алгоритми обробки даних, взаємодії з користувачем і системними процесами;

- реалізовано нейронну мережу для оптимізації вибору раціону харчування на основі минулих рішень користувача;
- створено та описано класи та інтерфейси, які визначають структуру об'єктів програми та правила їх взаємодії, що дозволяє реалізувати необхідну функціональність та забезпечити підтримку розширення в майбутньому;
- розроблено та описано базу даних проєкту, що містить структуру таблиць, зв'язки між ними та механізми доступу до даних, забезпечуючи збереження та обробку інформації в системі;
- створено та детально описано інтерфейс користувача, що включає розробку візуальних елементів, а також забезпечує зручність і доступність для кінцевих користувачів програми, надаючи зрозумілий та інтуїтивно зрозумілий дизайн;
- створено та описано класи та інтерфейси, які визначають структуру об'єктів програми та правила їх взаємодії, що дозволяє реалізувати необхідну функціональність та забезпечити підтримку розширення в майбутньому.

4 КЕРІВНИЦТВО ПРОГРАМІСТА

В даному розділі надаються детальні інструкції та рекомендації для розробників щодо роботи з програмним забезпечення. Основна мета цього розділу – забезпечити програмістів необхідними знаннями для підтримки, модифікації чи розширення проєкту, а також уникнення помилок і забезпечення відповідності вимогам проєкту. Інформація у розділі структурована таким чином, щоб бути зрозумілою навіть для нових членів команди, сприяючи ефективному введенню в проєкті та злагодженій командній роботі.

4.1 Призначення та умови застосування програми

4.1.1 Призначення програми

Даний застосунок призначений для покращення якості життя користувачів шляхом створення індивідуалізованих планів харчування, які враховують унікальні потреби, вподобання та параметри кожного користувача. Основна мета програми – допомогти користувачам підтримувати здоровий спосіб життя, збалансувати раціон харчування та досягати поставлених цілей, таких як контроль ваги, поліпшення фізичної форми чи забезпечення специфічних харчових потреб.

Програмне забезпечення надає інтуїтивний інтерфейс для налаштування особистого профілю, включаючи введення основних параметрів (вік, стать, зріст, вага), специфічних харчових уподобань (наприклад, вегетаріанство чи безглютенова дієта) та рівня фізичної активності. Використовуючи ці дані, програма автоматично генерує персоналізовані плани харчування, які враховують як поживний баланс, так і попередні вподобання користувача.

Окрім цього, застосунок забезпечує точний розрахунок добового споживання калорій на основі перевірених формул та даних про спосіб життя

користувача. Він не тільки адаптується до змін у потребах і звичках користувача, але й пропонує різноманітність у харчуванні, сприяючи довготривалому дотриманню здорового способу життя.

4.1.2 Функції програми

Основні функції, які є в програмному продукті:

- налаштування профілю користувача. Програмне забезпечення надає можливість створення та налаштування особистого профілю, де користувач може вказати такі дані, як ім'я, вік, стать, зріст, вага, а також специфічні потреби, такі як алергії, вподобання в харчуванні та рівень фізичної активності;
- створення персонального плану харчування на день. Після заповнення профілю, користувач отримає персоналізований план харчування, що містить від 3 до 5 прийомів їжі на день. План враховує не тільки індивідуальні вимоги користувача, але й забезпечує різноманітність раціону;
- персоналізація плану харчування за минулими виборами користувача. Програмне забезпечення вміє адаптувати план харчування на основі минулих виборів користувача. Це дозволяє враховувати уподобання та звички в харчуванні, що були зроблені раніше, і таким чином робить план більш індивідуальним;
- розрахунок денного споживання калорій. Програмне забезпечення має можливість автоматично розраховувати денну норму калорій на основі особистих даних користувача, таких як вага, вік, рівень активності та інші параметри. Це включає використання формул для розрахунку базового метаболізму (BMR) та коригування потреби в калоріях залежно від рівня фізичної активності. Таким чином, користувач отримає точну кількість калорій, необхідних для підтримки здоров'я, схуднення або набору ваги.

4.1.3 Умови застосування програми

Для коректної роботи програми пристрій повинен відповідати наступним технічним та програмним вимогам:

- оперативна пам'ять – мінімум 8 ГБ для забезпечення плавної роботи програми.;
- місце на диску – мінімум 1500 МБ вільного простору для встановлення та збереження даних програми;
- операційна система – Windows 10 або новіша версія для підтримки всіх функцій та оновлень;
- обсяг жорсткого диска – не менше 256 ГБ для зберігання додаткових файлів;
- процесор – багатоядерний, з тактовою частотою не менше 1.5 ГГц, для обробки даних та швидкої взаємодії з користувачем

Для розробки та модифікації програми доцільно використовувати Visual Studio 2019 разом із платформою .NET Framework 4.7.2, а також Python версії 3.10.

4.2 Характеристика програми

Застосунок розроблено з використанням мов програмування C# і Python у середовищі Visual Studio 2019. Для створення графічного інтерфейсу використано WPF, а базовою платформою виступає .NET Framework 4.7.2. Код Python працює на версії 3.10.

Основний екран програмного продукту: «EatMe» містить кнопки, які забезпечують зручну навігацію між різними розділами застосунку.

4.2.1 Структура програми

Для запуску програми необхідно активувати її графічну підсистему шляхом відкриття виконуваного файлу *.exe. Після запуску користувач автоматично перенаправляється до головного вікна.

Архітектура програми включає такі основні компоненти: бібліотечні файли, вихідний код на мовах *cs і *py, а також файли розмітки *xaml. Детальний опис структури представлено на рисунку 4.1.

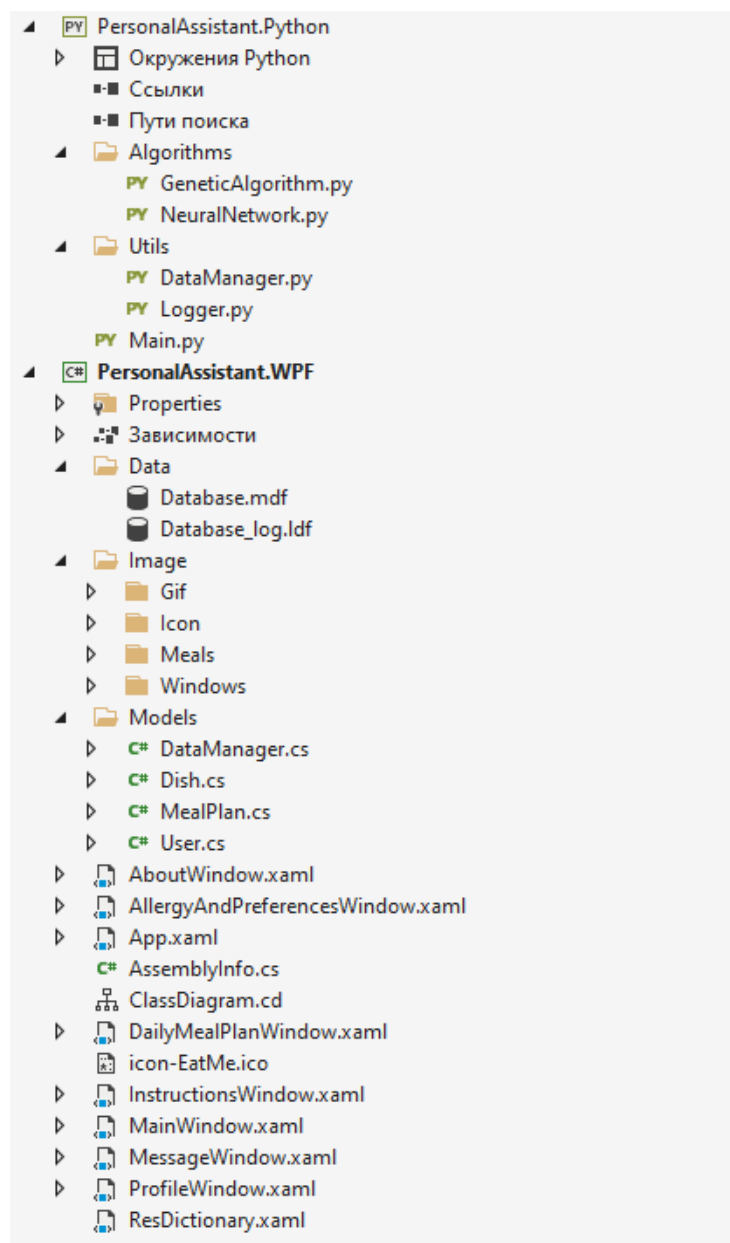


Рисунок 4.1 – Структура програми

4.3 Звертання до програми

Програма підтримує наступні способи взаємодії з користувачем:

- використання лівої кнопки миші для роботи з кнопками та текстовими полями;
- введення інформації за допомогою клавіатури.

4.4 Вхідні та вихідні дані

4.4.1 Вхідні дані

Програма працює з такими вхідними даними:

- сигнали від користувача про взаємодію з мишею чи клавіатурою;
- інформація про обрані продуктів за допомогою CheckBox, що відображують вибір алергій чи харчових вподобань;
- файли форматів *json і *csv;
- дані, які збережені у базі даних;
- мультимедійні ресурси, такі як зображення, GIF-файли та іконки програми.

4.4.2 Вихідні дані

Програма видає такі вихідні дані як:

- інформація про раціони;
- спливаючі підказки;
- інформація про програму;
- інструкція до програми;
- інформація про профіль.

4.5 Повідомлення

У програмі передбачено такі типи сповіщень:

- «Помилка запуску процесу» – вказує на те, що сталася помилка під час виконання Python скрипту;
- «Помилка оновлення вікна» – вказує на те, що сталася помилка під час оновлення вікна з раціонами на день;
- «Помилка при генерації плану» – вказує на те, що сталася помилка під час генерації раціонів на день;
- «Помилка при завантаженні даних» – вказує на те, що сталася помилка під час завантаження даних про страви з БД у файли;
- «Помилка під час відкриття профілю» – вказує на те, що сталася помилка під час відкриття вікна з профілем;
- «Помилка під час відкриття створення раціону на день» – вказує на те, що сталася помилка під час відкриття вікна зі створенням раціону на день;
- «Помилка під час відкриття інструкцій» – вказує на те, що сталася помилка під час відкриття вікна з інструкцією до програми;
- «Помилка під час відкриття інформації про програму» – вказує на те, що сталася помилка під час відкриття вікна з інформацією про програму;
- «Помилка завантаження даних користувача» – вказує на те, що сталася помилка під час завантаження даних користувача з файлу користувача;
- «Помилка збереження даних користувача» – вказує на те, що сталася помилка під час збереження даних користувача у файл;
- «Помилка завантаження плану харчування» – вказує на те, що сталася помилка під час завантаження даних раціонів на день;
- «Помилка додавання плану харчування» – вказує на те, що сталася помилка під час додавання нового плану харчування у файл;
- «Помилка завантаження даних їжі» – вказує на те, що сталася помилка під час завантаження даних про страви з БД;

- «Помилка під час підключення або виконання запиту» – вказує на те, що сталася помилка під час виконання запиту до БД;
- «Помилка при завантаженні даних користувача» – вказує на те, що сталася помилка під час завантаження даних користувача у вікно профілю;
- «Дані успішно збережено!» – вказує на те, що данні користувача успішно збережені;
- «Помилка при збереженні даних» – вказує на те, що сталася помилка під час збереження даних користувача;
- «Некоректні данні» – вказує на те, що введені некоректні данні у профілі користувача;
- «Помилка при заповненні даних» – вказує на те, що сталася помилка під час заповнення вікна з профілем користувача;
- «Помилка при відкритті вікна алергій» – вказує на те, що сталася помилка під час відкриття вікна з алергією;
- «Помилка при відкритті вікна вподобань» – вказує на те, що сталася помилка під час відкриття вікна з вподобаннями;
- «Помилка оновлення даних» – вказує на те, що оновлення даних користувача неуспішне.

5 КЕРІВНИЦТВО ОПЕРАТОРА

Даний розділ містить основну інформацію, необхідну для успішного використання програмного забезпечення. У ньому наведено рекомендації щодо підготовки до роботи з програмою, описано інтерфейс, механізми взаємодії користувача з системою, а також порядок виконання основних операцій. Інструкції представлені таким чином, щоб забезпечити простоту та зручність у використанні навіть для користувачів із мінімальним досвідом роботи з подібними програмами.

5.1 Призначення програми

Цей застосунок створено для покращення якості життя користувачів шляхом розробки персоналізованих планів харчування, які враховують їхні унікальні потреби та вподобання. Програма допомагає підтримувати здоровий спосіб життя, збалансувати раціон та досягати індивідуальних цілей, таких як контроль ваги чи забезпечення специфічних харчових потреб.

Інтуїтивний інтерфейс застосунку дозволяє налаштувати особистий профіль, включаючи параметри користувача, харчові вподобання та рівень активності. На основі введених даних програма автоматично формує раціон із збалансованим поживним складом, забезпечуючи різноманітність та адаптацію до змін у звичках користувача.

5.2 Умови виконання програми

5.2.1 Апаратні вимоги програми

Щоб програма працювала стабільно та ефективно, пристрій повинен відповідати таким системним і програмним вимогам:

- оперативна пам'ять щонайменше 8 ГБ для забезпечення безперебійного функціонування;

- вільний простір на диску не менше 1500 МБ для встановлення програми та зберігання її даних;
- операційна система – Windows 10 або пізніша версія для доступу до всіх функцій і оновлень;
- місткість жорсткого диска потрібна мінімум 256 ГБ для розміщення додаткових файлів.

5.2.2 Вимоги до операторів

Для ефективного використання програми користувачеві необхідно:

- володіти українською мовою;
- мати базові навички роботи з комп'ютерною мишею та клавіатурою;
- ознайомитися з керівництвом користувача для правильного використання програми.

5.3 Виконання програми

5.3.1 Запуск програми

Для запуску програми необхідно двічі клацнути лівою кнопкою миші на виконуваному файлі EatMe.exe або його ярлику.

5.3.2 Виконання роботи з програмою

Після запуску застосунку «EatMe» відкривається головне вікно, яке представлено на рисунку 5.1

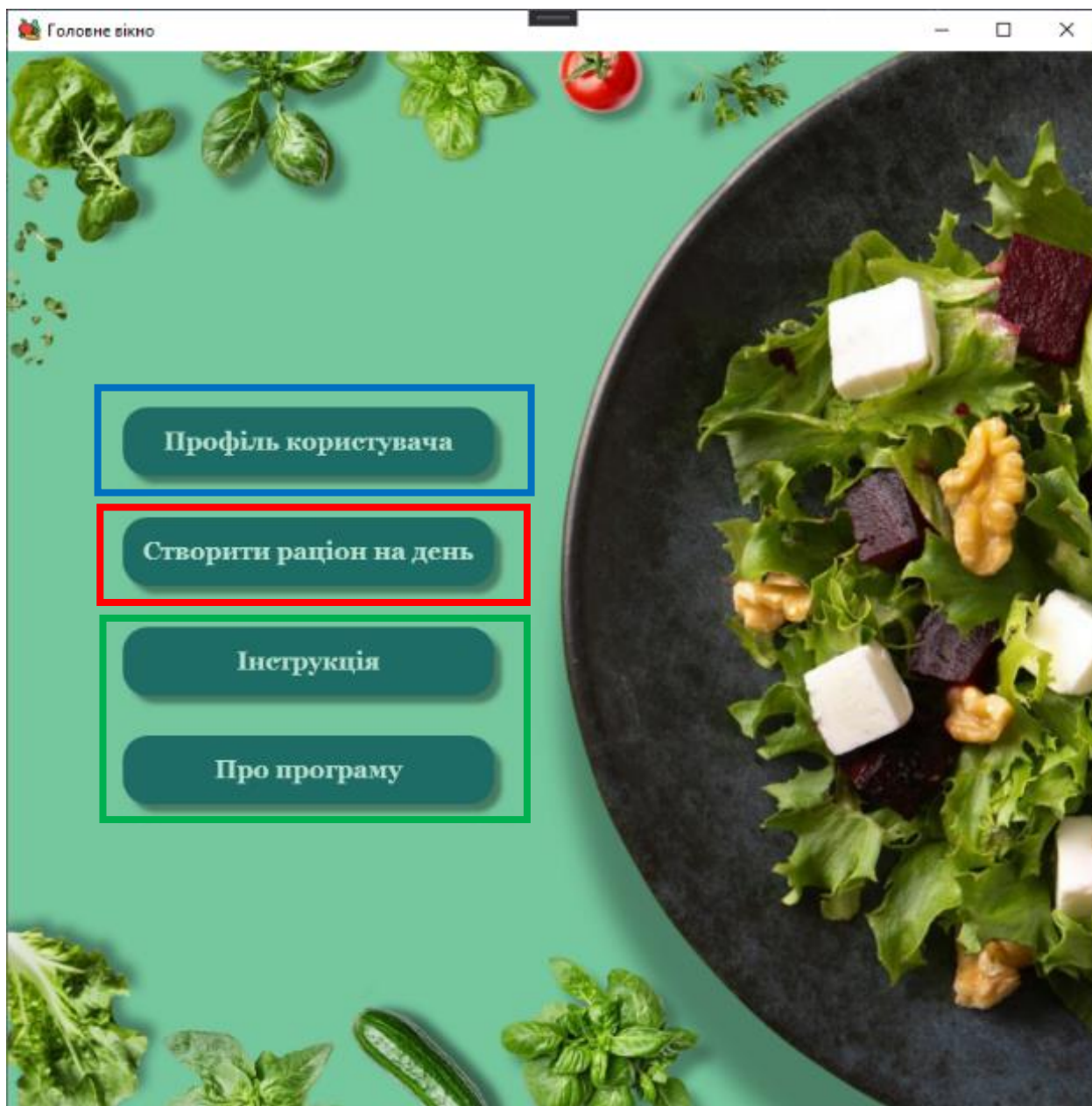


Рисунок 5.1 – Головне вікно

На рисунку 5.1 у синьому прямокутнику виділено кнопку «Профіль користувача» за допомогою неї можна відкрити вікно з профілем користувача, що відображено на рисунку 5.2.

За допомогою цього вікна можна відредагувати інформацію про користувача. Ця інформація є базовою для створення раціону на день, бо вона використовується під час розрахунку у формулах.

Профіль користувача

Ім'я	Вага у кілограмах
Джон	60,0
Вік	Зріст у метрах
20	2,01
Стать	Рівень активності
Чоловік	Активний
Алергії	Зберегти
Вподобання	

Рисунок 5.2 – Вікно «Профіль користувача»

На рисунку 5.2 у червоному прямокутнику відображено поля для вводу інформації з клавіатури. Тут треба вказати таку інформацію як:

- ім'я;
- вага користувача в кілограмах (більше 0 та через крапку).

Наприклад: 70,7;

- вага користувача в метрах (більше 0 та через крапку). Наприклад: 1,70;

- вік (від 1 до 120). Також треба вказувати повні роки. Наприклад: 18.

Також для розуміння про те, що сталась помилка створено вікно з сповіщенням про помилку. Наприклад, коли введено некоректні дані у поле з віком користувача. Саме це вікно показано на рисунку 5.3

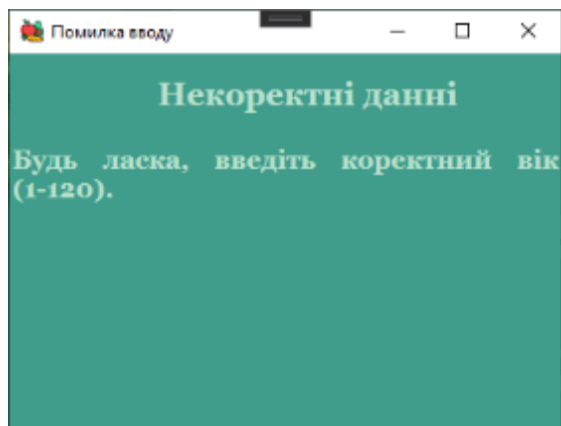


Рисунок 5.3– Сповіщення про помилку

На рисунку 5.2 у синьому прямокутнику відображено поля для вводу інформації з спливаючого списку. Тут треба вказати таку інформацію як:

- стать (чоловік або жінка);
- рівень активності (сидячий спосіб життя, помірно активний, активний, дуже активний).

На рисунку 5.2 у зеленому прямокутнику відображено кнопки для вибору алергій та вподобань. Ці вікна мають однаковий вигляд. Різниця в заголовку вікна. На рисунку 5.4 показано приклад вікна з вподобаннями.

В цих вікна автоматично зберігається інформація про їжу обрані раніше продукти. Також є підказки при наведенні на картинку для кращого розуміння, якщо користувач не розуміє графічне зображення продукту.

Також слід сказати, що види схожих продуктів об'єднанні в окремі групи, а саме:

- овочі;
- фрукти;
- м'ясо та риба;
- молочні продукти;

- ягоди;
- спеції, цукати та соуси;
- інше.



Рисунок 5.4 – Вікно з вподобаннями

На рисунку 5.2 у жовтому прямокутнику відображено кнопку для збереження інформації, яку ввів користувач.

Після оновлення інформації про користувача, якщо ця дія була, можна перейти до створення раціону. Це можна зробити натиснувши на кнопку «Створити раціон на день», що зображена на рисунку 5.1 у червоному прямокутнику. Після натиснення на кнопку відкривається вікно з раціонами, яке представлено на рисунку 5.5.

На рисунку 5.5 у синьому прямокутнику виділено основне поле, де відображається інформація про раціон (3-5 страв на день):

- загальну кількість калорій;
- інформація про сніданок;
- інформація про обід;
- інформація про вечерю;
- інформація про перекуси один та два, якщо вони присутні.

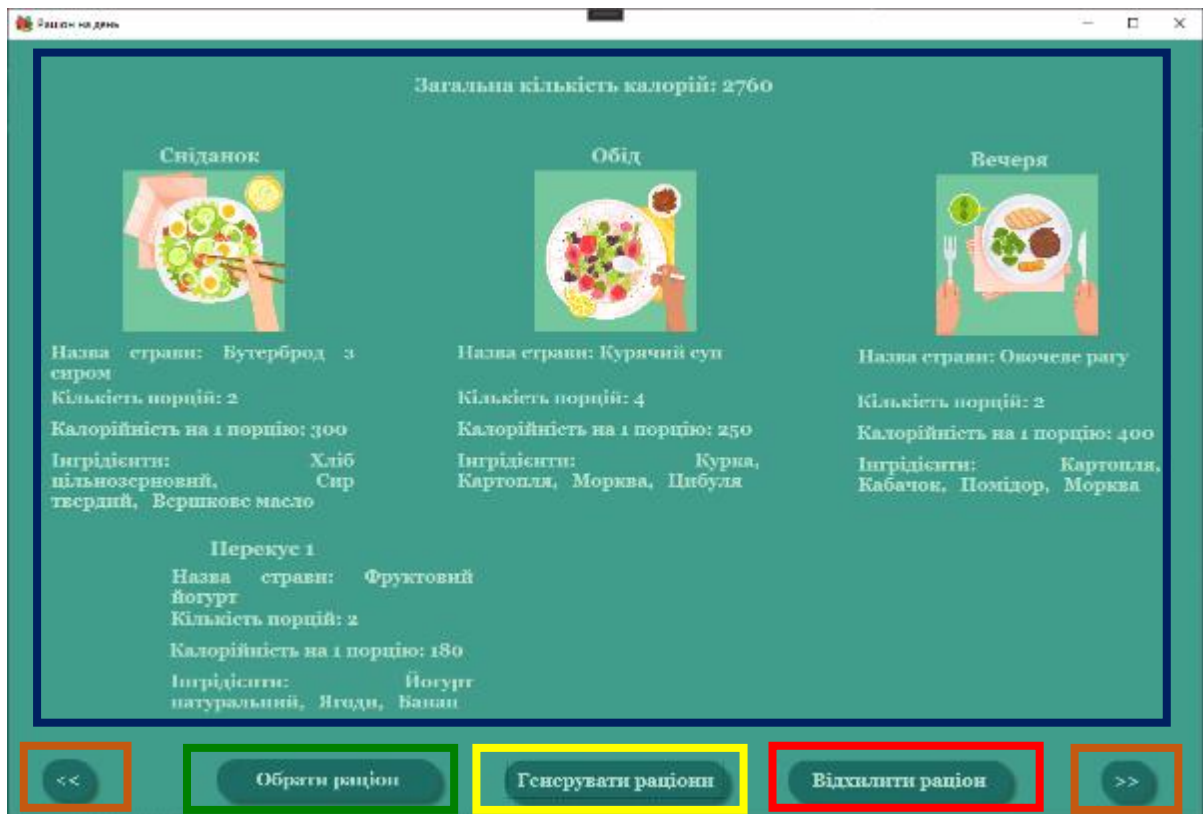


Рисунок 5.5 – Основне вікно

На рисунку 5.5 у жовтому прямокутнику виділено кнопку «Генерувати раціон», яка допомагає змінити варіанти раціонів на день, враховуючи минулі вибори користувача. Під час генерації відбувається очікування, тому для відображення цього процесу було створено анімацію, яка представлена на рисунку 5.6



Рисунок 5.6 – Вікно з анімацією

Після закінчення генерації користувач буде автоматично повернений до основного вікна за раціонами. На рисунку 5.5 у зеленому прямокутнику виділено кнопку «Обрати раціон». При натисканні на неї користувач обирає цей раціон – це означає, що продукти, які входять в цей раціон йому сподобались. В наступних генераціях раціону будуть частіше показувати користувачу страви з цими продуктами.

На рисунку 5.6 у червоному прямокутнику виділено кнопку «Відхилити раціон». При натисканні на неї користувач раціон пропадає зі списку варіантів можливих раціонів на день – це означає що продукти, які входять в цей раціон йому не сподобались. В наступних генераціях раціону будуть рідше показувати користувачу страви з цими продуктами.

На рисунку 5.6 у помаранчевому прямокутнику виділено кнопки для гортання раціонів, щоб подивитись всі можливі варіанти раціонів і обрати найкращий.

На рисунку 5.1 у зеленому прямокутнику виділено кнопки: «Інструкція» та «Про програму». Натискання на ці кнопки відкривають відповідні довідникові вікна, які відображено на рисунку 5.7.

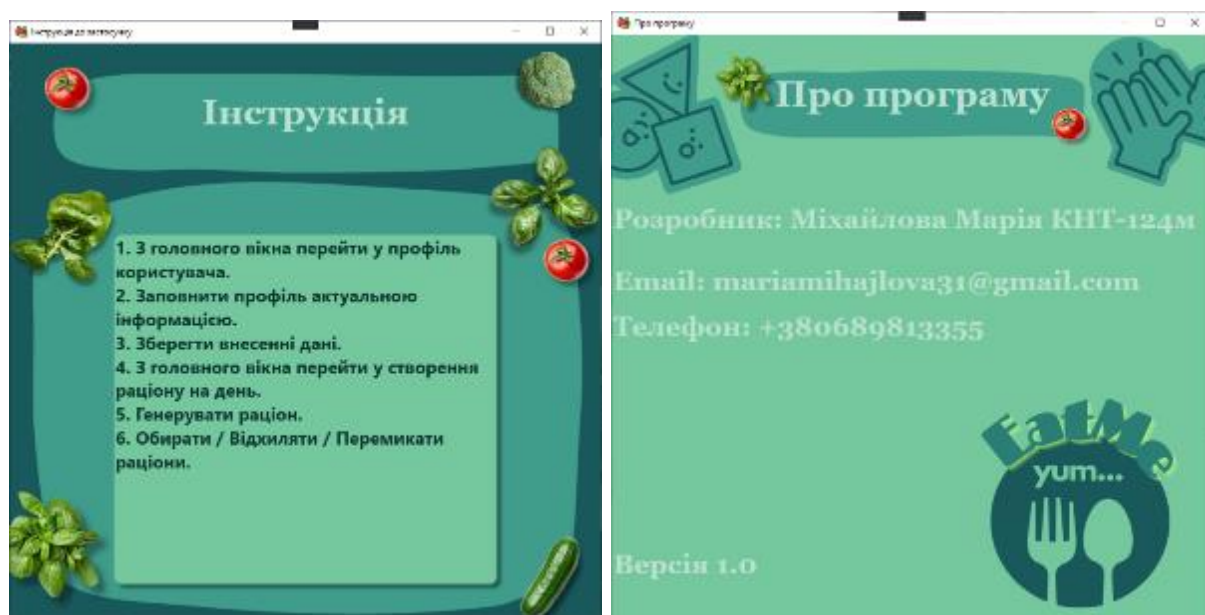


Рисунок 5.7 – Довідникові вікна

5.4 Повідомлення оператору

У програмі передбачено такі типи сповіщень:

- «Помилка запуску процесу» – вказує на те, що сталася помилка під час виконання Python скрипту;
- «Помилка оновлення вікна» – вказує на те, що сталася помилка під час оновлення вікна з раціонами на день;
- «Помилка при генерації плану» – вказує на те, що сталася помилка під час генерації раціонів на день;
- «Помилка при завантаженні даних» – вказує на те, що сталася помилка під час завантаження даних про страви з БД у файли;
- «Помилка під час відкриття профілю» – вказує на те, що сталася помилка під час відкриття вікна з профілем;
- «Помилка під час відкриття створення раціону на день» – вказує на те, що сталася помилка під час відкриття вікна зі створенням раціону на день;
- «Помилка під час відкриття інструкцій» – вказує на те, що сталася помилка під час відкриття вікна з інструкцією до програми;
- «Помилка під час відкриття інформації про програму» – вказує на те, що сталася помилка під час відкриття вікна з інформацією про програму;
- «Помилка завантаження даних користувача» – вказує на те, що сталася помилка під час завантаження даних користувача з файлу користувача;
- «Помилка збереження даних користувача» – вказує на те, що сталася помилка під час збереження даних користувача у файл;
- «Помилка завантаження плану харчування» – вказує на те, що сталася помилка під час завантаження даних раціонів на день;
- «Помилка додавання плану харчування» – вказує на те, що сталася помилка під час додавання нового плану харчування у файл;
- «Помилка завантаження даних їжі» – вказує на те, що сталася помилка під час завантаження даних про страви з БД;

- «Помилка під час підключення або виконання запиту» – вказує на те, що сталася помилка під час виконання запиту до БД;
- «Помилка при завантаженні даних користувача» – вказує на те, що сталася помилка під час завантаження даних користувача у вікно профілю;
- «Дані успішно збережено!» – вказує на те, що данні користувача успішно збережені;
- «Помилка при збереженні даних» – вказує на те, що сталася помилка під час збереження даних користувача;
- «Некоректні данні» – вказує на те, що введені некоректні данні у профілі користувача;
- «Помилка при заповненні даних» – вказує на те, що сталася помилка під час заповнення вікна з профілем користувача;
- «Помилка при відкритті вікна алергій» – вказує на те, що сталася помилка під час відкриття вікна з алергією;
- «Помилка при відкритті вікна вподобань» – вказує на те, що сталася помилка під час відкриття вікна з вподобаннями;
- «Помилка оновлення даних» – вказує на те, що оновлення даних користувача неуспішне.

6 ТЕСТУВАННЯ І РЕЗУЛЬТАТИ

6.1 Опис методики проведення тестування

Процес тестування здійснюватиметься відповідно до наступного сценарію:

а) налаштування профілю:

- 1) збереження з коректним ім'ям;
- 2) збереження з некоректним ім'ям;
- 3) збереження з коректним віком;
- 4) збереження з некоректним віком;
- 5) збереження з коректним зростом;
- 6) збереження з некоректним зростом;
- 7) збереження з коректною вагою;
- 8) збереження з некоректною вагою;
- 9) збереження з коректною статтю;
- 10) збереження з некоректною статтю;
- 11) збереження з коректним рівнем активності;
- 12) збереження з некоректним рівнем активності;
- 13) вибір алергій;
- 14) вибір вподобань;
- 15) завантаження даних користувача;

б) створення раціону на день:

- 1) генерація варіантів раціонів на день;
- 2) вибір раціону на день;
- 3) відхилення раціону на день;
- 4) перемикання поточного раціону;

в) головне вікно:

- 1) перехід у вікно «Створити раціон на день»;
- 2) перехід у вікно «Профіль користувача»;
- 3) перехід у вікно «Інструкція»;

4) перехід у вікно «Про програму»;

Для проведення тестування буде використано комп'ютерну систему з такими характеристиками:

- оперативна пам'ять на 16 ГБ;
- встановлена операційна система Windows 10;
- накопичувач із ємністю 1 ТБ.

6.2 Результати використання розробленого програмного забезпечення

6.2.1 Налаштування профілю користувача

У таблиці 6.1 представлено результати перевірки функціональності вікна «Профіль користувача».

Таблиця 6.1 – Тестування зміни даних користувача

Назва	Результат	Успішність
1	2	3
Збереження коректним ім'ям	Збереження успішне	+
Збереження з некоректним ім'ям	Збереження неуспішне	+
Збереження з коректним віком;	Збереження успішне	+
Збереження з некоректним віком	Збереження неуспішне	+
Збереження з коректним зростом	Збереження успішне	+
Збереження з некоректним зростом	Збереження неуспішне	+
Збереження з коректною вагою	Збереження успішне	+
Збереження з некоректною вагою	Збереження неуспішне	+
Збереження коректною статтю	Збереження успішне	+
Збереження з некоректною статтю	Збереження неуспішне	+
Збереження з коректним рівнем активності	Збереження успішне	+

Продовження таблиці 6.1

1	2	3
Збереження з некоректним рівнем активності	Збереження неуспішне	+
Вибір алергій	Вибрані алергії успішно збережені	+
Вибір вподобань	Вибрані вподобання успішно збережені	+
Завантаження даних користувача	Данні коректно завантажуються	+

6.2.2 Генерація раціону на день

У таблиці 6.2 представлено результати перевірки функціональності вікна «Раціон на день».

Таблиця 6.2 – Тестування генерації раціону

Назва	Результат	Успішність
Генерація варіантів раціонів на день	Варіанти згенеровані	+
Вибір раціону на день	Раціон успішно обрано	+
Відхилення раціону на день	Раціон успішно відхилено	+
Перемикання поточного раціону	Раціон успішно перемкнуто	+

6.2.3 Головне вікно

У таблиці 6.3 представлено результати перевірки функціональності головного вікна.

Таблиця 6.3 – Тестування головного вікна

Назва	Результат	Успішність
Перехід у вікно «Створити раціон на день»	Вікно успішно відкрито	+
Перехід у вікно «Профіль користувача»	Вікно успішно відкрито	+
Перехід у вікно «Інструкція»	Вікно успішно відкрито	+
Перехід у вікно «Про програму»	Вікно успішно відкрито	+

6.3 Результати тестування генетичного й нейромережевого алгоритмів

У результаті тестування роботи генетичного алгоритму було досягнуто задовільних результатів. Алгоритм демонструє високу ефективність у знаходженні раціонів, що максимально наближені до заданого цільового рівня калорійності. Розбіжність між калорійністю оптимального раціону та цільовим значенням знаходиться у межах від 1 до 10 ккал, що вказує на його точність і адаптивність. Візуалізацію цих результатів представлено на графіку на рисунку 6.1, який відображає динаміку збіжності значень під час роботи алгоритму.

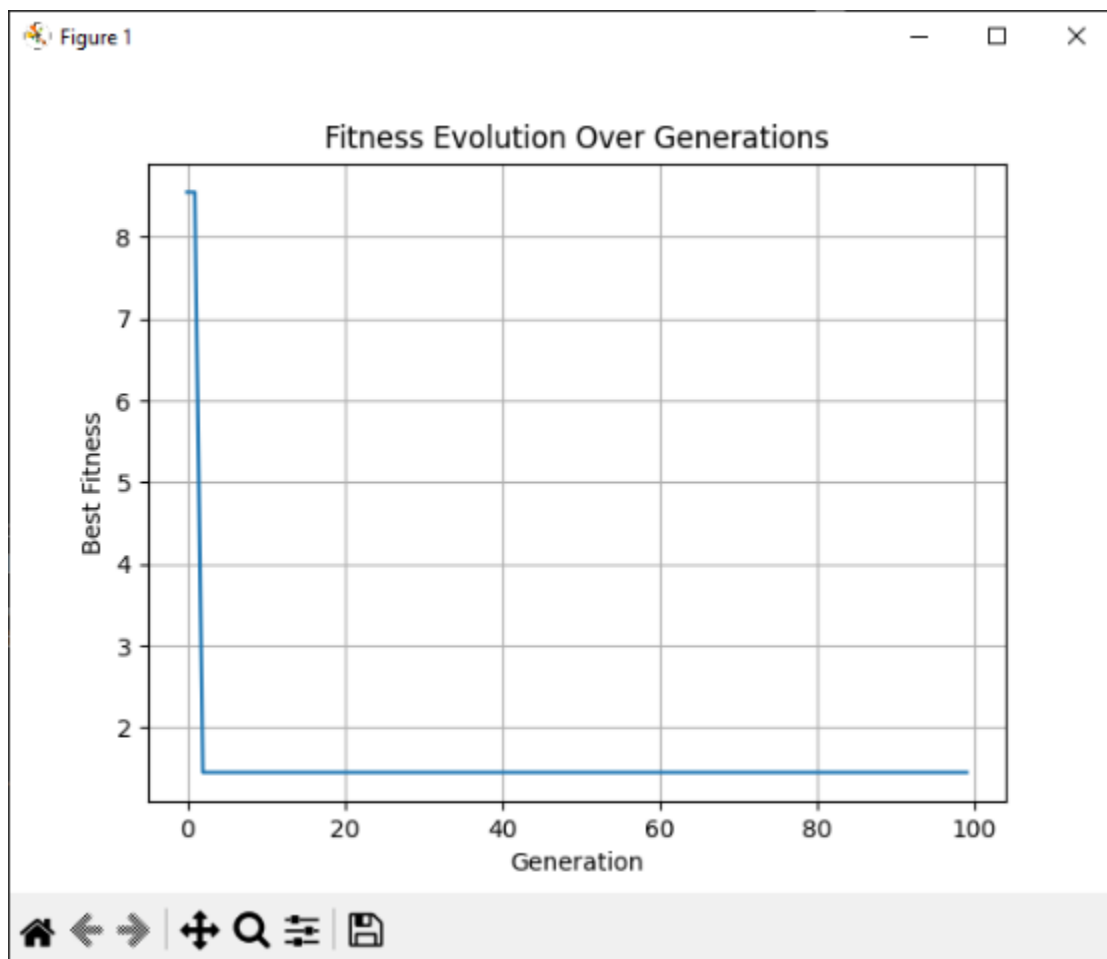


Рисунок 6.1 – Графік різниці між найкращим результатом покоління та цільовим значенням калорійності

Як видно з графіка на рисунку 6.2, результати тестування нейромережевого алгоритму продемонстрували його здатність досягати задовільної точності класифікації при навчанні на збережених даних про вибори та відхилення раціонів користувачем. Це забезпечує ефективність моделі для подальшого сортування раціонів. При цьому точність класифікації поступово зростає зі збільшенням обсягу релевантних даних, що свідчить про здатність алгоритму до навчання на основі накопичуваної інформації.

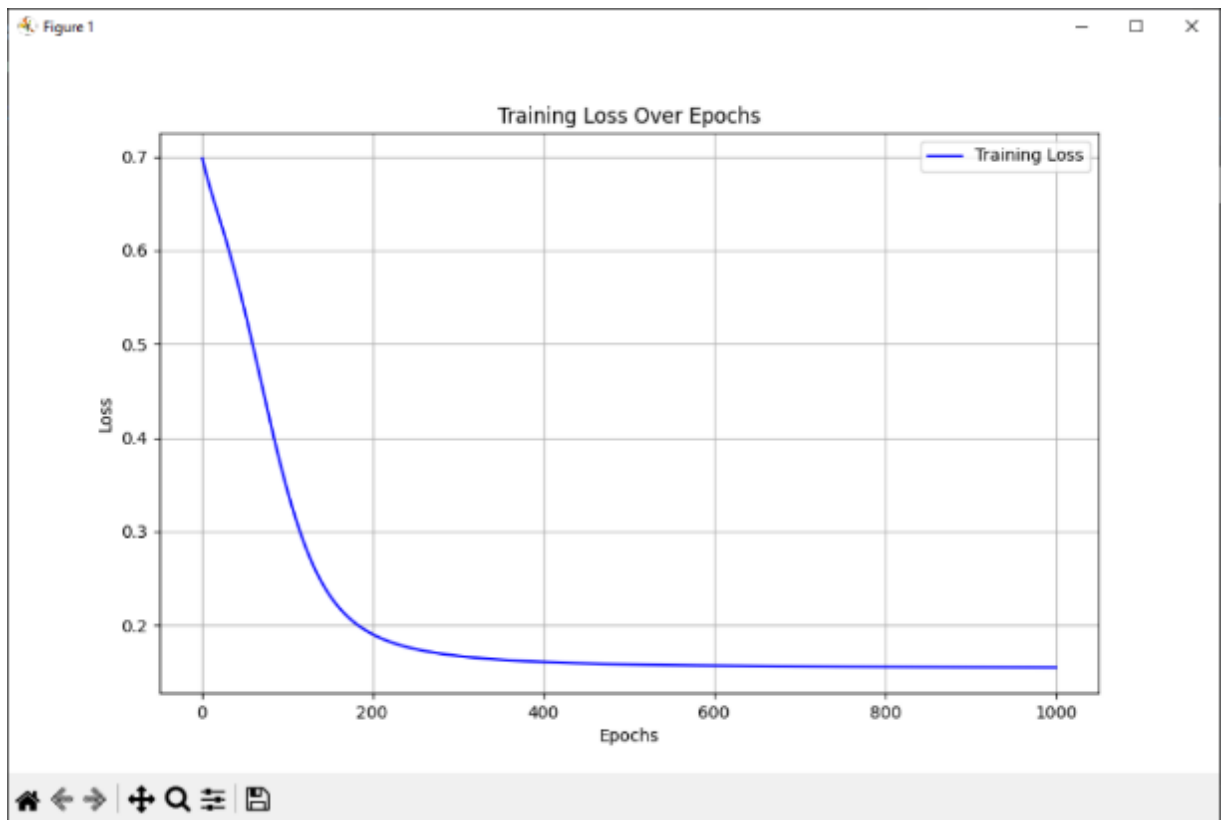


Рисунок 6.2 – Графік зміни шансу помилки нейромережевого алгоритму під час навчання

ВИСНОВКИ

У процесі виконання дипломної роботи було розроблено та модифіковано методи оптимізації й прогнозування, спрямовані на підвищення ефективності формування персоналізованого раціону харчування. Зокрема, було створено модифікований генетичний алгоритм (ГА) для задачі оптимізації та модифіковану багатоваріантну повнозв'язну нейронну мережу як представника DNN для прогнозування.

Проведене порівняння алгоритмів засвідчило, що саме модифікований генетичний алгоритм показав найкращі результати серед протестованих методів. Він забезпечив високу продуктивність пошуку оптимальних рішень, генерував більшу кількість варіантів раціонів і гарантував стабільну збіжність при мінімальному відхиленні від цільової калорійності. Найгірші результати продемонстрували класичні методи оптимізації, які мали обмежену здатність до пошуку глобального оптимуму та низьку різноманітність сформованих рішень.

Серед методів прогнозування найкращою виявилася модифікована MLP, яка забезпечила високу точність передбачень, швидке навчання й ефективне використання пам'яті. На відміну від неї, базові моделі прогнозування мали довший час навчання та нижчу адаптивність до різноманітних даних користувачів.

Таким чином, оптимальним рішенням для персоналізованого підбору раціону харчування є поєднання модифікованого генетичного алгоритму для оптимізації та модифікованої MLP для прогнозування. Такий підхід забезпечує високу точність, стабільність і продуктивність системи при збереженні ефективності використання обчислювальних ресурсів.

У першому розділі проведено системний аналіз існуючих методів розв'язання визначеної проблеми, а також здійснено огляд програмних рішень, що реалізують подібні функціональності. Виконано порівняння алгоритмів для оптимізації персоналізованого планування харчування,

зокрема аналіз підходів до адаптації раціону відповідно до індивідуальних потреб користувачів. Крім того, було розглянуто математичні формули для розрахунку індивідуальних показників.

Другий розділ присвячено обґрунтуванню вибору технологій для розробки програмного забезпечення, середовища розробки та системи управління базами даних.

Третій розділ містить детальний опис процесу реалізації програмного забезпечення, зокрема обрану нейронну мережу, а також опис класів, їхніх полів і методів, використаних у розробці. Окремо було проаналізовано інтерфейс користувача та структуру бази даних.

Четвертий та п'ятий розділи зосереджені на наданні інструкцій з експлуатації програмного продукту, що дозволяють користувачам ефективно взаємодіяти з системою. У шостому розділі розглянуто процес тестування програмного забезпечення, включаючи опис виконаних тестів та надано чек-лист для перевірки функціональності продукту.

Отримані результати мають як наукову, так і практичну цінність. Запропоновану систему можна інтегрувати у веб- або мобільні застосунки для автоматизованого підбору збалансованого раціону харчування. Подальші дослідження доцільно спрямувати на розширення набору ознак (врахування алергій, рівня фізичної активності, медичних показників) та адаптацію алгоритмів до роботи в режимі реального часу.

Робота була також апробована на конференції CS&SE@SW 2025: 8th Workshop for Young Scientists in Computer Science & Software Engineering, та подана як стаття в журналі «Реєстрація, зберігання і обробка даних».

Отже, виконання дипломної роботи надало можливість не лише закріпити теоретичні знання, але й здобути цінний практичний досвід у розробці програмного забезпечення. У процесі роботи було опановано методи створення ефективних та зручних рішень, орієнтованих на потреби користувачів, із дотриманням сучасних вимог до якості, функціональності та надійності програмних продуктів. Такий досвід став основою для розвитку

професійних компетенцій у галузі програмування та проєктування програмного забезпечення.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Основи харчування / М. І. Кручаниця, Н. В. Миронюк, В. В. Розумикова та ін. – Ужгород: УжНУ «Говерла», 2019. – 251 с.
2. Зубар Н. М. Основи фізіології та гігієни харчування / Н. М. Зубар. – Київ: Видавничий дім «Кондор», 2018. – 442 с.
3. Павлоцька Л. Ф. Фізіологія харчування / Л. Ф. Павлоцька, Н. В. Дуденко, В. В. Євлаш. – Харків: Світ книг, 2017. – 316 с.
4. Are Calories an Accurate Reflection of Good Nutrition? [Electronic resource]. – Access mode: <https://integrishealth.org/resources/on-your-health/2023/march/are-calories-an-accurate-reflection-of-good-nutrition>.
5. What is a macro diet, and how does it work? [Electronic resource]. – Access mode: <https://www.medicalnewstoday.com/articles/macro-diet>.
6. Nutrition tracking for real life [Electronic resource]. – Access mode: <https://www.myfitnesspal.com/en>.
7. AI in Food and Beverage Industry Size - Market Report on Share, Growth Trends & Forecasts Analysis (2025 - 2030) [Electronic resource]. – Access mode: <https://www.mordorintelligence.com/industry-reports/artificial-intelligence-in-food-and-beverages-market>.
8. AI in the Food Industry – How Artificial Intelligence and Robotics Are Shaping the Sector [Electronic resource]. – Access mode: <https://www.prismetric.com/ai-in-food-industry/#:~:text=AI%20is%20popular%20in%20food,reducing%20waste%20and%20optimizing%20production>.
9. AI in Personalized Nutrition: Comprehensive Overview and Real-World Applications [Electronic resource]. – Access mode: <https://medium.com/@preeti.rana.ai/ai-in-personalized-nutrition-comprehensive-overview-and-real-world-applications-ecad07b7f137#:~:text=AI%20in%20personalized%20nutrition%20involves,specifically%20designed%20for%20each%20individual>.

10. 12 Brilliant Mobile Apps for Plant-Based Eaters: Recipes, Nutrition, and More [Electronic resource]. – Access mode: <https://foodrevolution.org/blog/healthy-eating-nutrition-apps/#:~:text=MyFitnessPal,your%20caloric%20and%20macronutrient%20intake.>

11. Paprika recipe manager [Electronic resource]. – Access mode: <https://www.paprikaapp.com/>.

12. Samsung Food [Electronic resource]. – Access mode: <https://samsungfood.com/>.

13. Samsung оголошує про запуск Samsung Food: персоналізований сервіс з приготування їжі та рецептів на основі штучного інтелекту [Електронний ресурс]. – Режим доступу: <https://news.samsung.com/ua/samsung-announces-global-launch-of-samsung-food-an-ai-powered-personalized-food-and-recipe-service>.

14. With AI food recognition Samsung Food could be the ultimate meal-planning app [Electronic resource]. – Access mode: <https://www.theverge.com/2024/8/31/24231740/samsung-food-plus-vision-ai-food-list-management-ifa-2024>

15. Samsung Food: Meal Planning [Electronic resource]. – Access mode: <https://play.google.com/store/apps/details?id=com.foodient.whisk&hl=uk>.

16. Getting Real: Adam Devine and MyFitnessPal Talk Nutrition Tracking for Real Life [Electronic resource]. – Access mode: <https://www.prnewswire.com/news-releases/getting-real-adam-devine-and-myfitnesspal-talk-nutrition-tracking-for-real-life-302342315.html>.

17. Огляд MyFitnessPal. Чи варто завантажити додаток MyFitnessPal [Електронний ресурс]. – Режим доступу: https://icoola.ua/blog/my-fitness-pal-app/?srsltid=AfmBOoq9sVqT-sj4AGuGm9I-aiFLu3sqjTd8_nIwCgBNvcvq_JmIntbo.

18. MyFitnessPal: Calorie Counter [Electronic resource]. – Access mode: <https://play.google.com/store/apps/details?id=com.myfitnesspal.android&hl=uk>

19. Eat This Much [Electronic resource]. – Access mode: <https://www.eatthismuch.com/>
20. Aggarwal C. C. Data Mining: The Textbook / C. C. Aggarwal. – Springer, 2016. – 734 p.
21. Greedy Algorithms [Electronic resource]. – Access mode: <https://www.geeksforgeeks.org/greedy-algorithms/>.
22. Vanderbei R. J. Linear Programming: Foundations and Extensions / R. J. Vanderbei. – Springer, 2020. – 451 p.
23. Fishman G. S. Monte Carlo: Concepts, Algorithms, and Applications / G. S. Fishman. – Springer, 2017. – 713 p.
24. Олійник, А. Стохастичний підхід для вилучення правил асоціацій / А. Олійник, С. Субботін // Розпізнавання образів та аналіз зображень. – 2016. – Т. 26, № 2. – С. 419–426.
25. Michalewicz Z. How to Solve It: Modern Heuristics / Z. Michalewicz, D. B. Fogel. – Springer, 2019. – 554 p.
26. Субботін С. О., Олійник А. О., Федорченко Є. М. та ін. Математичні та програмні засоби для прийняття рішень, розпізнавання образів і інтелектуального діагностування / С. О. Субботін, А. О. Олійник, Є. М. Федорченко та ін. – НУ "Запорізька політехніка", 2020. – 271 с.
27. Liu Z. A study of random forest algorithm for data mining applications / Z. Liu, Y. Liu. – In: Proceedings of the International Conference on Applied Computing and Information Technology (ACIT), 2016. – P. 257-261.
28. Chen T. XGBoost: A Scalable Tree Boosting System / T. Chen, C. Guestrin. – In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016. – P. 785-794.
29. Zhang X. Learning Deep Architectures for AI / X. Zhang, Y. Bengio. – Foundations and Trends® in Machine Learning, 2018. – Vol. 10, Issue 3-4. – P. 147-239.
30. Obesity and overweight [Electronic resource]. – Access mode: <https://www.who.int/news-room/fact-sheets/detail/obesity-and-overweight>.

31. Assessing Your Weight and Health Risk [Electronic resource]. – Access mode: https://www.nhlbi.nih.gov/health/educational/lose_wt/risk.htm.
32. Basal Metabolic Rate Calculator [Electronic resource]. – Access mode: <https://www.garnethealth.org/news/basal-metabolic-rate-calculator#:~:text=BMR%20Definition%3A%20Your%20Basal%20Metabolic,stay%20in%20bed%20all%20day>.
33. Scott, M. L. Programming Language Pragmatics / M. L. Scott. – 4th ed. – Boston: Addison-Wesley, 2016. – 1140 p.
34. The Importance of Selecting Programming Languages Carefully [Electronic resource]. – Access mode: <https://iies.in/blog/importance-of-selecting-programming-languages-carefully/>.
35. Technology 2024 [Electronic resource]. – Access mode: <https://survey.stackoverflow.co/2024/technology#1-programming-scripting-and-markup-languages>.
36. The Most Popular Programming Languages in 2024 [Electronic resource]. – Access mode: <https://www.darly.solutions/blog/the-most-popular-programming-languages-in-2021>.
37. TIOBE Index for January 2025 [Electronic resource]. – Access mode: <https://www.tiobe.com/tiobe-index/>.
38. PYPL PopularitY of Programming Language [Electronic resource]. – Access mode: <https://pypl.github.io/PYPL.html>.
39. What is the C# programming language? [Electronic resource]. – Access mode: <https://www.pluralsight.com/resources/blog/cloud/what-is-the-c-programming-language>.
40. A History Timeline About C# [Electronic resource]. – Access mode: <https://historytimelines.co/timeline/c>
41. C# language documentation [Electronic resource]. – Access mode: <https://learn.microsoft.com/en-us/dotnet/csharp/>.

42. C#: History, .NET, Pros/Cons [Electronic resource]. – Access mode: <https://dev.to/snelson723/cs-history-net-relationship-and-advantagesdisadvantages-102b>.
43. Python – the language of today and tomorrow [Electronic resource]. – Access mode: <https://pythoninstitute.org/about-python#:~:text=Python%20was%20created%20by%20Guido,called%20Monty%20Python's%20Flying%20Circus>.
44. Real Python Tutorials [Electronic resource]. – Access mode: <https://realpython.com/>.
45. Top IDE index [Electronic resource]. – Access mode: <https://pypl.github.io/IDE.html>.
46. Visual Studio 2022 [Electronic resource]. – Access mode: <https://visualstudio.microsoft.com/>.
47. Windows Presentation Foundation documentation [Electronic resource]. – Access mode: <https://learn.microsoft.com/en-us/dotnet/desktop/wpf/?view=netdesktop-9.0>.
48. Що таке СУБД і для чого вони потрібні [Електронний ресурс]. – Режим доступу: <https://foxminded.ua/systema-upravlinnia-bazamy-danykh/>.
49. Database Defined [Electronic resource]. – Access mode: <https://www.oracle.com/ua/database/what-is-database/>.
50. Microsoft SQL Server [Electronic resource]. – Access mode: <https://www.microsoft.com/uk-ua/sql-server>.
51. PyTorch [Electronic resource]. – Access mode: <https://pytorch.org/>.
52. Numpy 2.2.2 [Electronic resource]. – Access mode: <https://pypi.org/project/numpy/>.
53. Scikit-learn [Electronic resource]. – Access mode: <https://scikit-learn.org/stable/>.
54. Functional Diagrams: the Flow of Information within a Control System [Electronic resource]. – Access mode: <https://control.com/textbook/instrumentation-documents/functional-diagrams/>.

55. A Deep Architecture: Multi-Layer Perceptron [Electronic resource]. – Access mode: <https://medium.com/@nlunge786/a-deep-architecture-multi-layer-perceptron-164bc5ff3842>
56. Схеми СУБД: внутрішні, концептуальні, зовнішні [Електронний ресурс]. – Режим доступу: <https://www.guru99.com/uk/dbms-schemas.html>.
57. Colour [Electronic resource]. – Access mode: <https://www.britannica.com/science/color>.
58. 10 Rules of Good UI Design to Follow On Every Web Design Project [Electronic resource]. – Access mode: <https://www.elegantthemes.com/blog/resources/10-rules-of-good-ui-design-to-follow-on-every-web-design-project>.
59. How to create a user-friendly interface: 16 best techniques [Electronic resource]. – Access mode: <https://geniusee.com/single-blog/16-techniques-for-creating-a-user-friendly-interface>.
60. Color Psychology in UX Design [Electronic resource]. – Access mode: <https://supercharge.design/blog/color-psychology-in-ux-design>.
61. ColorMagic Health [Electronic resource]. – Access mode: <https://colormagic.app/palette/67865cb04db034c1b8687e63>.

ДОДАТОК А
Технічне завдання

A.1 Вступ

У сучасному світі технології відіграють ключову роль у вирішенні багатьох щоденних завдань, зокрема в сфері здоров'я та харчування. Зі зростанням обсягу доступної інформації та різноманіття варіантів продуктів, важливим завданням стає оптимізація раціону для забезпечення здоров'я та добробуту. Програмні рішення стають важливими інструментами, які дозволяють зручно управляти харчуванням, враховуючи індивідуальні потреби та обмеження користувачів.

Правильне харчування є основою підтримки здоров'я та високої продуктивності, однак вибір відповідних продуктів і створення збалансованого раціону може бути складним процесом. Урахування таких факторів, як вік, стать, рівень фізичної активності та харчові вподобання, є важливим для досягнення оптимальних результатів. Розробка програмного забезпечення, яке автоматизує цей процес, дає можливість користувачам швидко та ефективно отримати персоналізовані рекомендації, що сприяють досягненню здорового способу життя.

A.1.1 Назва програми

Назва програми: «EatMe».

A.1.2 Призначення та область застосування

Програмний продукт «EatMe» призначений для створення персоналізованих планів харчування, що враховують індивідуальні параметри користувача, такі як вік, стать, рівень фізичної активності, харчові вподобання та обмеження. Програма забезпечує зручний інтерфейс для введення особистих даних і автоматичного генерування рекомендацій щодо

раціону, сприяючи здоровому способу життя та ефективному управлінню харчуванням.

Область застосування: приватні особи.

A.2 Підстави для розробки

Основою для проведення є завдання на дипломну роботу на тему «на тему «Розробка та дослідження програмного забезпечення особистого помічника для формування щоденного раціону харчування», затверджене наказом Національного університету «Запорізька політехніка» № 447 від 30 вересня 2025 р.

A.3 Призначення розробки

Програма «EatMe» розроблена для створення персоналізованих планів харчування, що враховують індивідуальні потреби користувачів. Це програмне забезпечення сприяє покращенню здорового способу життя шляхом автоматичного генерування збалансованих раціонів, які відповідають особистим параметрам, таким як вік, стать, рівень фізичної активності, харчові вподобання та обмеження, а також можливі алергії чи дієтичні рекомендації.

Основною метою розробки є створення ефективного інструменту для складання індивідуальних планів харчування, що дозволяють користувачам досягати своїх цілей, таких як підтримка оптимальної ваги, поліпшення фізичної форми або забезпечення специфічних дієтичних потреб. Програма також дозволяє враховувати зміни в режимі життя користувача, адаптуючи раціони харчування відповідно до нових параметрів. Вона інтегрує передові технології, зокрема генетичні алгоритми та нейронні мережі, для більш точного та ефективного генерування харчових планів, що забезпечує максимальну персоналізацію та зручність користування.

Основне призначення:

- створення персоналізованих планів харчування з урахуванням індивідуальних параметрів користувача, таких як: вік, стать, рівень фізичної активності, харчові уподобання, алергії;
- автоматичне генерування планів харчування, що базується на необхідній калорійності для кожного користувача в залежності від його фізичних параметрів;
- адаптація планів харчування в залежності від змін у фізичному стані користувача, таких як зміна ваги, рівня активності чи здоров'я;
- інтеграція генетичних алгоритмів та нейронних мереж для точнішого підбору планів харчування, що враховують всі необхідні індивідуальні дані користувача;
- покращення якості життя користувачів через спрощення процесу складання раціонів харчування, зосереджених на калорійності;
- зменшення часу на пошук інформації щодо здорового харчування, а також доступність здорових і простих у приготуванні рецептів;
- забезпечення зручного і доступного інтерфейсу для легкого управління персональними налаштуваннями, що дозволяє користувачеві без труднощів формувати власний план харчування.

A.4 Вимоги до програми

A.4.1 Вимоги до функціональних характеристик

Вимоги до функціональних характеристик:

- програмне забезпечення повинно забезпечувати можливість створення та налаштування профілю користувача, де можна вказати особисті дані, такі як ім'я, вік, стать, зріст, вага.;
- після заповнення профілю, програмне забезпечення повинно автоматично створювати персоналізований план харчування, що включає від 3 до 5 прийомів їжі на день;

- програма повинна забезпечувати різноманітність раціону, зберігаючи відповідність рекомендаціям щодо здорового харчування;
- програмне забезпечення повинно мати можливість адаптувати план харчування на основі минулих виборів користувача. Це дозволяє врахувати звички та вподобання користувача, що покращує персоналізацію плану та підвищує зручність у використанні;
- програмне забезпечення повинно автоматично розраховувати необхідну кількість калорій на день, використовуючи особисті дані користувача, такі як вага, вік, рівень фізичної активності та інші параметри;
- програмне забезпечення повинно мати можливість збереження особистих даних користувача, планів харчування, обраних страв, а також інформації про калорії;
- програмне забезпечення повинно мати зручний, інтуїтивно зрозумілий інтерфейс, що дозволяє користувачам без труднощів вводити та редагувати свої дані, переглядати план харчування та отримувати необхідну інформацію.

A.4.2 Вимоги до надійності

Вимоги до надійності:

- програмне забезпечення повинно працювати безперервно та без збоїв протягом тривалого часу, гарантуючи надійний доступ до персоналізованих планів харчування для користувачів;
- у разі виникнення технічних проблем, система повинна забезпечити коректне відновлення роботи без втрати даних користувачів;
- програма повинна бути здатною ефективно працювати при великих обсягах даних та високих навантаженнях, наприклад, при обробці великої кількості одночасних запитів або при збереженні численних профілів користувачів;

– програмне забезпечення повинно містити систему моніторингу, що автоматично виявляє потенційні проблеми та нештатні ситуації, а також веде логування подій для подальшого аналізу та забезпечення надійності.

A.4.3 Вимоги до технічних та програмних засобів

Для коректної роботи програми пристрій повинен відповідати наступним технічним та програмним вимогам:

- оперативна пам'ять – мінімум 8 ГБ для забезпечення стабільної роботи програми, особливо при обробці великих обсягів даних;
- місце на диску – мінімум 1500 МБ вільного простору для встановлення основного програмного забезпечення та збереження бази даних;
- операційна система – Windows 10 або новіша версія для підтримки всіх функцій, оновлень;
- обсяг жорсткого диска – не менше 256 ГБ;
- процесор – багатоядерний, з тактовою частотою не менше 1.5 ГГц.

A.5 Вимоги до програмної документації

Документація програмного забезпечення повинна містити наступні розділи:

- технічне завдання, що ґрунтується на дослідженні вимог користувачів та специфіки галузі;
- детальний опис функціональності та роботи програми;
- пояснювальну записку, що розкриває принципи розробки та основні рішення проекту.

A.6 Стадії і етапи розробки

A.6.1 Стадії розробки

Стадії розробки:

- дослідження та формування концепції;
- розробка технічного завдання;
- проектування структури програмного забезпечення;
- розробка компонентів програмного забезпечення;
- тестування готового програмного забезпечення;
- оформлення документації та завершення розробки.

A.6.2 Етапи розробки та зміст роботи на кожному етапі

Етапи розробки:

- аналіз предметної області – проведення дослідження актуальності теми, визначення сфери застосування, уточнення ключових понять і концепцій;
- аналіз існуючих методів – розгляд і оцінка наявних підходів до вирішення завдань у заданій предметній області;
- аналіз існуючих аналогів – вивчення програмних засобів, які вже реалізують подібні функціональні можливості;
- дослідження алгоритмів персонального планування раціону харчування – аналіз алгоритмів, які можливо використати для формування індивідуальних раціонів харчування;
- оцінка підходів до оптимізації персоналізованого раціону харчування – вивчення методів оптимізації для підвищення ефективності персоналізації раціону;
- аналіз формул для розрахунку індивідуальних показників – розгляд математичних формул, що застосовуються для визначення ключових характеристик;

- формування вимог до програмного забезпечення – встановлення функціональних характеристик і цілей, які має забезпечувати програмний продукт;
- формування вимог до надійності – визначення необхідного рівня стійкості та працездатності програмного забезпечення;
- формування технічних і програмних вимог – затвердження мінімальних характеристик обладнання та середовища, необхідних для стабільної роботи програмного забезпечення;
- формування вимог до документації – опис структури та змісту програмної документації;
- проектування інтерфейсної підсистеми – розробка макету інтерфейсу користувача з урахуванням зручності використання;
- проектування аналітичної підсистеми – розробка концепції основних алгоритмів;
- проектування структури бази даних – визначення основних сутностей і зв'язків між ними для побудови бази даних;
- розробка інтерфейсної підсистеми – створення елементів інтерфейсу та налаштування їх функціональності;
- розробка аналітичної підсистеми – реалізація основних алгоритмів і застосування;
- розробка бази даних і її наповнення – створення таблиць бази даних і заповнення їх початковими даними;
- інтеграція компонентів програмного забезпечення – об'єднання інтерфейсної та аналітичної підсистем у єдину систему;
- тестування програмного забезпечення – перевірка коректності виконання функцій і відповідності вимогам;
- усунення дефектів – виправлення виявлених помилок та покращення функціональності;

- підготовка опису програми – оформлення документації, що включає загальні відомості, логіку роботи, алгоритми, технічні вимоги та інструкції для користувачів;
- завершення пояснювальної записки – підготовка повного тексту записки з додатками для фінального оформлення.

ДОДАТОК Б
Текст програми

Б.1 Файл GeneticAlgorithm.py

```

import random
from Utils.Logger import Logger

# Ініціалізація логгера
logger = Logger(log_file="GeneticAlgorithm.log", log_dir="logs")

class GeneticAlgorithm:
    def __init__(self, population_size, generations, mutation_rate):
        # Ініціалізація параметрів генетичного алгоритму
        self.population_size = population_size
        self.generations = generations
        self.mutation_rate = mutation_rate

    # Метод для створення індивіда (раціону) на основі заданих страв
    def create_individual(self, breakfast_items, lunch_items, dinner_items,
snacks_items, user):
        try:
            num_dishes = random.randint(3, 5) # Від 3 до 5 різних страв
            selected_dishes = []

            # Створення раціону
            selected_dishes.append(self.create_dish(breakfast_items, user))
            if num_dishes == 3:
                selected_dishes.append(self.create_dish(lunch_items, user))
            else:
                selected_dishes.append(self.create_dish(snacks_items, user))

            if num_dishes == 3:
                selected_dishes.append(self.create_dish(dinner_items, user))
            else:
                selected_dishes.append(self.create_dish(lunch_items, user))

            if num_dishes == 4:
                selected_dishes.append(self.create_dish(dinner_items, user))
            elif num_dishes == 5:
                selected_dishes.append(self.create_dish(snacks_items, user))
                selected_dishes.append(self.create_dish(dinner_items, user))

            servings = [random.randint(1, 5) for _ in selected_dishes] #
Випадкова кількість порцій
            return MealPlan(list(zip(selected_dishes, servings))) #
Створення MealPlan
        except Exception as e:
            logger.error(f"Помилка при створенні індивіда: {str(e)}")
            raise

    # Метод для створення окремої страви
    def create_dish(self, dishes, user):
        try:
            if random.random() < 0.5: # 50% ймовірність вибору улюблених
страв
                favorite_dishes = [d for d in dishes if any(ingredient in
user.preferences for ingredient in d.ingredients)]
                if favorite_dishes:
                    dish = random.choice(favorite_dishes)
                else:

```

```

        dish = random.choice(dishes) # Якщо немає улюблених,
вибираємо випадково
    else:
        dish = random.choice(dishes)
        return dish
    except Exception as e:
        logger.error(f"Помилка при створенні страви: {str(e)}")
        raise

    # Метод для оцінки індивіда за відповідністю до калорійної мети
    def fitness(self, individual, calorie_goal):
        """Оцінити індивіда на основі відповідності до калорійної мети."""
        return abs(individual.total_calories - calorie_goal)

    # Метод для мутації індивіда
    def mutate(self, individual, breakfast_items, lunch_items, dinner_items,
snacks_items):
        try:
            if random.random() < self.mutation_rate: # Якщо потрібно
здійснити мутацію
                index = random.randint(0,
len(individual.dishes_with_servings) - 1)
                if random.random() < 0.5: # Зміна порції
                    servings = random.randint(1, 5)
                    individual.dishes_with_servings[index] =
(individual.dishes_with_servings[index][0], servings)
                else: # Заміна страви
                    existing_dishes = [dish for dish, _ in
individual.dishes_with_servings]
                    dishes = []
                    # Вибір страви для заміни залежно від індексу
                    if index == 0:
                        dishes = breakfast_items
                    elif index == 1:
                        if len(existing_dishes) == 3:
                            dishes = lunch_items
                        else:
                            dishes = snacks_items
                    elif index == 2:
                        if len(existing_dishes) == 3:
                            dishes = dinner_items
                        else:
                            dishes = lunch_items
                    elif index == 3:
                        if len(existing_dishes) == 4:
                            dishes = dinner_items
                        else:
                            dishes = snacks_items
                    elif index == 4:
                        dishes = dinner_items

                    # Вибір нової страви
                    new_dish = random.choice([d for d in dishes if d not in
existing_dishes])

                    servings = random.randint(1, 5)
                    individual.dishes_with_servings[index] = (new_dish,
servings)

                individual.total_calories = individual.calculate_total_calories()

# Оновлення калорій
        return individual
    except Exception as e:

```

```

        logger.error(f"Помилка при мутації індивіда: {str(e)}")
        raise

    # Метод для виконання схрещування двох батьків
    def crossover(self, parent1, parent2):
        try:
            pivot = random.randint(1, len(parent1.dishes_with_servings) - 1)
# Точка схрещування
            child_dishes_with_servings = (
                parent1.dishes_with_servings[:pivot]
                +
                parent2.dishes_with_servings[pivot:]
            )
            return MealPlan(child_dishes_with_servings) # Створення нащадка
        except Exception as e:
            logger.error(f"Помилка при виконанні схрещування: {str(e)}")
            raise

    # Основний метод для виконання генетичного алгоритму
    def run(self, breakfast_items, lunch_items, dinner_items, snacks_items,
            calorie_goal, user):
        try:
            # Створення початкової популяції
            population = [self.create_individual(breakfast_items,
            lunch_items, dinner_items, snacks_items, user) for _ in range(self.population_size)]

            for generation in range(self.generations):
                # Сортування популяції за fitness
                population = sorted(population, key=lambda ind:
                self.fitness(ind, calorie_goal))
                next_population = population[:self.population_size // 2] #
                Відбір найкращих

                # Схрещення та мутація для створення нової популяції
                while len(next_population) < self.population_size:
                    parent1, parent2 = random.sample(next_population, 2)
                    child = self.crossover(parent1, parent2)
                    next_population.append(self.mutate(child,
                    breakfast_items, lunch_items, dinner_items, snacks_items))

                population = next_population

            return population # Повертаємо найкращих індивідів
        except Exception as e:
            logger.error(f"Помилка при виконанні алгоритму: {str(e)}")
            raise

    # Клас MealPlan для представлення плану харчування
    class MealPlan:
        def __init__(self, dishes_with_servings):
            self.dishes_with_servings = dishes_with_servings # Список пар
            (страва, кількість порцій)
            self.total_calories = self.calculate_total_calories() # Загальна
            кількість калорій

        # Метод для підрахунку загальної кількості калорій
        def calculate_total_calories(self):
            """Підрахувати та зберегти загальну кількість калорій в раціоні."""
            return sum(dish.calories_per_serving * servings for dish, servings in
            self.dishes_with_servings)

        # Метод для зручного виведення інформації про план харчування
        def __repr__(self):

```

```

        dishes_info = ', '.join(f"{dish.name}: {servings} порцій" for dish,
servings in self.dishes_with_servings)
        return f"MealPlan({dishes_info}, Total Calories:
{self.total_calories})"

# Клас Dish для представлення страви
class Dish:
    def __init__(self, name, calories_per_serving, ingredients):
        self.name = name
        self.calories_per_serving = calories_per_serving
        self.ingredients = ingredients # Список інгредієнтів

    # Метод для зручного виведення інформації про страву
    def __repr__(self):
        return f"Dish(name={self.name},
calories_per_serving={self.calories_per_serving}, ingredients={self.ingredients})"

```

Б.2 Файл NeuralNetwork.py

```

import torch
import torch.nn as nn
import torch.optim as optim
from sklearn.preprocessing import LabelEncoder
from Utils.Logger import Logger

# Ініціалізація логгера
logger = Logger(log_file="NeuralNetwork.log", log_dir="logs")

class NeuralNetwork:
    def __init__(self, ingredient_vocab, dish_name_vocab, embedding_dim=16,
hidden_size=64, learning_rate=0.001):
        try:
            self.ingredient_encoder = LabelEncoder() # Ініціалізація
кодувальника для інгредієнтів
            self.dish_name_encoder = LabelEncoder() # Ініціалізація
кодувальника для назв страв

            # Ініціалізація словників для інгредієнтів і назв страв
            self.ingredient_encoder.fit(ingredient_vocab)
            self.dish_name_encoder.fit(dish_name_vocab)

            # Розміри словників
            ingredient_vocab_size = len(self.ingredient_encoder.classes_)
            dish_name_vocab_size = len(self.dish_name_encoder.classes_)

            # Ініціалізація шарів для векторизації інгредієнтів і назв страв
            self.ingredient_embedding = nn.Embedding(ingredient_vocab_size,
embedding_dim)
            self.dish_name_embedding = nn.Embedding(dish_name_vocab_size,
embedding_dim)

            # Модель нейромережі
            self.model = nn.Sequential(
                nn.Linear(embedding_dim * 2, hidden_size), # Вхідний шар
                nn.ReLU(), # Функція активації
                nn.Linear(hidden_size, 1), # Вихідний шар
                nn.Sigmoid() # Сигмоїд для ймовірності
            )

```

```

        self.criterion = nn.BCELoss() # Втрата для бінарної класифікації
(Binary Cross-Entropy Loss)
        self.optimizer = optim.Adam(self.model.parameters(),
lr=learning_rate) # Оптимізатор Adam
        logger.info("Модель ініціалізована успішно.")
    except Exception as e:
        logger.error(f"Помилка при ініціалізації нейромережі: {str(e)}")
        raise

    def encode_mealplan(self, mealplan):
        try:
            """Кодує MealPlan у числовий формат."""
            encoded_features = []
            for dish, servings in mealplan.dishes_with_servings:
                # Кодування інгредієнтів
                ingredient_indices =
self.ingredient_encoder.transform(dish.ingredients) # Перетворення інгредієнтів у
індекси
                ingredient_embedding =
self.ingredient_embedding(torch.tensor(ingredient_indices,
dtype=torch.long)).mean(dim=0) # Векторне представлення інгредієнтів

                # Кодування назви страви
                dish_index = self.dish_name_encoder.transform([dish.name])[0]
# Перетворення назви страви в індекс
                dish_embedding =
self.dish_name_embedding(torch.tensor([dish_index], dtype=torch.long)).squeeze() #
Векторне представлення назви страви

                # Додавання векторів
                combined_features = torch.cat([ingredient_embedding,
dish_embedding])

                encoded_features.append(combined_features)

            # Агрегація по стравам
            if encoded_features:
                mealplan_vector = torch.stack(encoded_features).mean(dim=0)
# Середнє значення по всіх стравах
            else:
                mealplan_vector =
torch.zeros(self.ingredient_embedding.embedding_dim * 2) # Якщо раціон порожній
            return mealplan_vector
        except Exception as e:
            logger.error(f"Помилка при кодуванні MealPlan: {str(e)}")
            raise

    def train(self, liked_mealplans, disliked_mealplans, epochs=100):
        try:
            """Тренування моделі."""
            # Формуємо дані для тренування
            training_data = liked_mealplans + disliked_mealplans
            labels = torch.tensor([1] * len(liked_mealplans) + [0] *
len(disliked_mealplans), dtype=torch.float32)

            # Кодування вхідних даних
            input_data = torch.stack([self.encode_mealplan(mp) for mp in
training_data])

            # Тренувальний цикл
            for epoch in range(epochs):
                self.model.train() # Переводимо модель в режим тренування

```

```

        predictions = self.model(input_data).squeeze() #
Прогнозування
        loss = self.criterion(predictions, labels) # Обчислення
втрат

        # Оновлення ваг
        self.optimizer.zero_grad()
        loss.backward(retain_graph=True)
        self.optimizer.step()

        # Логування кожні 10 епох
        if (epoch + 1) % 10 == 0:
            logger.info(f"Epoch {epoch+1}/{epochs}, Loss:
{loss.item():.4f}")

        logger.info("Тренування завершено успішно.")
    except Exception as e:
        logger.error(f"Помилка під час тренування: {str(e)}")
        raise

    def predict(self, mealplans):
        try:
            """Оцінює список MealPlan."""
            self.model.eval() # Переводимо модель в режим оцінки
            input_data = torch.stack([self.encode_mealplan(mp) for mp in
mealplans])
            scores = self.model(input_data).squeeze().detach().numpy() #
Прогнозування та від'єднання від графу
            return scores
        except Exception as e:
            logger.error(f"Помилка при прогнозуванні: {str(e)}")
            raise

    def sort_mealplans(self, mealplans):
        try:
            """Сортує MealPlan за оцінками моделі."""
            scores = self.predict(mealplans) # Отримуємо оцінки
            return [mp for _, mp in sorted(zip(scores, mealplans), key=lambda
x: x[0], reverse=True)] # Сортуємо за зменшенням оцінки
        except Exception as e:
            logger.error(f"Помилка при сортуванні MealPlans: {str(e)}")
            raise

```

Б.3 Файл RespieDataBase.py

```

import json
import csv
from Algorithms.GeneticAlgorithm import Dish, MealPlan
from Utils.Logger import Logger

# Ініціалізація логера
logger = Logger(log_file="DataManager.log", log_dir="logs")

class DataManager(object):
    # Метод для завантаження даних користувача з JSON файлу
    def load_user_data(self, file_path):
        try:
            with open(file_path, 'r', encoding='utf-8') as file:

```

```

        data = json.load(file)
        return User.from_json(data)
    except FileNotFoundError:
        logger.error(f"Файл з шляхом {file_path} не знайдено.")
        raise
    except json.JSONDecodeError:
        logger.error(f"Помилка декодування JSON у файлі {file_path}.")
        raise
    except Exception as e:
        logger.error(f"Неочікувана помилка при завантаженні даних з {file_path}: {str(e)}")
        raise

# Метод для завантаження даних про страви з CSV файлу
def load_food_data(self, source, is_database=False, db_config=None):
    dishes = []
    try:
        with open(source, 'r', newline='', encoding='utf-8') as file:
            reader = csv.DictReader(file)
            for row in reader:
                dish = Dish(
                    name=row['\ufeffname'],
                    calories_per_serving=int(row['calories_per_serving']),
                    ingredients=row['ingredients'].split(',')
                )
                dishes.append(dish)
            return dishes
    except FileNotFoundError:
        logger.error(f"Файл {source} не знайдено.")
        raise
    except KeyError as e:
        logger.error(f"Відсутній необхідний стовпець у файлі {source}: {str(e)}")
        raise
    except Exception as e:
        logger.error(f"Неочікувана помилка при завантаженні даних з {source}: {str(e)}")
        raise

# Метод для збереження даних про страви у файл
def save_dishes(self, dishes, file_path):
    try:
        dishes_data = [
            {
                "dishes_with_servings": [
                    {
                        "name": serving[0].name,
                        "calories_per_serving":
serving[0].calories_per_serving,
                        "ingredients": serving[0].ingredients,
                        "servings": serving[1]
                    }
                    for serving in dish.dishes_with_servings
                ],
                "total_calories": dish.total_calories
            }
            for dish in dishes
        ]
        with open(file_path, 'w', encoding='utf-8') as file:
            json.dump(dishes_data, file, ensure_ascii=False, indent=4)
        logger.info(f"Дані успішно збережено у {file_path}")
    
```

```

except Exception as e:
    logger.error(f"Помилка при збереженні страв у {file_path}:"
{str(e)})
    raise

# Метод для завантаження MealPlan з JSON файлу
def load_mealplans_from_json(self, file_path):
    """Завантажує список MealPlan із JSON-файлу."""
    mealplans = []
    try:
        with open(file_path, 'r', encoding='utf-8') as file:
            data = json.load(file)

            for mealplan_data in data:
                dishes_with_servings = [
                    (Dish(
                        name=dish_data['name'],
calories_per_serving=dish_data['calories_per_serving'],
                        ingredients=dish_data['ingredients']
                    ), dish_data['servings'])
                    for dish_data in mealplan_data['dishes_with_servings']
                ]
                mealplans.append(MealPlan(dishes_with_servings))
            return mealplans
    except FileNotFoundError:
        logger.error(f"Файл з шляхом {file_path} не знайдено.")
        raise
    except json.JSONDecodeError:
        logger.error(f"Помилка декодування JSON у файлі {file_path}.")
        raise
    except Exception as e:
        logger.error(f"Неочікувана помилка при завантаженні mealplans з
{file_path}: {str(e)}")
        raise

# Метод для екстракції лексикону інгредієнтів та назв страв
def extract_vocab(self, *mealplan_lists):
    ingredient_vocab = set()
    dish_name_vocab = set()
    try:
        for mealplan_list in mealplan_lists:
            for mealplan in mealplan_list:
                for dish, _ in mealplan.dishes_with_servings:
                    dish_name_vocab.add(dish.name)
                    ingredient_vocab.update(dish.ingredients)

        return sorted(ingredient_vocab), sorted(dish_name_vocab)
    except Exception as e:
        logger.error(f"Помилка при екстракції лексикону: {str(e)}")
        raise

# Клас користувача
class User:
    def __init__(self, name, age, height, weight, gender, bmi, diet_category,
allergies, preferences, caloric_needs):
        self.name = name
        self.age = age
        self.height = height
        self.weight = weight
        self.gender = gender
        self.bmi = bmi

```

```

        self.diet_category = diet_category
        self.allergies = allergies
        self.preferences = preferences
        self.caloric_needs = caloric_needs

    @classmethod
    def from_json(cls, json_data):
        return cls(
            name=json_data['Name'],
            age=json_data['Age'],
            height=json_data['Height'],
            weight=json_data['Weight'],
            gender=json_data['Gender'],
            bmi=json_data['BMI'],
            diet_category=json_data['BMICategory'],
            allergies=json_data['Allergies'],
            preferences=json_data['Preferences'],
            caloric_needs=json_data['Calories']
        )

```

Б.4 Файл Main.py

```

import os
from Algorithms.GeneticAlgorithm import GeneticAlgorithm
from Algorithms.NeuralNetwork import NeuralNetwork
from Utils.DataManager import DataManager
from Utils.Logger import Logger
import sys
import json

# Ініціалізація логгера
logger = Logger(log_file="Main.log", log_dir="logs")

def main():
    try:
        # Завантаження даних
        data_manadger = DataManager()

        user_file_path = "D:\\PersonalAssistant\\userData.json"
        user_data = data_manadger.load_user_data(user_file_path)

        breakfast_file_path = "D:\\PersonalAssistant\\breakfastData.csv"
        breakfast_items = data_manadger.load_food_data(breakfast_file_path)

        lunch_file_path = "D:\\PersonalAssistant\\lunchData.csv"
        lunch_items = data_manadger.load_food_data(lunch_file_path)

        dinner_file_path = "D:\\PersonalAssistant\\dinnerData.csv"
        dinner_items = data_manadger.load_food_data(dinner_file_path)

        snacks_file_path = "D:\\PersonalAssistant\\snacksData.csv"
        snacks_items = data_manadger.load_food_data(snacks_file_path)

        # Фільтрація страв за алергією
        breakfast_items = [
            dish for dish in breakfast_items
            if not any(ingredient in user_data.allergies for ingredient in
dish.ingredients)

```

```

    ]
    lunch_items = [
        dish for dish in lunch_items
        if not any(ingredient in user_data.allergies for ingredient in
dish.ingredients)
    ]
    dinner_items = [
        dish for dish in dinner_items
        if not any(ingredient in user_data.allergies for ingredient in
dish.ingredients)
    ]
    snacks_items = [
        dish for dish in snacks_items
        if not any(ingredient in user_data.allergies for ingredient in
dish.ingredients)
    ]

    # Генетичний алгоритм для пошуку раціону
    ga = GeneticAlgorithm(population_size=1000, generations=100,
mutation_rate=0.01)
    optimal_meal_plan = ga.run(breakfast_items, lunch_items,
dinner_items, snacks_items, calorie_goal=user_data.caloric_needs, user=user_data)

    # Вибір унікальних рекомендацій та видалення неоптимальних варіантів
    top_meal_calories = optimal_meal_plan[0].total_calories
    unique_list = []

    for meal_plan in optimal_meal_plan[:]:
        # Для кожного
meal_plan в optimal_meal_plan
        # Перевірка унікальності
        unique = True
        if len(unique_list) != 0:
            for item in unique_list:
                if len(item.dishes_with_servings) ==
len(meal_plan.dishes_with_servings):
                    unique_proof = False
                    for (item_dish, item_servings), (meal_plan_dish,
meal_plan_servings) in zip(item.dishes_with_servings,
meal_plan.dishes_with_servings):
                        if item_dish.name != meal_plan_dish.name or
item_servings != meal_plan_servings:
                            unique_proof = True
                    if unique_proof == False:
                        unique = False

            if unique == True:
                # Якщо унікальне
                dish_names = [dish.name for dish, _ in
meal_plan.dishes_with_servings] # то зчитуємо список унікальних варіантів страв
                if len(dish_names) == len(set(dish_names)):
                    # якщо немає дублікатів
                    if meal_plan.total_calories == top_meal_calories:
                        # якщо оптимальні калорії
                        unique_list.append(meal_plan)
                    # додати у список унікальних meal_plan і перейти до наступного у початок циклу

    # Завантаження даних
    liked_mealplans =
data_manadger.load_mealplans_from_json("D:\\PersonalAssistant\\likedData.json")

```

```

        disliked_mealplans
data_manadger.load_mealplans_from_json("D:\\PersonalAssistant\\dislikedData.json")

        # Визначення даних
        ingredient_vocab,          dish_name_vocab
data_manadger.extract_vocab(liked_mealplans, disliked_mealplans, unique_list)

        # Ініціалізація моделі
        nn_model = NeuralNetwork(ingredient_vocab, dish_name_vocab)

        # Тренування моделі
        nn_model.train(liked_mealplans, disliked_mealplans, epochs=1000)

        # Сортуювання раціонів
        sorted_mealplans = nn_model.sort_mealplans(unique_list)

        # Запис у файл .json
        meal_plan_path = "D:\\PersonalAssistant\\mealPlan.json"
        data_manadger.save_dishes(sorted_mealplans, meal_plan_path)

    except Exception as e:
        logger.error(f"Сталася помилка в Main: {str(e)}")

if __name__ == '__main__':
    main()

```

Б.5 Файл DailyMealPlanWindow.xaml.cs

```

using PersonalAssistant.WPF.Models;
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Media.Imaging;
using WpfAnimatedGif;

namespace PersonalAssistant.WPF
{
    public partial class DailyDietWindow : Window
    {
        // Конструктор вікна
        public DailyDietWindow()
        {
            InitializeComponent();
            // Ініціалізація кнопок для маніпулювання планами харчування
            NextRationButton.IsEnabled = false;
            PastRationButton.IsEnabled = false;
            ChooseMealPlanButton.IsEnabled = false;
            RejectMealPlanButton.IsEnabled = false;
            GifLoad();
        }

        // Список планів харчування та поточний індекс
        private List<MealPlan> mealPlansList = new List<MealPlan>();
        private int currentMealPlanIndex = 0;

        // Метод для отримання списку планів харчування з Python-скрипту

```

```

private List<MealPlan> GetDailyDietList()
{
    // Шляхи до Python і скрипту
    string pythonPath =
@"D:\PersonalAssistant\PersonalAssistant.Python\env\Scripts\python.exe";
    string scriptPath =
@"D:\PersonalAssistant\PersonalAssistant.Python\Main.py";

    // Налаштування процесу для запуску Python-скрипту
    ProcessStartInfo psi = new ProcessStartInfo
    {
        FileName = pythonPath,
        Arguments = $"{scriptPath}",
        RedirectStandardOutput = true,
        RedirectStandardError = true,
        UseShellExecute = false,
        CreateNoWindow = true
    };

    // Виконання процесу
    try
    {
        using (Process = Process.Start(psi))
        {
            // Читання виводу процесу
            string output = process.StandardOutput.ReadToEnd();
            string errors = process.StandardError.ReadToEnd();
            process.WaitForExit();

            // Виведення результатів або помилок у консоль
            Console.WriteLine("Текст консолі:");
            Console.WriteLine(output);
            if (!string.IsNullOrEmpty(errors))
            {
                Console.WriteLine("Помилка:");
                Console.WriteLine(errors);
            }
        }
    }
    catch (Exception ex)
    {
        // Обробка помилки запуску процесу
        MessageWindow message = new MessageWindow("Помилка", "Помилка
запуску процесу", ex.Message);
        message.ShowDialog();
    }

    // Завантаження планів харчування з локального файлу через
DataManager
    DataManager = new DataManager();
    List<MealPlan> mealPlans = dataManager.LoadMealPlanData();
    return mealPlans;
}

// Оновлення вікна з поточними даними
private void UpdateWindow()
{
    try
    {
        // Оновлення елементів інтерфейсу згідно з поточним планом
харчування

```

```

MealTotalCaloriesLabel.Content = "Загальна кількість калорій:
" + mealPlansList[currentMealPlanIndex].TotalCalories.ToString();

// Ініціалізація індексів для кожного прийому їжі
int BreakfastIndex = -1;
int Snacks1Index = -1;
int LunchIndex = -1;
int Snacks2Index = -1;
int DinnerIndex = -1;

// Визначення індексів в залежності від кількості страв
switch
(mealPlansList[currentMealPlanIndex].DishesWithServings.Count)
{
    case 3:
        BreakfastIndex = 0;
        LunchIndex = 1;
        DinnerIndex = 2;
        break;
    case 4:
        BreakfastIndex = 0;
        Snacks1Index = 1;
        LunchIndex = 2;
        DinnerIndex = 3;
        break;
    case 5:
        BreakfastIndex = 0;
        Snacks1Index = 1;
        LunchIndex = 2;
        Snacks2Index = 3;
        DinnerIndex = 4;
        break;
    default:
        throw new
ArgumentOutOfRangeException(nameof(currentMealPlanIndex), "Неприпустимий індекс.");
}

// Оновлення елементів UI для кожного прийому їжі (сніданок,
обід, вечеря, перекуси)
BreakfastDishNameLabel.Content = "Назва страви: " +
mealPlansList[currentMealPlanIndex].DishesWithServings[BreakfastIndex].Name;
BreakfastDishCaloriesPerServingLabel.Content = "Калорійність
на 1 порцію: " +
mealPlansList[currentMealPlanIndex].DishesWithServings[BreakfastIndex].CaloriesPerServing.ToString();
BreakfastDishIngredientsLabel.Content = "Інгредієнти: " +
string.Join(", ",
mealPlansList[currentMealPlanIndex].DishesWithServings[BreakfastIndex].Ingredients);
BreakfastDishServingsLabel.Content = "Кількість порцій: " +
mealPlansList[currentMealPlanIndex].DishesWithServings[BreakfastIndex].Servings.ToString();

LunchDishNameLabel.Content = "Назва страви: " +
mealPlansList[currentMealPlanIndex].DishesWithServings[LunchIndex].Name;
LunchDishCaloriesPerServingLabel.Content = "Калорійність на 1
порцію: " +
mealPlansList[currentMealPlanIndex].DishesWithServings[LunchIndex].CaloriesPerServing.ToString();
LunchDishIngredientsLabel.Content = "Інгредієнти: " +
string.Join(", ",
mealPlansList[currentMealPlanIndex].DishesWithServings[LunchIndex].Ingredients);

```

```

        LunchDishServingsLabel.Content = "Кількість порцій: " +
mealPlansList[currentMealPlanIndex].DishesWithServings[LunchIndex].Servings.ToString(
);

        DinnerDishNameLabel.Content = "Назва страви: " +
mealPlansList[currentMealPlanIndex].DishesWithServings[DinnerIndex].Name;
        DinnerDishCaloriesPerServingLabel.Content = "Калорійність на
1 порцію: " +
mealPlansList[currentMealPlanIndex].DishesWithServings[DinnerIndex].CaloriesPerServing.ToString();
        DinnerDishIngredientsLabel.Content = "Інгредієнти: " +
string.Join(", ",
mealPlansList[currentMealPlanIndex].DishesWithServings[DinnerIndex].Ingredients);
        DinnerDishServingsLabel.Content = "Кількість порцій: " +
mealPlansList[currentMealPlanIndex].DishesWithServings[DinnerIndex].Servings.ToString(
);

        // Обробка видимості перекусів залежно від наявності
        if (Snacks1Index == -1)
        {
            Snacks1MealNameLabel.Visibility = Visibility.Hidden;
            Snacks1DishNameLabel.Visibility = Visibility.Hidden;
            Snacks1DishCaloriesPerServingLabel.Visibility =
Visibility.Hidden;
            Snacks1DishIngredientsLabel.Visibility =
Visibility.Hidden;
            Snacks1DishServingsLabel.Visibility = Visibility.Hidden;
        }
        else
        {
            Snacks1MealNameLabel.Visibility = Visibility.Visible;
            Snacks1DishNameLabel.Visibility = Visibility.Visible;
            Snacks1DishCaloriesPerServingLabel.Visibility =
Visibility.Visible;
            Snacks1DishIngredientsLabel.Visibility =
Visibility.Visible;
            Snacks1DishServingsLabel.Visibility = Visibility.Visible;

            Snacks1DishNameLabel.Content = "Назва страви: " +
mealPlansList[currentMealPlanIndex].DishesWithServings[Snacks1Index].Name;
            Snacks1DishCaloriesPerServingLabel.Content =
"Калорійність на 1 порцію: " +
mealPlansList[currentMealPlanIndex].DishesWithServings[Snacks1Index].CaloriesPerServing.ToString();
            Snacks1DishIngredientsLabel.Content = "Інгредієнти: " +
string.Join(", ",
mealPlansList[currentMealPlanIndex].DishesWithServings[Snacks1Index].Ingredients);
            Snacks1DishServingsLabel.Content = "Кількість порцій: " +
mealPlansList[currentMealPlanIndex].DishesWithServings[Snacks1Index].Servings.ToString(
);
        }

        if (Snacks2Index == -1)
        {
            Snacks2MealNameLabel.Visibility = Visibility.Hidden;
            Snacks2DishNameLabel.Visibility = Visibility.Hidden;
            Snacks2DishCaloriesPerServingLabel.Visibility =
Visibility.Hidden;
            Snacks2DishIngredientsLabel.Visibility =
Visibility.Hidden;
            Snacks2DishServingsLabel.Visibility = Visibility.Hidden;
        }
    }
}

```

```

else
{
    Snacks2MealNameLabel.Visibility = Visibility.Visible;
    Snacks2DishNameLabel.Visibility = Visibility.Visible;
    Snacks2DishCaloriesPerServingLabel.Visibility =
Visibility.Visible;
    Snacks2DishIngredientsLabel.Visibility =
Visibility.Visible;
    Snacks2DishServingsLabel.Visibility = Visibility.Visible;

    Snacks2DishNameLabel.Content = "Назва страви: " +
mealPlansList[currentMealPlanIndex].DishesWithServings[Snacks2Index].Name;
    Snacks2DishCaloriesPerServingLabel.Content =
"Калорійність на 1 порцію: " +
mealPlansList[currentMealPlanIndex].DishesWithServings[Snacks2Index].CaloriesPerServing.ToString();
    Snacks2DishIngredientsLabel.Content = "Інгредієнти: " +
string.Join(", ",
mealPlansList[currentMealPlanIndex].DishesWithServings[Snacks2Index].Ingredients);
    Snacks2DishServingsLabel.Content = "Кількість порцій: " +
mealPlansList[currentMealPlanIndex].DishesWithServings[Snacks2Index].Servings.ToString();
}
}
catch (Exception ex)
{
    // Виведення повідомлення про помилку при оновленні
    MessageWindow message = new MessageWindow("Помилка", "Помилка
оновлення вікна", ex.Message);
    message.ShowDialog();
}

// Обробник натискання кнопки для генерації нового плану харчування
private async void GenerateMealPlanButton_Click(object sender,
RoutedEventArgs e)
{
    try
    {
        NextRationButton.IsEnabled = true;
        PastRationButton.IsEnabled = true;
        ChooseMealPlanButton.IsEnabled = true;
        RejectMealPlanButton.IsEnabled = true;
        LoadGifBox.Visibility = Visibility.Visible;
        mealPlansList = await Task.Run(() => GetDailyDietList());
        currentMealPlanIndex = 0;
        UpdateWindow();
        LoadGifBox.Visibility = Visibility.Hidden;
    }
    catch (Exception ex)
    {
        // Виведення повідомлення про помилку
        MessageWindow message = new MessageWindow("Помилка", "Помилка
при генерації плану", ex.Message);
        message.ShowDialog();
    }
}

// Обробник натискання кнопки для перегляду попереднього раціону
private void PastRationButton_Click(object sender, RoutedEventArgs e)
{
    if (currentMealPlanIndex == 0)

```

```

        {
            currentMealPlanIndex = mealPlansList.Count - 1;
        }
        else
        {
            currentMealPlanIndex--;
        }
        UpdateWindow();
    }

    // Обробник натискання кнопки для перегляду наступного раціону
    private void NextRationButton_Click(object sender, RoutedEventArgs e)
    {
        if (currentMealPlanIndex == mealPlansList.Count - 1)
        {
            currentMealPlanIndex = 0;
        }
        else
        {
            currentMealPlanIndex++;
        }
        UpdateWindow();
    }

    // Обробник натискання кнопки для вибору поточного плану
    private void ChooseMealPlanButton_Click(object sender,
    RoutedEventArgs e)
    {
        DataManager = new DataManager();
        // Додавання обраного плану харчування до JSON

        dataManager.AddMealPlanToJson("D:\\PersonalAssistant\\likedData.json",
        mealPlansList[currentMealPlanIndex]);
        // Вимкнення кнопок після вибору
        NextRationButton.IsEnabled = false;
        PastRationButton.IsEnabled = false;
        ChooseMealPlanButton.IsEnabled = false;
        RejectMealPlanButton.IsEnabled = false;
    }

    // Обробник натискання кнопки для відхилення поточного плану
    private void RejectMealPlanButton_Click(object sender,
    RoutedEventArgs e)
    {
        DataManager = new DataManager();
        // Додавання відхиленого плану до JSON

        dataManager.AddMealPlanToJson("D:\\PersonalAssistant\\dislikedData.json",
        mealPlansList[currentMealPlanIndex]);
        mealPlansList.RemoveAt(currentMealPlanIndex);
        UpdateWindow();
    }

    public async void GifLoad()
    {
        // Приховуємо GIF до завантаження
        LoadGifBox.Visibility = Visibility.Hidden;

        // Завантажуємо GIF асинхронно
        await Task.Run(() =>
        {
            // Завантажуємо GIF в окремому потоці

```

```

Uri gifUri = new
Uri(@"D:\PersonalAssistant\PersonalAssistant.WPF\Image\Gif\food on a fork.gif");

// Оновлюємо UI потік після завершення завантаження
Dispatcher.Invoke(() =>
{
    // Завантажуємо анімований GIF з бібліотеки
    var gifImage = new BitmapImage(gifUri);
    ImageBehavior.SetAnimatedSource(LoadGifBox, gifImage);
});
}
}
}
}

```

ДОДАТОК В
Слайди презентації

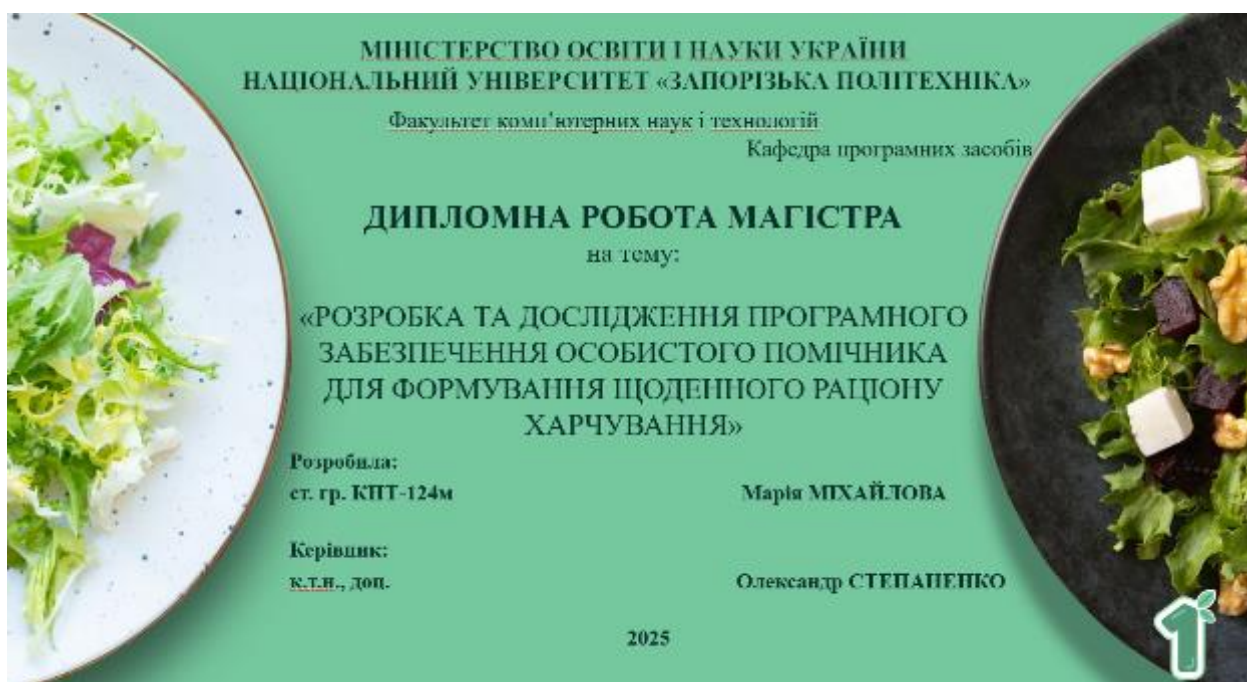


Рисунок В.1 – Перший слайд



Рисунок В.2 –Другий слайд

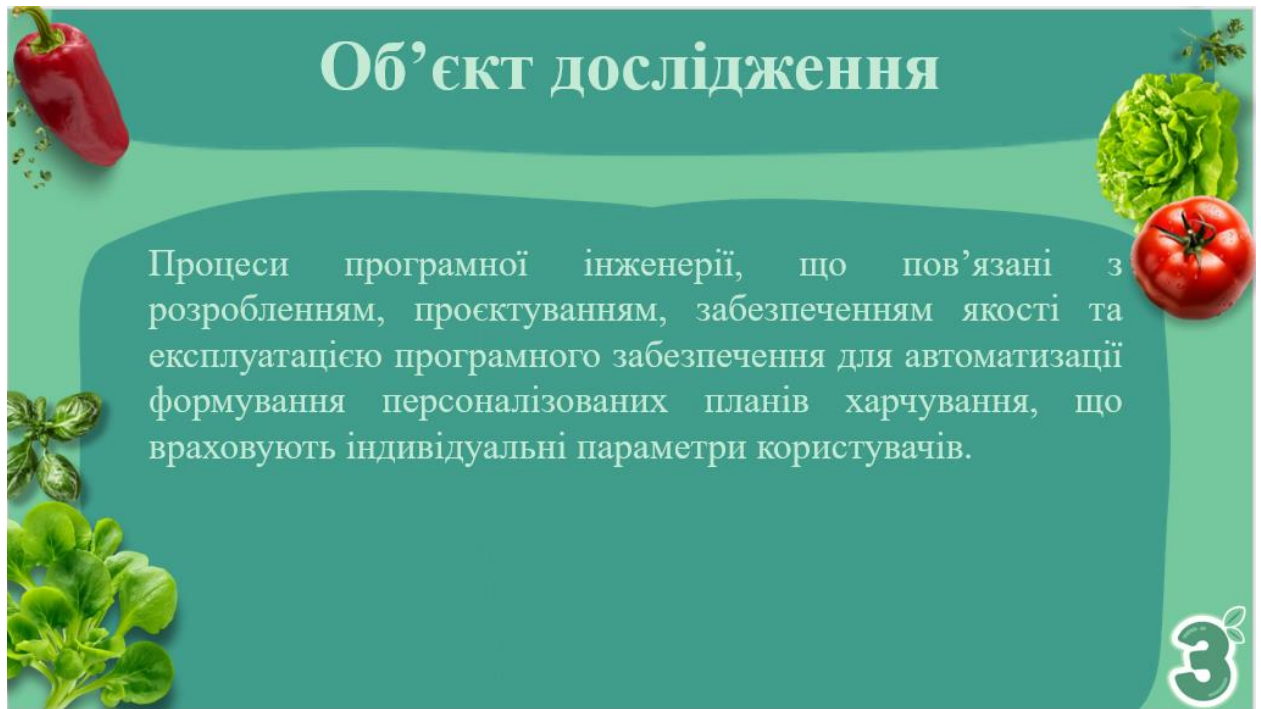


Рисунок В.3 –Третій слайд

Порівняння існуючих аналогів

Характеристика	EatMe	Samsung Food	MyFitnessPal	Eat This Much
Налаштування профілю користувача	+	+	+	+
Персоналізація харчування	+	+	+	+
Створення плану харчування	+	+	+	+
Розрахунок норми калорій на день	+	+	+	+
Алергії та вподобання	+	+	+	+
Десктопна версія	+	-	-	+
Платний контент	-	+	+	+
Україномовний інтерфейс	+	-	-	-
Зручний інтерфейс	+	Частково	Частково	-
Використання ІІІ	+	+	-	-

Рисунок В.4 –Четвертий слайд

Порівняння мов програмування

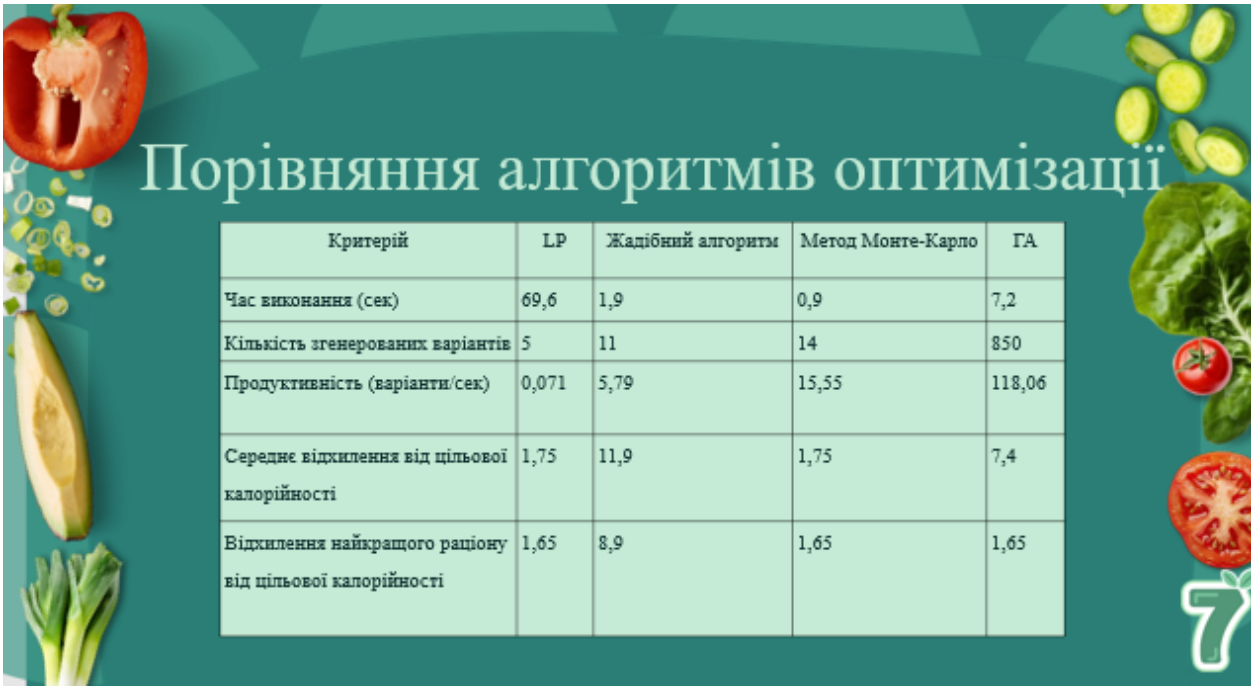
Ознака	C#	Python	Java	C++
Тип мови	Об'єктно-орієнтована, статично типізована	Високорівнева, динамічно типізована	Об'єктно-орієнтована, статично типізована	Об'єктно-орієнтована, статично типізована
Сфера застосування	Розробка інтерфейсів, десктопних додатків, веб-розробка	Штучний інтелект, наука про дані, веб-розробка	Мобільні додатки, серверні додатки	Ігри, високопродуктивні додатки
Основні переваги	Потужний для розробки інтерфейсів	Легкість у вивченні, багатий набір бібліотек для ІІІ	Підтримка багатозадачності	Висока швидкість
Основні недоліки	Обмежена кросплатформність	Не найкраща продуктивність	Не найкраща продуктивність	Складність синтаксису, висока ймовірність помилок у пам'яті
Продуктивність	Висока	Повірна	Висока	Дуже висока
Популярність	Висока, особливо для бізнес-додатків	Дуже висока, особливо в ІІІ та науці про дані	Висока, особливо для корпоративних додатків	Висока серед розробників інфраструктури та ігор

Рисунок В.5 –П'ятий слайд

Порівняння середовищ розробки

Характеристика	Microsoft Visual Studio	PyCharm	JetBrains Rider	VS Code
Підтримка Python	Вбудована через розширення	Рідна підтримка	Обмежена через плагіни	Розширення
Підтримка C#	Рідна підтримка	Обмежена через плагіни	Вбудована підтримка	Розширення
Особливості	Інтеграція з .NET, розширення для Python	Зручне середовище для Python, розширення для підтримки C#	Чудово підходить для C#, розширення для Python	Легка вага, плагіни для Python і C#
Мінуси	Великий розмір, потребує більше ресурсів	Менш підходить для великих C# проєктів	Не така зручна інтеграція для Python, платне середовище	Менше інструментів для великих проєктів

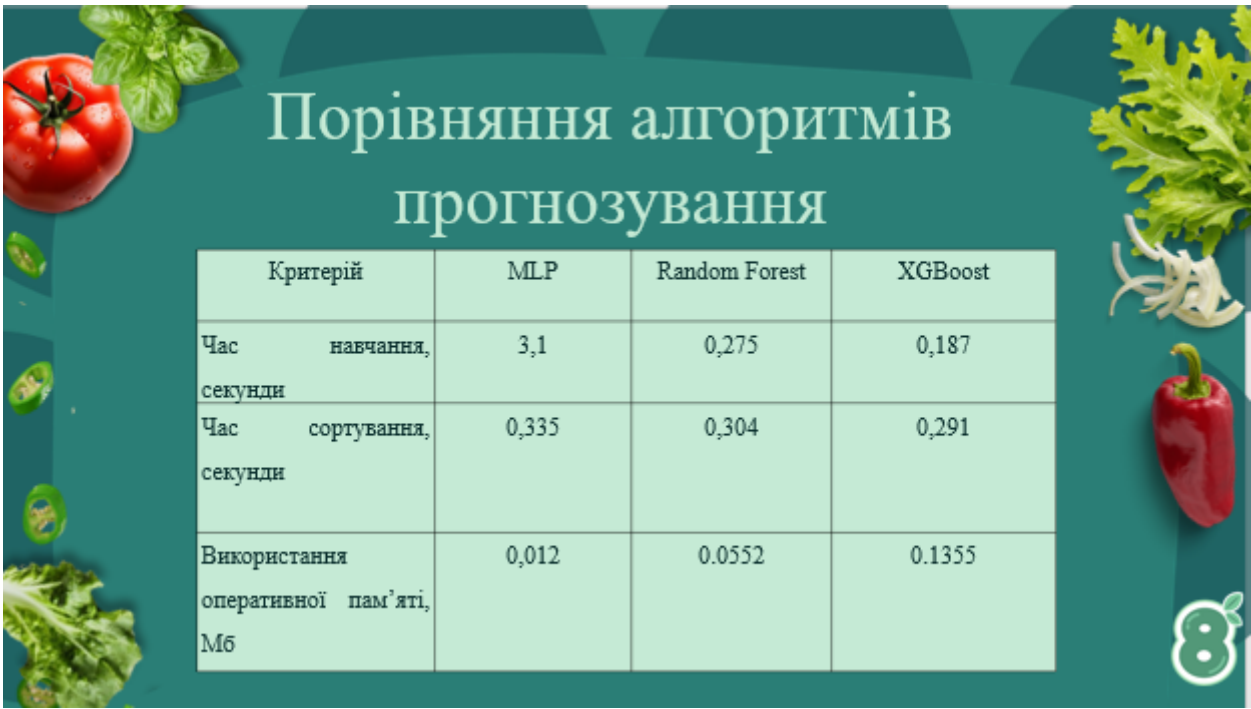
Рисунок В.6 – Шостий слайд



Порівняння алгоритмів оптимізації

Критерій	LP	Жадібний алгоритм	Метод Монте-Карло	ГА
Час виконання (сек)	69,6	1,9	0,9	7,2
Кількість згенерованих варіантів	5	11	14	850
Продуктивність (варіанти/сек)	0,071	5,79	15,55	118,06
Середнє відхилення від цільової калорійності	1,75	11,9	1,75	7,4
Відхилення найкращого раціону від цільової калорійності	1,65	8,9	1,65	1,65

Рисунок В.7 –Сьомий слайд



Порівняння алгоритмів прогнозування

Критерій	MLP	Random Forest	XGBoost
Час навчання, секунди	3,1	0,275	0,187
Час сортування, секунди	0,335	0,304	0,291
Використання оперативної пам'яті, Мб	0,012	0.0552	0.1355

Рисунок В.8 –Восьмий слайд



Рисунок В.11 – Одинадцятий слайд



Рисунок В.12 – Дванадцятий слайд



Рисунок В.13 – Тринадцятий слайд



Рисунок В.14 – Чотирнадцятий слайд



Рисунок В.15 – П'ятнадцятий слайд

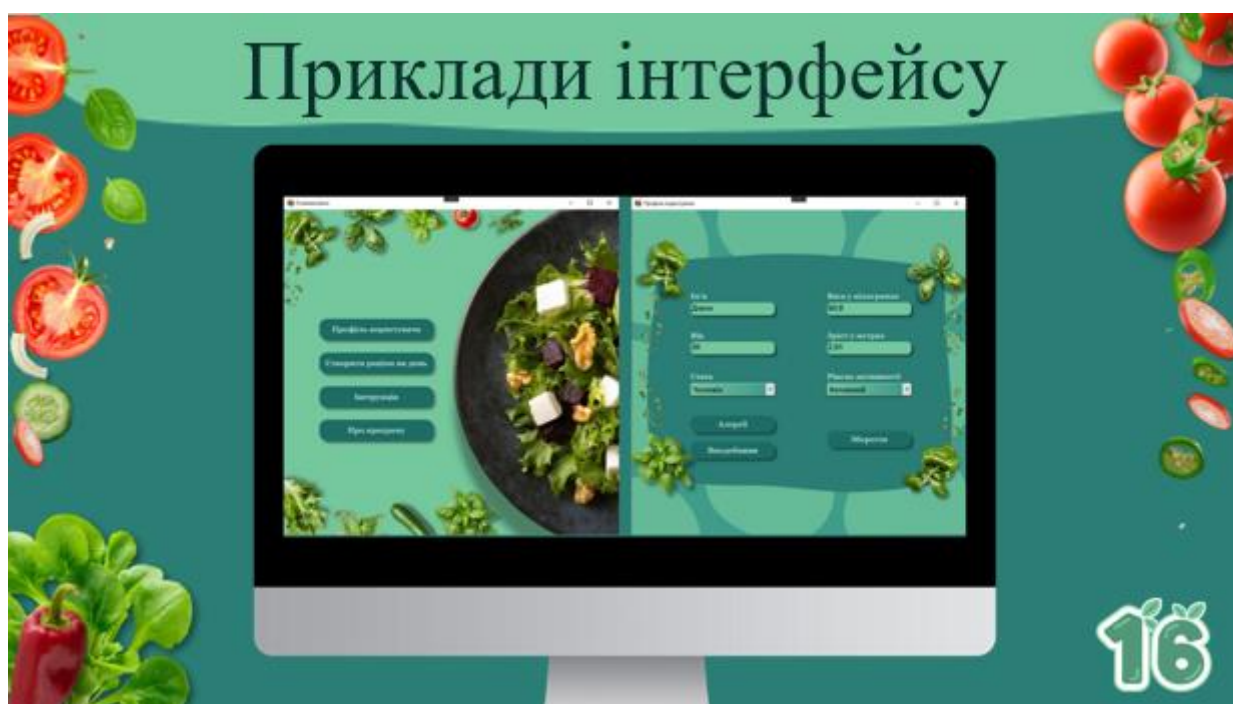


Рисунок В.16 – Шістнадцятий слайд

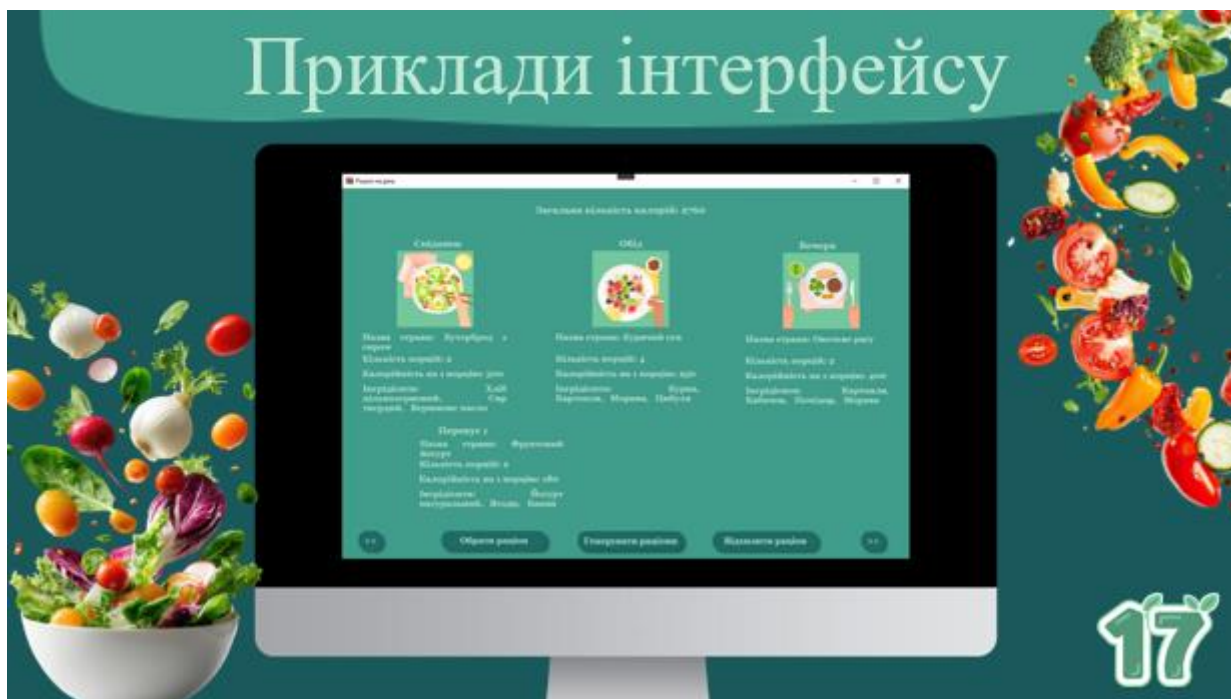


Рисунок В.17 – Сімнадцятий слайд

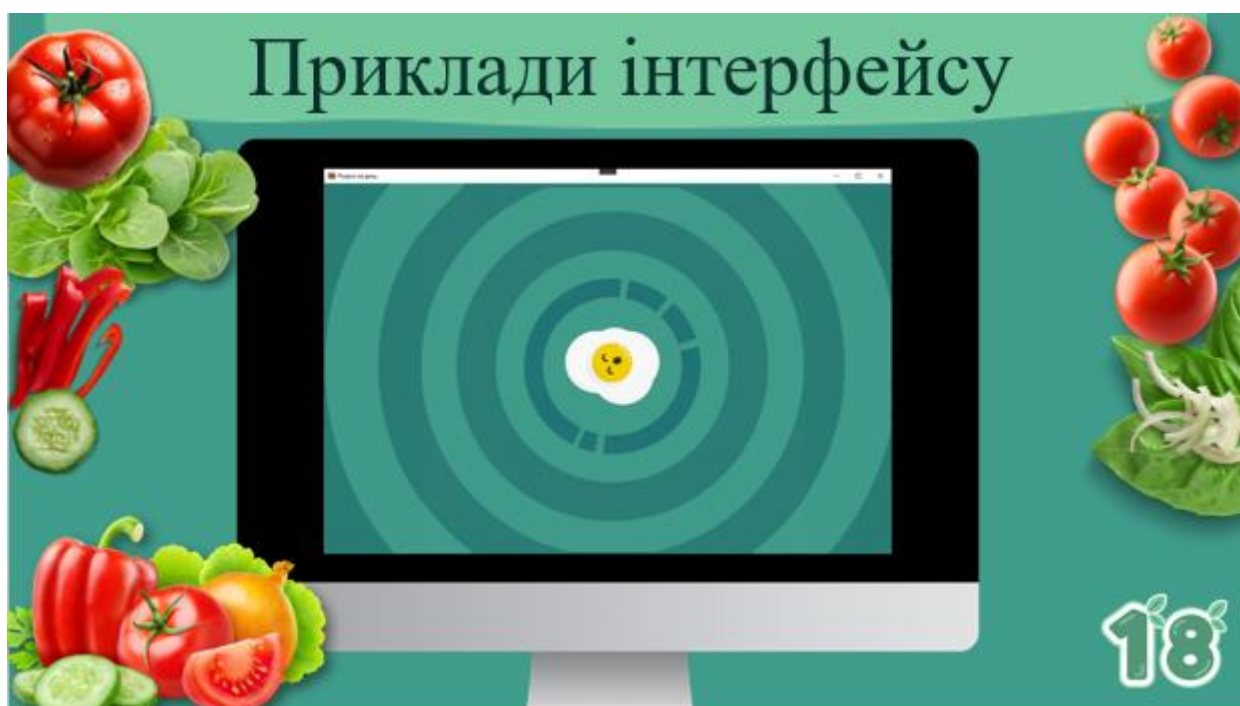


Рисунок В.18 –Вісімнадцятий слайд

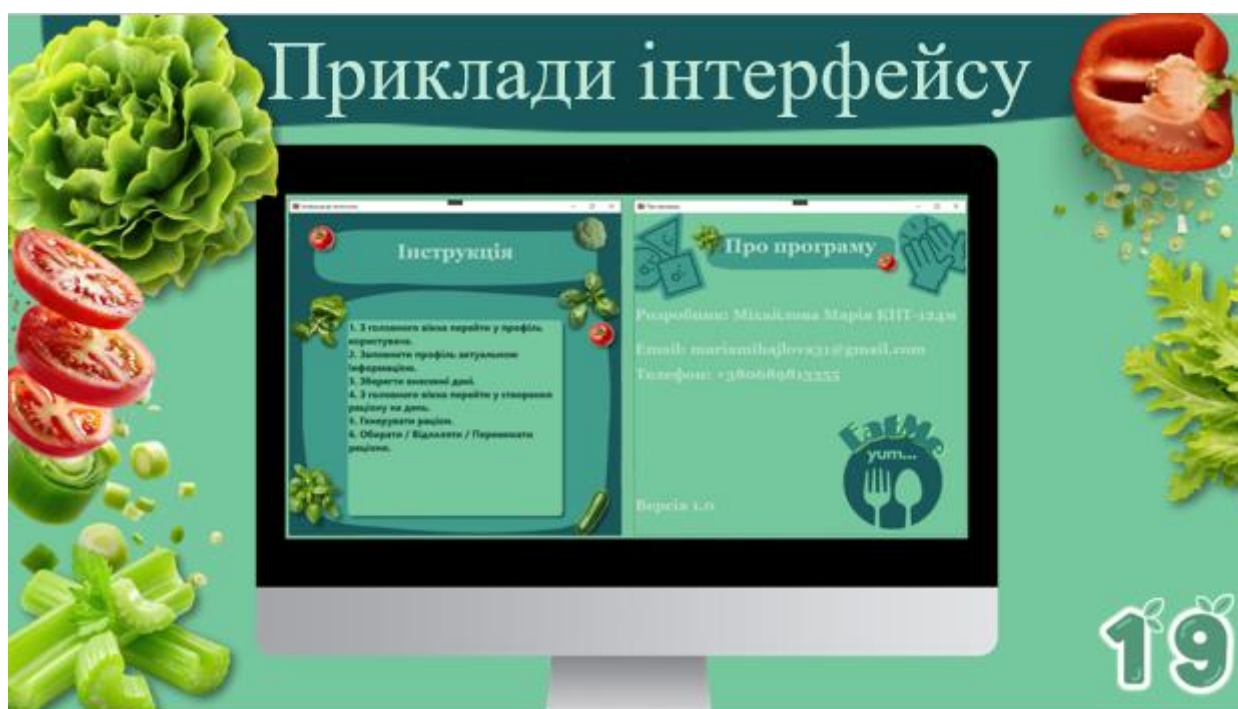


Рисунок В.19 –Дев'ятнадцятий слайд

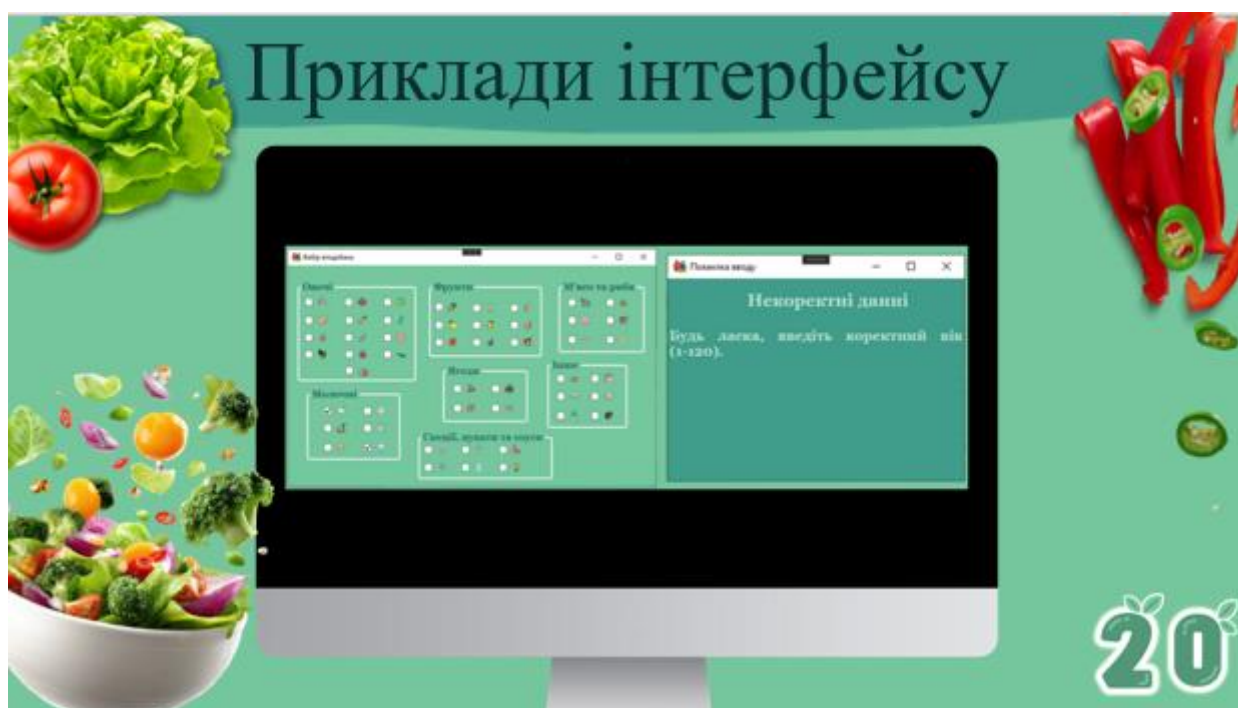


Рисунок В.20 –Двадцятий слайд



Рисунок В.21 –Двадцять перший слайд



Рисунок В.22 –Двадцять другий слайд