

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 проекту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ КЛІЄНТ-СЕРВЕРНОЇ СИСТЕМИ КУЛІНАРНИХ РЕЦЕПТІВ З БЕЗПЕЧНИМ ЗБЕРІГАННЯМ ПАРОЛІВ КОРИСТУВАЧІВ
RESEARCH AND SOFTWARE IMPLEMENTATION OF A CLIENT-SERVER SYSTEM FOR CULINARY RECIPES WITH SECURE STORAGE OF USER PASSWORDS

Виконав(ла): студент(ка) 4 курсу, групи КНТ-110

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

МІХАЙЛОВА М.С.

(ПРИЗВИЩЕ та ініціали)

Керівник СТЕПАНЕНКО О.О.

(ПРИЗВИЩЕ та ініціали)

Рецензент ПОЛЯКОВ М.О.

(ПРИЗВИЩЕ та ініціали)

2024

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

Факультет ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
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
“ ” 2024 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) Дослідження та програмна реалізація клієнт-серверної системи кулінарних рецептів з безпечним зберіганням паролів користувачів. Research and Software Implementation of a Client-Server System for Culinary Recipes with Secure Storage of User Passwords

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 “24” квітня 2024 року № 168

2. Строк подання студентом проєкту (роботи) 10 червня 2024 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	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-6 Основна частина	СТЕПАНЕНКО О.О., доцент		
Нормоконтролер	ДЕЙНЕГА Л.Ю., старший викладач		

7. Дата видачі завдання “ 15 ” квітня 2024 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.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка архітектури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділ 3
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4,5,6
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

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

Керівник проєкту (роботи)

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

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 194 с., 41 табл., 58 рис., 4 дод., 45 джерел.

РЕЦЕПТ, ПРОДУКТИ, КУЛІНАРІЯ, ШИФРУВАННЯ, КЛІЄНТ-СЕРВЕРНА АРХІТЕКТУРА, C#, VISUAL STUDIO 2019, WPF, WCF, SHA-256.

Об'єкт дослідження – розробка десктопного програмного застосунку з кулінарними рецептами з використанням криптографічного шифрування паролів.

Предмет дослідження – використання хеш-функцій у розробці програмного застосунку для шифрування паролів.

Метою роботи є створення функціонального та інтуїтивно зрозумілого застосунку з рецептами та шифруванням паролю з метою забезпечення високого рівня безпеки, який надасть користувачам зручність, вдосконалений пошук, організацію та обмін кулінарними рецептами.

Матеріали, методи та технічні засоби: мова програмування C#, середовище розробки Microsoft Visual Studio 2019 з графічною підсистемою WPF, СКБД Microsoft SQL Server, клієнт-серверна архітектура, персональний комп'ютер з процесором Intel Core 7 Duo під управлінням операційної системи Microsoft Windows 10.

Результати. Розроблено програмне забезпечення, яке надає користувачу зручний інтерфейс для організації роботи з рецептами, що дозволяє легко зберігати, організовувати та швидко знаходити рецепти будь-яких страв за допомогою вдосконаленого пошуку. Також використано сучасні методи шифрування для захисту особистої інформації користувачів, такої як паролі.

Висновки. Розроблене програмне рішення «Ресріє» має зручний та зрозумілий інтерфейс, забезпечує достатній рівень безпеки і допомагає краще організувати роботу з рецептами.

Сфера застосування – використання у будь-якій професійній діяльності та у власних інтересах.

ABSTRACT

Explanatory note to the diploma qualifying work of the paper: 194 pages, 58 figures, 41 tables, 4 appendixes, 45 sources.

RECIPE, INGREDIENTS, CULINARY, ENCRYPTION, CLIENT-SERVER ARCHITECTURE, C#, VS 2019, WPF, WCF, SHA-256.

Research Object – development of a desktop application for culinary recipes with the use of cryptographic password encryption.

Subject of Research – the use of hash functions in the development of a software application for password encryption.

The aim of the work is to create a functional and user-friendly application with recipes and password encryption to ensure a high level of security, providing users with convenience, improved search, organization, and exchange of culinary recipes.

Materials, Methods, and Technical Means: C# programming language, Microsoft Visual Studio 2019 development environment with the WPF graphical subsystem, Microsoft SQL Server database management system, client-server architecture, personal computer with an Intel Core i7 Duo processor running the Microsoft Windows 10 operating system.

The results. Developed software provides users with a user-friendly interface for organizing recipes, allowing for easy storage, organization, and quick retrieval of recipes. Additionally, modern encryption methods are employed to protect users' personal information, such as passwords.

Conclusions. The developed software solution, «Recipe» has a convenient and understandable interface, ensures an adequate level of security, and helps better organize work with recipes.

Application Area. Can be used in any professional activity and for personal interests.

ЗМІСТ

	С.
Перелік умовних позначень, символів, одиниць, скорочень і термінів	11
Вступ.....	12
1 Аналіз предметної області.....	13
1.1 Огляд існуючих методів вирішення завдання	14
1.2 Огляд існуючих програмних засобів, що вирішують аналогічні завдання	15
1.2.1 Сервіс «Allrecipes».....	15
1.2.2 Сервіс «Smachno.ua».....	18
1.2.3 Сервіс «1000.menu»	21
1.2.4 Порівняльна таблиця існуючих та розроблюваного програмних продуктів.....	23
1.3 Порівняння методів шифрування пароллю	25
1.4 Постановка завдання роботи.....	28
1.5 Висновки за розділом 1	29
2 Вибір та обґрунтування структури системи, яка проектується.....	30
2.1 Вибір інструментарію розробки програмного забезпечення	30
2.1.1 Огляд особливостей мови програмування	30
2.1.2 Огляд особливостей обраного середовища розробки	37
2.1.3 Огляд особливостей обраної системи управління базами даних	41
2.2 Структурна схема програми	44
2.3 Опис модулів розроблюваної системи.....	46
2.4 Висновки за розділом 2	48
3 Основні рішення щодо реалізації компонентів системи.....	49
3.1 Алгоритм функціонування розробленої програми.....	49
3.1.1 Алгоритм реєстрації	50
3.1.2 Алгоритм авторизації	51
3.1.3 Алгоритм «Розрахування калорійності»	52
3.1.4 Алгоритм сортування за популярністю.....	53

3.1.5 Алгоритм сортування за датою	54
3.1.6 Алгоритм пошуку за продуктами.....	54
3.1.7 Алгоритм пошуку за назвою.....	55
3.1.8 Алгоритм «Коментування рецепту»	56
3.1.9 Алгоритм «Редагування рецепту».....	57
3.1.10 Алгоритм «Видалення рецепту»	58
3.1.11 Алгоритм «Створення рецепту».....	59
3.1.12 Алгоритм «Додавання рецепту до улюблених»	59
3.1.13 Алгоритм «Видалення рецепту з улюблених».....	60
3.1.14 Алгоритм «Сортування за популярністю»	61
3.1.15 Алгоритм шифрації.....	61
3.2 Вибір шаблону проектування	63
3.3 Опис модулів розроблюваної системи.....	65
3.3.1 Клас About	69
3.3.2 Клас Authorization	69
3.3.3 Клас CalorieCalculator.....	70
3.3.4 Клас ProductBoxItem.....	71
3.3.5 Клас CreateRecipe.....	72
3.3.6 Клас Favorite	73
3.3.7 Клас ImageByte.....	77
3.3.8 Клас Information	78
3.3.9 Клас MainWindow	78
3.3.10 Клас Products	82
3.3.11 Клас Recipe	83
3.3.12 Клас RecipeMessageBox	85
3.3.13 Клас Registration.....	85
3.3.14 Клас Search	86
3.3.15 Клас ClientException	90
3.3.16 Клас ProductObject	91
3.3.17 Клас RecipeObject.....	91

3.3.18 Клас RecpieDataBase.....	92
3.3.19 Клас ServerUser	94
3.3.20 Клас ServiceRecpie	94
3.3.21 Інтерфейс IServiceRecpie.....	96
3.3.22 Інтерфейс IServerRecpieCallback.....	97
3.3.23 Клас Program.....	97
3.3.24 Клас RecpieHashing.....	97
3.3.25 Клас SHA256Hash	98
3.4 Проектування бази даних.....	99
3.5 Основні рішення щодо розробки інтерфейсу.....	102
4 Керівництво програміста.....	114
4.1 Призначення та умови застосування програми	114
4.1.1 Призначення програми.....	114
4.1.2 Функції програми.....	114
4.1.3 Умови застосування програми для клієнтської та серверної частини	115
4.2 Характеристика програми	115
4.2.1 Структура програми	116
4.3 Звертання до програми	117
4.4 Вхідні та вихідні дані.....	117
4.4.1 Вхідні дані	117
4.4.2 Вихідні дані	117
4.5 Повідомлення	118
5 Керівництво оператора	119
5.1 Призначення програми	119
5.2 Умови виконання програми	119
5.2.1 Апаратні вимоги програми	119
5.2.2 Вимоги до операторів.....	119
5.3 Виконання програми.....	120
5.3.1 Запуск програми.....	120

	10
5.3.2 Виконання роботи з програмою	120
5.4 Повідомлення оператору	129
6 Тестування і результати.....	130
6.2 Результати використання розробленого програмного забезпечення ...	132
6.2.1 Реєстрація акаунту	132
6.2.2 Авторизація	132
6.2.3 Головне вікно	133
6.2.4 Вікно з рецептом	133
6.2.5 Вікно створення рецепту.....	134
6.2.6 Вікно «Мої та улюблені»	134
6.2.7 Вікно з розширеним пошуком.....	135
6.2.8 Вікно «Калькулятор калорій».....	136
Висновки	137
Перелік джерел посилання	138
Додаток А Технічне завдання	142
Додаток Б Опис програми	149
Додаток В Текст програми	157
Додаток Г Слайди презентації	184

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД – бази даних;

ООП – об'єктно-орієнтоване програмування;

ОС – операційна система;

ПЗ – програмне забезпечення;

ПК – персональний комп'ютер

СКБД – системи керування базами даних;

ТЗ – технічне завдання\$

VS – Visual Studio;

WCF – Windows Communication Foundation;

WPF – Windows Presentation Foundation.

ВСТУП

Без комп'ютерних технологій не можливо уявити сьогодення. Зі впровадженням різних інновацій та удосконалень в інформаційній системі, все більшої уваги приділяється дослідженню всіх сфер життя людини.

У сучасному світі, де цифрова безпека стає важливим аспектом нашого онлайн-життя, розробка програмного забезпечення для ефективного та безпечного управління особистими даними стає нагальною задачею. Кулінарні знання та гастрономічні досягнення займають особливе місце в повсякденному житті, тому все більше людей шукають способи організації та зберігання своїх улюблених кулінарних рецептів.

У цьому випадку потрібно розробити клієнт-серверний додаток для рецептів, який буде працювати на десктопних пристроях. Особливий акцент буде зроблено на забезпеченні безпеки паролем користувача. Таке програмне забезпечення надає зручність та ефективність в створенні, організації та пошуку рецептів. Воно допомагає користувачам зберігати, категоризувати та збирати рецепти з різних джерел в одному місці, а також забезпечує безпеку особистих даних за допомогою криптографічних методів.

Завданням дипломної роботи є реалізація програмного застосунку «Ресріє», який буде надавати користувачам безпеку особистих даних, зручний та інтуїтивно зрозумілий інтерфейс, що спрощує процес пошуку та організації рецептів, надає змогу зберігати персональні налаштування та уподобання, а також сприяє обміну рецептами з іншими користувачами.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Під час проведеного дослідження перед початком роботи було визначено, що основна проблема, яку необхідно вирішити, полягає у відсутності зручного україномовного застосунку для роботи з кулінарними рецептами, який би включав в себе механізми криптографічного шифрування паролів користувачів.

У дипломній роботі предметною областю визначено реалізацію роботи з кулінарними рецептами за допомогою десктопного клієнт-серверного застосунку.

Кулінарний рецепт – це, перш за все, текстова інструкція, яка докладно описує процес приготування страви. У рецепті надається інформація про необхідні інгредієнти, вказується кухонне приладдя, яке слід використовувати, а також детально пояснюється метод приготування. Крім того, рецепт включає часові рекомендації, щоб допомогти зрозуміти, скільки часу знадобиться для приготування страви від початку до кінця [1].

Криптографія – це наука, яка розробляє методи перетворення зрозумілої інформації в шифротекст, що є недоступним для сприйняття сторонніми особами, з метою захисту інформації від несанкціонованого доступу [2].

Шифрування – це процес, при якому захищена інформація перетворюється з відкритого тексту або даних у зашифроване повідомлення або шифрований текст. Це здійснюється за допомогою спеціальних алгоритмів і правил, визначених у шифрі [3].

Метою роботи є розроблення інтуїтивно зрозумілого застосунку з рецептами та шифруванням паролів з метою забезпечення високого рівня безпеки, що надає користувачам доступ до широкого вибору кулінарних рецептів, дозволяє зручно шукати, зберігати улюблені рецепти, створювати списки покупок та надає корисні функції для полегшення процесу готування їжі.

1.1 Огляд існуючих методів вирішення завдання

Для того, щоб вирішити поставлене завдання, можна використовувати такі методи:

- кулінарні книги та журнали. Традиційні кулінарні друковані видання містять багато рецептів та порад, але вони обмежені фізичним простором і можуть бути неактуальними або складними у використанні;

- вебсайти представляють собою набір взаємопов'язаних веб-сторінок, які об'єднані спільним доменним іменем та є доступними для широкого загалу [4]. Інтернет-платформи, які пропонують кулінарні рецепти, дозволяють шукати рецепти за різними категоріями, переглядати відгуки користувачів, але вони можуть бути незручними у використанні на мобільних пристроях і не мають інтегрованих функцій;

- мобільні додатки – це тип програмного забезпечення, створений для функціонування на переносних гаджетах [5]. Існують додатки для мобільних пристроїв, які надають доступ до рецептів, можливість зберігати улюблені рецепти, створювати списки покупок та навіть надають покрокові інструкції для готування. Однак, деякі додатки можуть бути обмеженими в обсязі доступних рецептів або мати обмежені функції;

- соціальні медіа та кулінарні спільноти. Платформи соціальних медіа, такі як Instagram, Pinterest або спеціалізовані кулінарні спільноти, дозволяють користувачам ділитися своїми рецептами, знаходити натхнення та обмінюватися думками з іншими кухарями. Проте, ці платформи можуть бути неструктурованими і важкодоступними для пошуку конкретних рецептів;

- десктопні застосунки це програмні продукти, що інсталюються на персональний комп'ютер або ноутбук і працюють в рамках операційної системи пристрою [6]. Десктопні застосунки встановлюються безпосередньо на комп'ютер користувача і можуть пропонувати багато функціональних можливостей для кулінарних потреб. Вони можуть мати зручний інтерфейс,

більшу ефективність та швидкодію порівняно з веб-сайтами або мобільними додатками.

1.2 Огляд існуючих програмних засобів, що вирішують аналогічні завдання

Спочатку було вивчено та проаналізовано ринок наявних сервісів і програм у відповідній предметній області. Було виявлено, що існує велика кількість аналогів, і кожен з них має свої функції, недоліки та переваги. Проаналізувавши вже існуючі аналоги, були обрані найбільш схожі за функціоналом та змістом сервіси: Allrecipes, Smachno.ua, 1000.menu.

1.2.1 Сервіс «Allrecipes»

Allrecipes є одним з найпопулярніших веб-сайтів та мобільних додатків для кулінарних рецептів. Він пропонує велику кількість рецептів, відгуки користувачів, оцінки та можливість зберігати улюблені рецепти.

За всі ці роки Allrecipes реалізувала багато унікальних стратегій, які допомогли їй розвиватися швидше, ніж її аналоги. На рисунку 1.1 [7] показано експоненціальне зростання сайту за останні два десятиліття [7].

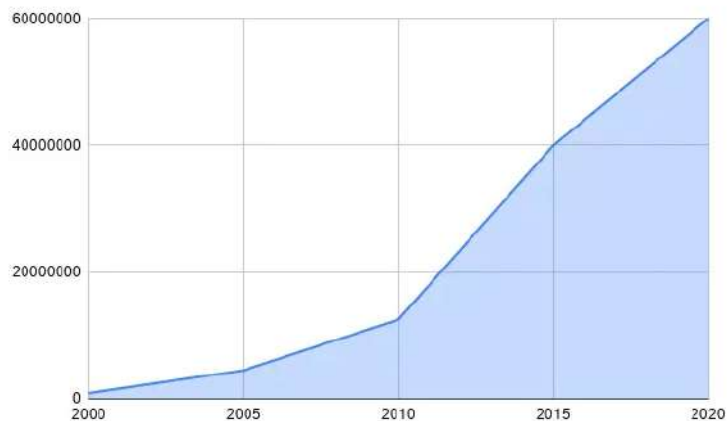


Рисунок 1.1 – Графік відвідування сайту Allrecipes [7]

Основні функції:

- пошук рецептів. Allrecipes надає потужний інструмент для пошуку рецептів. Ви можете шукати рецепти за ключовими словами, категоріями, типом страви, складниками та багатьма іншими параметрами. Це дозволяє знайти рецепти, які відповідають вашим вподобанням;
- збереження рецептів. Ви можете зберігати свої улюблені рецепти в обліковому записі Allrecipes. Це дозволяє легко знаходити їх пізніше і повторно використовувати. Ви можете створювати власні колекції рецептів, які вам сподобалися, і організовувати їх за категоріями;
- відгуки та рейтинги. Користувачі Allrecipes можуть залишати відгуки та рейтинги до рецептів, які вони спробували. Це допомагає іншим користувачам оцінити якість рецепту та поділитися своїми власними враженнями. Відгуки також можуть містити корисні поради та варіації рецепту;
- спільнота користувачів. Allrecipes має активну спільноту користувачів, де ви можете обговорювати рецепти, ділитися своїми власними ідеями та отримувати поради від інших любителів готування. Це створює можливість обміну досвідом та збагачує вашу кулінарну подорож;
- персоналізовані рекомендації. Allrecipes надає персоналізовані рекомендації рецептів на основі вашої активності та вподобань. Ви отримуватимете пропозиції щодо нових рецептів, які можуть вас зацікавити, або варіацій рецептів, які ви вже спробували.

Сьогодні Allrecipes є улюбленим джерелом рецептів для 85 мільйонів домашніх кухарів. Щороку його відвідувачі переглядають 3 мільярди сторінок. Він має 19 глобальних сайтів, які обслуговують у 24 країнах 12 мовами. Щосекунди на Allrecipes переглядається 95 рецептів [7].

Робота сервісу продемонстрована на рисунку 1.2 [8].

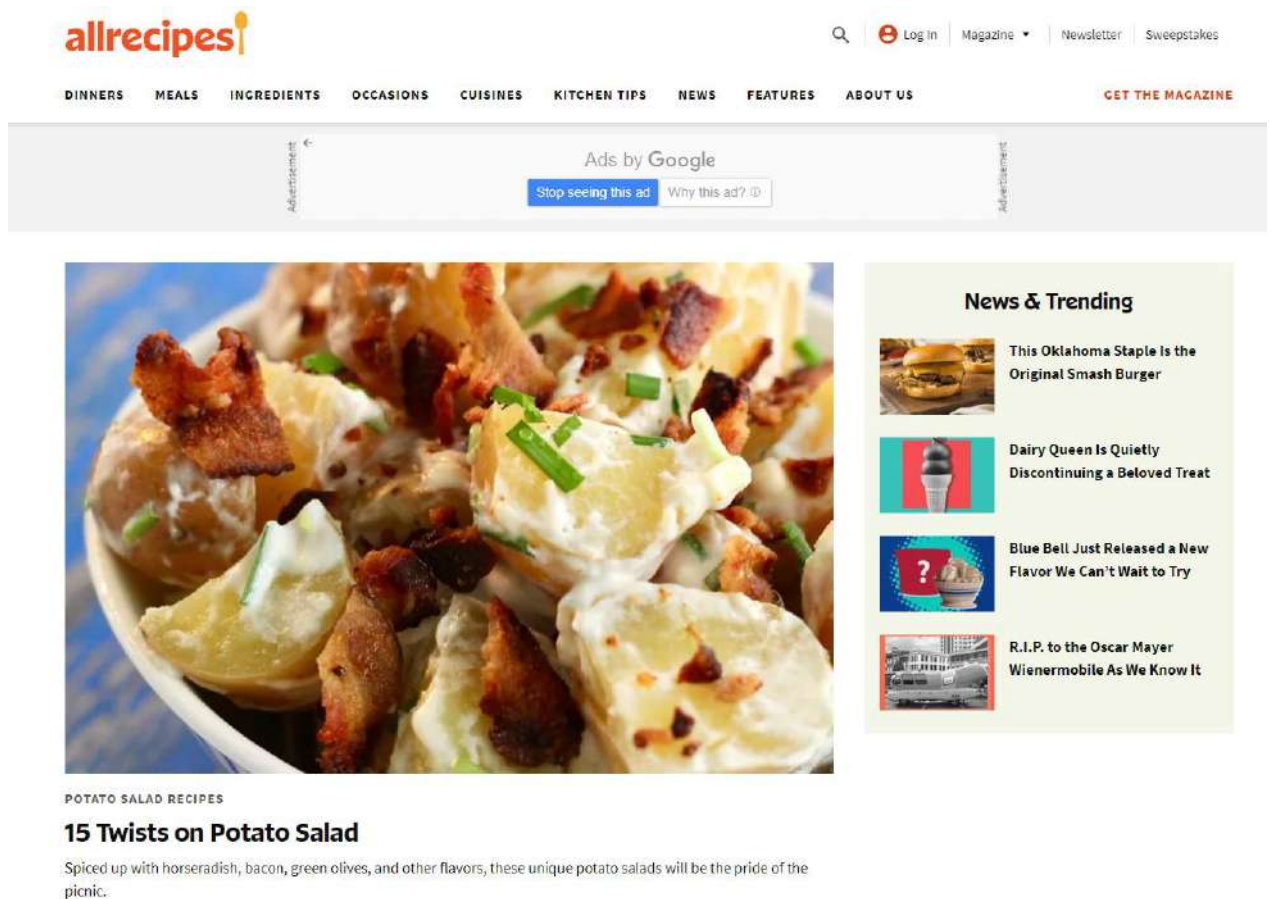


Рисунок 1.2 – Робота з сервісом «Allrecipes» [8]

Переваги:

- велика база даних рецептів. Allrecipes має вражаючу кількість рецептів з різних кухонь і категорій;
- спільнота користувачів. Allrecipes має активну спільноту користувачів, де можна отримати поради, обговорити рецепти та поділитися своїми кулінарними враженнями;
- персоналізовані рекомендації. Allrecipes надає персоналізовані рекомендації рецептів на основі вашої активності та вподобань;
- легкість використання. Платформа Allrecipes має зручний інтерфейс, що робить навігацію та пошук рецептів простими.

Недоліки Allrecipes:

- багато реклами. Allrecipes може містити багато рекламних матеріалів, які можуть бути відволікаючими під час користування платформою;

- обмежені фільтри пошуку. Деякі користувачі відзначають, що фільтри пошуку на Allrecipes можуть бути обмеженими, що може ускладнити точний пошук конкретних типів рецептів;

- якість рецептів: Оскільки рецепти на Allrecipes додаються користувачами, які не завжди є професіоналами, які перевіряються, якість деяких рецептів може варіюватися;

- немає україномовного інтерфейсу.

Відсутність авторизації на сервісі Allrecipes може вплинути на користувачів, а саме: користувачі без авторизації можуть мати обмежений доступ до функцій, таких як зберігання улюблених рецептів, створення списків покупок, відгуків тощо. Також Без авторизації користувач не може отримати персоналізований досвід, такий як рекомендації на основі попередніх взаємодій з сайтом або перегляду інших користувачів. Без авторизації може бути обмежений доступ до можливостей спілкування з іншими користувачами, таких як обговорення рецептів, обмін думками та порадами щодо приготування страв. І основне – це авторизація дозволяє забезпечити безпеку та конфіденційність даних користувачів, оскільки вони можуть мати більший контроль над своєю особистою інформацією та аккаунтом.

Загалом, Allrecipes є корисною та зручною платформою для пошуку та зберігання кулінарних рецептів, але важливо бути уважним при використанні та перевіряти якість рецептів за допомогою відгуків та рейтингів.

1.2.2 Сервіс «Smachno.ua»

Smachno.ua – це веб-ресурс із кулінарною тематикою, що відрізняється яскравим оформленням та зручним функціоналом. На сайті представлені сучасні рецепти страв, які можна приготувати з доступних продуктів, із докладним поясненням кожного кроку, супроводжувані фото та відео [9].

Основні функції:

- пошук рецептів. Smachno.ua надає зручний інструмент для пошуку рецептів. Ви можете шукати рецепти за ключовими словами, категоріями, типом страви, складниками та багатьма іншими параметрами;
- персоналізовані рекомендації. Smachno.ua аналізує вашу активність та вподобання, щоб надавати персоналізовані рекомендації;
- збереження рецептів. Ви можете зберігати свої улюблені рецепти на Smachno.ua;
- відгуки та рейтинг. Користувачі Smachno.ua можуть залишати відгуки та рейтинги до рецептів, які вони спробували;
- меню та планування страв. Smachno.ua пропонує можливість створювати персоналізовані меню та планувати страви на різні періоди часу. Ви можете створювати щоденні, тижневі або місячні меню, вибирати рецепти та автоматично генерувати список покупок;
- спільнота користувачів. Smachno.ua має активну спільноту користувачів, де можна обговорювати рецепти, ділитися враженнями, задавати питання та знаходити нові ідеї.

В сервісі немає авторизації та акаунтів у користувачів. Тобто користувачі не можуть створювати власні рецепти, додавати рецепти, які сподобались, в улюблені. На сайті додає рецепти відповідальна людина за редагування сайту. Зазвичай викладають рецепти відомих шеф-поварів. Коментувати та оцінювати рецепти можна, але треба для цього мати створену електронну пошту.

Робота сервісу продемонстрована на рисунку 1.3 [10].



Відео ▾
Рецепти ▾

Головна / Рецепти / Основні страви / Банош зі шкварками і бринзою: покроковий рецепт з відео

БАНОШ ЗІ ШКВАРКАМИ І БРИНЗОЮ: ПОКРОКОВИЙ РЕЦЕПТ З ВІДЕО

Другі страви
Українська кухня
Обід

★★★★★

Шеф-кухар пропонує спробувати справжній український банош – "багату" кукурудзяну кашу з беконом або шкварками й бринзою. Приготувати таку геть не складно, та й продуктів знадобиться не багато, але результат буде неперевершений

 легко
 30-60 хвилин

СКЛАДОВІ

Сметана жирна домашня	200 г
Молоко	700 мл
Кукурудзяна крупа	250 г
Сіль	1 ч. л.
Цукор	1 ст. л.
Бекон (сало)	200 г
Бринза	100 г



Рисунок 1.3 – Робота з сервісом «Smachno.ua» [10]

Переваги:

- широкий вибір рецептів. Smachno.ua пропонує велику базу даних з кулінарними рецептами на різний смак і захоплення;
- зручний пошук та фільтрація. Smachno.ua має зручну систему пошуку, яка дозволяє швидко знаходити рецепти за різними параметрами;
- спільнота та обмін досвідом: Smachno.ua має активну спільноту користувачів, де ви можете обговорювати рецепти, ділитися враженнями, задавати питання та знаходити нові ідеї. Викладаються рецепти відомих шеф-поварів;
- наявний україномовний інтерфейс.

Недоліки:

- відсутність деяких спеціалізованих функцій. У порівнянні з деякими іншими платформами, Smachno.ua відсутні деякі спеціалізовані

функцій, таких як калькулятор харчової цінності, власні кулінарні блоги користувачів тощо;

- відсутність можливості самостійно створювати рецепти;
- відсутність можливості додавати рецепти до улюблених;
- наявність реклами та платного контенту: Smachno.ua може містити рекламу та платний контент, який може бути роздратовуючим для деяких користувачів. Деякі функції або рецепти можуть бути доступні лише за певну плату або вимагати підписки.

1.2.3 Сервіс «1000.menu»

1000.menu є онлайн-платформою з кулінарними рецептами, яка надає користувачам доступ до широкого вибору страв та кулінарних ідей.

Основні функції:

- велика база рецептів. Платформа пропонує велику кількість рецептів різних категорій, кухонь, типів страв та складників;
- фільтрація та пошук. 1000.menu має зручну систему фільтрації та пошуку, що дозволяє користувачам швидко знайти рецепти за різними параметрами;
- оцінки та відгуки користувачів: користувачі можуть залишати оцінки та відгуки про рецепти, які вони спробували;
- плани готування. Платформа дозволяє створювати персоналізовані плани готування на різні періоди часу, наприклад, на тиждень чи місяць;
- спільнота користувачів. 1000.menu має активну спільноту користувачів, де люди можуть обмінюватися досвідом, питаннями та порадами зі світу кулінарії.

Без авторизації на сервесі Smachno.ua користувач не може мати доступ до особистого облікового запису та зберігати персоналізовані дані, такі як улюблені рецепти, списки покупок, або історія взаємодій з сайтом. Також, без авторизації може бути обмежений доступ до деяких функцій, які

доступні лише авторизованим користувачам, наприклад, коментування рецептів чи спілкування з іншими учасниками спільноти.

Робота сервісу продемонстрована на рисунку 1.4 [11].

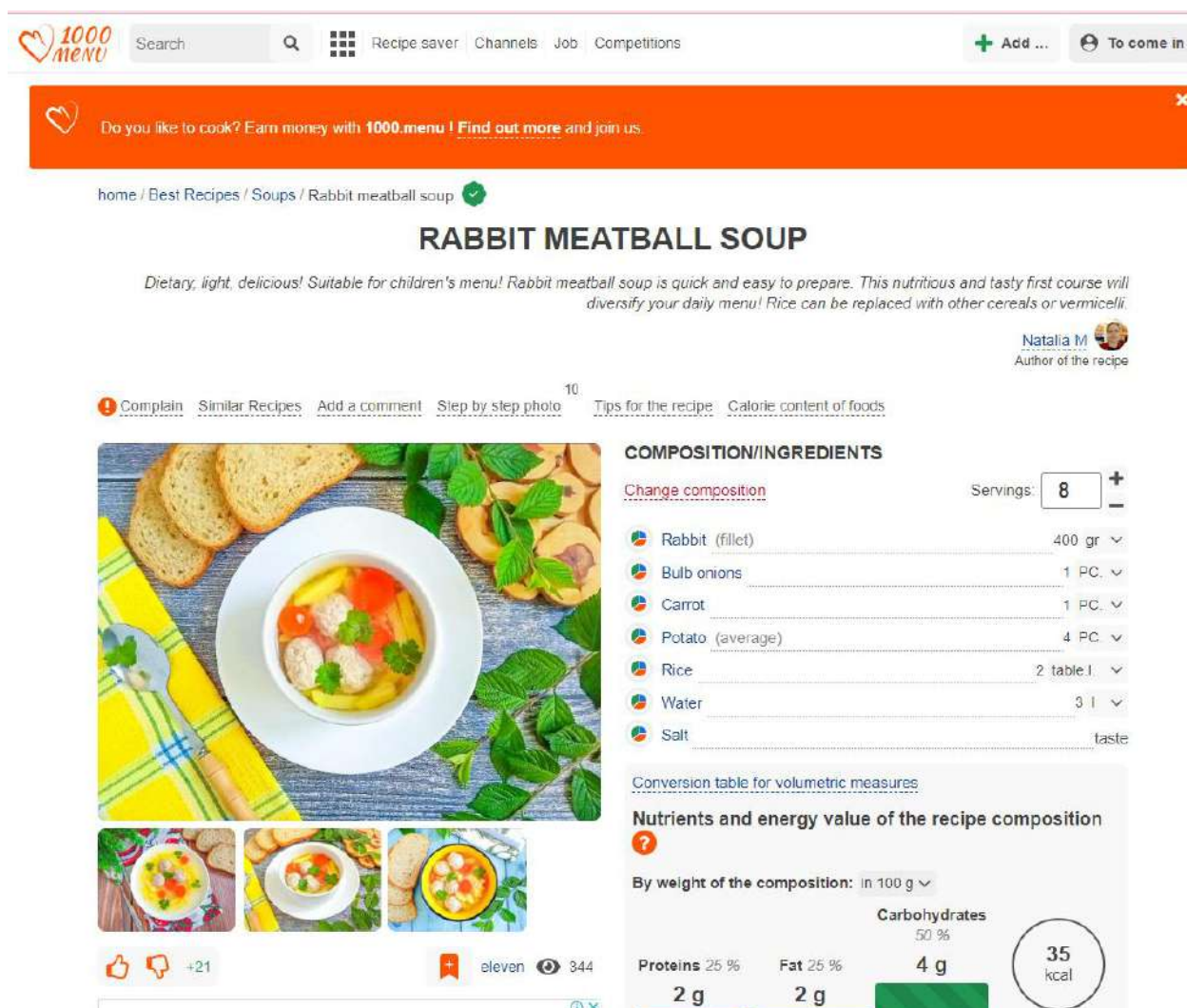


Рисунок 1.4 – Робота з сервісом «1000.menu.» [11]

Переваги:

- широкий вибір рецептів. 1000.menu пропонує велику базу рецептів різних категорій та складнощів;
- фільтрація та пошук. Платформа має зручну систему фільтрації та пошуку, що дозволяє користувачам швидко знайти рецепти за різними параметрами, такими як тип страви, складники, кухня, час приготування та інші;

- практичні поради і техніки. 1000.menu надає корисні поради та техніки, які допомагають користувачам покращити свої навички в кулінарії;
- оцінки та відгуки користувачів. Користувачі можуть залишати оцінки та відгуки про рецепти, які вони спробували;
- можливість зберігання улюблених рецептів. 1000.menu дозволяє користувачам зберігати улюблені рецепти у власному обліковому записі.

Недоліки:

- можливість застарілих рецептів. Інформація про деякі рецепти може бути застарілою або не оновлюватися;
- відсутність персоналізації. 1000.menu не завжди надає персоналізовані рекомендації користувачам;
- обмеженість розширених функцій. У порівнянні з деякими іншими платформами, 1000.menu може бути обмеженим у розширених функціях, таких як планування приготування страв, розрахунок калорійності та інші додаткові інструменти;
- можливість помилок та неточностей: Як і в будь-якій іншій платформі з рецептами, можливість помилок та неточностей не може бути повністю виключена;
- 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	Respie	1000.menu	Smachno.ua	Allrecipes
Реєстрація та авторизація користувачів	+	+	-	+
Збереження та організація рецептів	+	+	-	+
Пошук та фільтрація рецептів	+	+	+	+
Рецепти з кроками та інгредієнтами	+	+	+	+
Відображення характеристик рецептів	+	+	+	+
Обмін рецептами та спільнота	+	-	+	+
Калькулятор калорій	+	-	+	+
Україномовний інтерфейс	+	-	+	-
Зручний інтерфейс	+	+	Частково	Частково
Шифрування паролів	+	+	-	+

В результаті порівняння в таблиці 1.1 можна побачити, що застосунок «Respie» має більш широкий набір функцій порівняно з 1000.menu, який не має можливості обміну рецептами та калькулятору калорій. Водночас, Allrecipes має більш схожий набір функцій із застосунком «Respie», з певними відмінностями, такими як відсутність калькулятора калорій у Allrecipes. В той же час Smachno.ua не має авторизації та реєстрації, з цього виходить, що відсутня можливість для користувачів створювати власні рецепти. Також застосунок «Respie», як і сервіс «Smachno.ua», мають україномовний інтерфейс, на відміну від Allrecipes та 1000.menu. Як видно з таблиці 1.1 шифрування паролів є поширеною практикою у сайтах та

застосунках, які мають авторизацію. Це дозволяє забезпечити безпеку та конфіденційність даних користувачів.

1.3 Порівняння методів шифрування паролю

У сучасній цифровій епохі забезпечення безпеки та конфіденційності інформації є вельми важливим аспектом. Шифрування відіграє важливу роль у досягненні цієї мети, дозволяючи перетворювати дані в незрозумілий формат, доступний лише авторизованим користувачам. Цей процес базується на використанні криптографічних ключів та математичних алгоритмів.[12].

Симетричне шифрування використовує один і той же криптографічний ключ як для зашифрування, так і для розшифрування. Ця простота одного ключа спрощує процес взагалі [12].

Основні переваги [12]:

- простота. Використання одного ключа для обох операцій спрощує процес;
- швидкість. Алгоритми симетричного шифрування працюють швидше, ніж асиметричні;
- обчислювальна потужність. Симетричне шифрування вимагає менше обчислювальних ресурсів;
- цей вид шифрування майже не впливає на швидкість передачі даних через Інтернет.

Якщо в симетричному шифруванні використовується один ключ для шифрування і дешифрування, то в асиметричному шифруванні застосовується складніший підхід – кілька математично взаємопов'язаних ключів.

Асиметричне шифрування має кілька переваг [12]:

- підвищений рівень безпеки. Використання відкритого ключа для шифрування та відповідного закритого для розшифрування забезпечує високий ступінь захисту;

- забезпечує можливість створення зашифрованих з'єднань без необхідності обміну ключами, що спрощує процес;
- автентифікація. Використання відкритого ключа гарантує, що тільки призначений одержувач, що володіє відповідним закритим ключем, зможе розшифрувати дані та отримати до них доступ.

У комп'ютерних науках давно використовуються хеш-функції. Вони представляють собою математичні чи інші функції, які приймають на вхід рядок змінної довжини (відомий як прообраз) і перетворюють його у рядок фіксованої, зазвичай меншої, довжини (званий значенням хеш-функції). Одним із прикладів простої хеш-функції може бути функція, яка отримує прообраз і повертає байт, який представляє собою XOR всіх вхідних байтів. Головна ідея хеш-функції полягає в тому, щоб отримати характерну ознаку, звану значенням хеш-функції, яка представляє прообраз та використовується для аналізу різних прообразів при розв'язанні оберненої задачі. Оскільки хеш-функція зазвичай встановлює відношення «багато до одного», неможливо з гарантією стверджувати, що два рядки однакові. Проте їх можна використовувати для отримання прийнятної оцінки точності [13].

Хеш-функція – це тип (комп'ютерної) функції, що здійснює перетворення даних будь-якого розміру (наприклад, текстового рядка, файлу) у дані фіксованого розміру (наприклад, трьохзначне число) [14].

Криптографічна хеш-функція переваги [14]:

- кожен раз, коли подається те ж саме повідомлення, результатом буде той самий вихідний хеш;
- неможливо відновити вхідне повідомлення з його вихідного хеш-значення, окрім застосування грубої сили;
- неможливо знайти два різних вхідних повідомлення, які мають однаковий вихідний хеш;
- навіть невелика зміна вхідного повідомлення призводить до великої зміни вихідного значення хешу, що робить новий хеш значно відмінним від попереднього.

У таблиці 1.2 було порівняно декілька методів шифрування, які було розглянуто.

Таблиця 1.2 – Порівняння методів шифрування

Характеристика	Симетричне шифрування	Асиметричне шифрування	Шифрування з використанням хеш-функції
Основний принцип	Двостороннє перетворення	Двостороннє перетворення	Одностороннє перетворення
Використання ключів	Один ключ для шифрування і розшифрування	Пара ключів публічний і приватний.	Немає ключів
Швидкість	Висока	Менш висока	Висока
Використання для шифрування паролів	Використовується	Не використовується	Використовується
Приклади алгоритмів	AES, DES, 3DES	RSA, ECC, Diffie-Hellman	SHA-256, MD5, bcrypt

В результаті порівняння в таблиці можна побачити, що асиметричне шифрування зазвичай не використовується у роботі з паролями через більш повільну швидкодію ніж інші методи. При цьому шифрування з використанням хеш-функції та симетричне шифрування мають часто приблизно однакову швидкість роботи. Також видно, що шифрування з використанням хеш-функції має лише одностороннє перетворення, це робить його привабливим для зберігання паролів бо таке перетворення унеможливорює відновлення оригінального паролю злоумисниками.

1.4 Постановка завдання роботи

Підчас аналізу технічного завдання (ТЗ) було виявлено основні функції, які повинно виконувати програмне забезпечення, а саме:

- реєстрація та авторизація користувачів: Застосунок повинен мати можливість реєстрації нових користувачів і авторизації існуючих. Це дозволить зберігати та відновлювати персональні дані, а також забезпечить доступ до індивідуальних налаштувань;

- шифрування паролів користувачів: Застосунок повинен шифрувати паролі від акаунтів користувачів перед збереженням їх у базі даних за для підвищення безпеки зберігання конфіденційних даних;

- збереження та організація рецептів. Застосунок повинен надавати можливість додавати, редагувати, видаляти рецепти та додавати до улюблених;

- пошук та фільтрація рецептів. Застосунок повинен мати можливість швидкого та ефективного пошуку рецептів залежно від різних параметрів, таких як назва, інгредієнти, тип страви. Користувачі повинні мати можливість сортувати рецепти за популярністю та датою;

- рецепти з кроками та інгредієнтами. Застосунок повинен дозволити додавати докладні кроки готування для кожного рецепту, а також перелік необхідних інгредієнтів з відповідними кількостями. Це допоможе користувачам зрозуміти процес готування та виконати його успішно;

- відображення характеристик рецептів. Застосунок повинен надавати користувачам інформацію про час приготування, кількість порцій та інші характеристики рецепту. Це дозволить користувачам швидко оцінити відповідність рецепту їхнім потребам та обмеженням;

- обмін рецептами та спільнота. Користувачі повинні мати можливість ділитися своїми рецептами, залишати коментарі та відгуки;

- калькулятор калорій. Застосунок повинен давати можливість користувачам розраховувати калорійність страви.

1.5 Висновки за розділом 1

В результаті було здійснено:

- аналіз предметної області;
- огляд існуючих методів вирішення завдання;
- огляд та порівняння існуючих аналогів;
- огляд та порівняння існуючих методів шифрування;
- постановка завдання роботи.

Для ґрунтовного опрацювання різних аналогів було проаналізовано декілька основних: 1000.menu, Smachno.ua та Allrecipes.

Кожен аналог має свої переваги та недоліки, вивчення яких допомогло обрати прототип програмного рішення «Респіє». Таким чином прототипом було обрано – Allrecipes, як найбільш схожий ще функціоналом сервіс з рецептами.

Після аналізу порівняних методів шифрування, було обрано метод шифрування за допомоги хеш-функції, як найбільш відповідний до потреб застосунку в плані шифрування паролів від акаунтів зареєстрованих користувачів.

2 ВИБІР ТА ОБҐРУНТУВАННЯ СТРУКТУРИ СИСТЕМИ, ЯКА ПРОЕКТУЄТЬСЯ

2.1 Вибір інструментарію розробки програмного забезпечення

На сучасному етапі інформатизації всіх сфер життя, перед людством постають нові задачі, які вимагають доступних та ефективних рішень. В даному розділі розглянуті основні особливості програмних засобів, за допомогою яких реалізовано дипломну роботу. В процесі виконання дипломної роботи було обрано середовище розробки, мову програмування, систему керування базами даних (СКБД) та описано основні модулі. Було визначено основні переваги обраних програмних засобів, які найбільше підходять для виконання поставлених задач та цілей програмного рішення «Ресріє». Для вибору були враховані такі критерії: зручність, функціональність, доступність, зрозумілість, легкість у використанні.

2.1.1 Огляд особливостей мови програмування

Мова програмування - це система інструкцій, які програміст складає для передачі комп'ютеру з метою виконання певних завдань. Цей набір інструкцій, який зазвичай представлений у вигляді набору коду, організований відповідно до конкретного синтаксису мови програмування [15].

Обираючи мову програмування для розробки застосунку постає питання, яка з мов є найкращою. Існує багато критеріїв вибору. Один із суттєвих критеріїв – це частота використання мови у глобальній спільноті розробників.

Наразі існує велика кількість індексів та рейтингів, які використовуються для визначення популярності мов програмування. Кожен з цих рейтингів може використовувати власні критерії та методи для аналізу та ранжування мов.

Індекс TIOBE (The Importance of Being Earnest), також відомий як TIOBE Index, є інструментом для визначення популярності мов програмування. Цей індекс визначається на підставі публічно доступної інформації, зокрема, кількості пошукових запитань в Інтернеті, пов'язаних із певною мовою програмування [16].

Індекс визначається на основі обчислення популярності мов програмування шляхом аналізу кількості пошукових запитань для кожної мови у популярних пошукових системах, таких як Google, Bing, Yahoo!, Wikipedia та інші [16].

Популярність визначається на основі величини, яку називають «рейтингом TIOBE». Цей рейтинг вказує на частоту пошукових запитань, пов'язаних із конкретною мовою програмування.

Індекс враховує широкий спектр мов програмування, включаючи популярні, такі як Java, C, Python, JavaScript, а також менш популярні мови. Рейтинг TIOBE оновлюється щомісячно, що дозволяє відстежувати тенденції та зміни у популярності мов програмування.

На рисунку 2.1 [17] показано графіки популярності мов програмування з 2002 по 2023 рік.

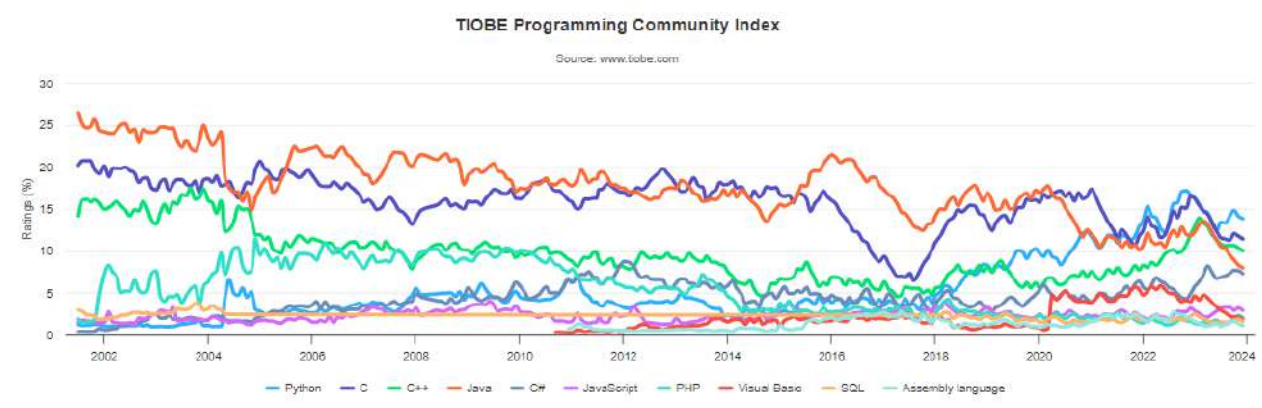


Рисунок 2.1 – Індекс TIOBE [17]

На цьому рисунку можна побачити графіки популярності таких мов: Python, C, C++, Java, C#, JavaScript та інші.

Для того, щоб робити висновки про популярність мови програмування в поточний момент, можна використати таблицю на рисунку 2.2 [17].

грудень 2023 р	грудень 2022 р	Зміна	Мова програмування	Рейтинги	Зміна
1	1		 Python	13,86%	-2,80%
2	2		 C	11,44%	-5,12%
3	3		 C++	10,01%	-1,92%
4	4		 Java	7,99%	-3,83%
5	5		 C#	7,30%	+2,38%

Рисунок 2.2 – Рейтинг найпопулярніших мов програмування TIOBE

Наприкінці 2022 року здавалося, що C# стане мовою програмування того року. Але в останній момент несподівано C++ отримала це звання. Цього року саме мова C# може отримати це звання [17].

За 2023 рік C# отримала +2,38%, тоді як його найближчі конкуренти в п'ятірці популярних мов мають зниження індексу. Може здатися трохи дивним, що більшість із 20 найпопулярніших мов у 2023 році втратили популярність. Відповідь криється в довгому хвості, де знаходяться всі нові мови. Усі вони трохи просуваються вгору та наближаються до популярних мов. Також з рисунку можна побачити що різниця між мовами Java та C# стала менше 1% [17].

На рисунку 2.3 [18] можна побачити окремо графік популярності мови C#. На цьому рисунку можна побачити що в середині 2022 року, відбувся спад популярності, але вже весною 2023 почався різкий підйом. Цей аналіз говорить про те, що мова не втратила своєї популярності, а навпаки залишається в п'ятірці найкращих.

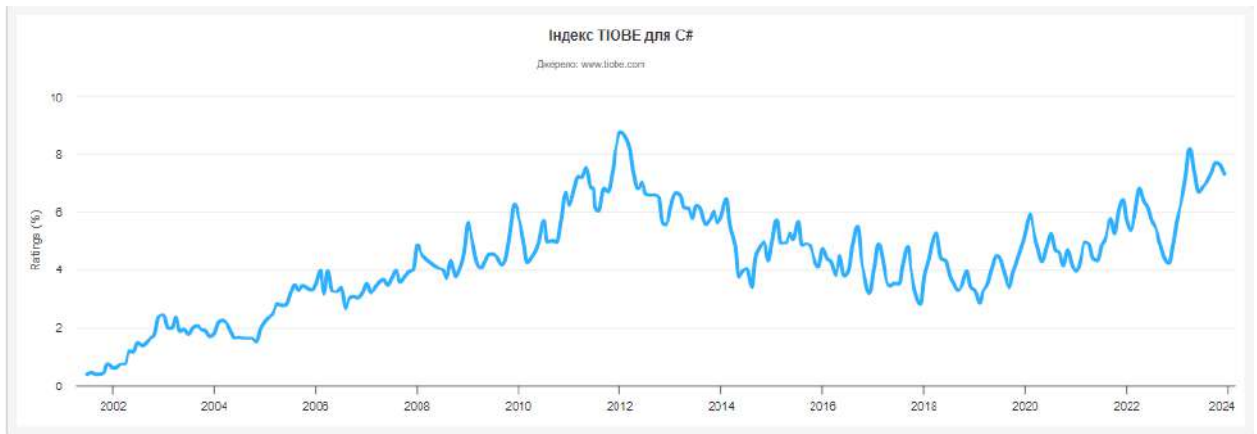


Рисунок 2.3 – Індекс ТІОВЕ для мови С#

Також для аналізу популярності мов можна використати річний опитувальник розробників, який проводить Stack Overflow [19]. Це опитування є важливим джерелом інформації про тенденції, вибір технологій, популярність мов програмування та інші аспекти в ІТ-сфері.

Як видно з рисунку 2.4 [19] С# є бажаною мовою для 21,30% опитувачів, також 62,87 % хочуть використовувати цю мову і в наступному році. Це доволі високий показник, наприклад інша популярна мова С++ має менший показник зацікавленості.

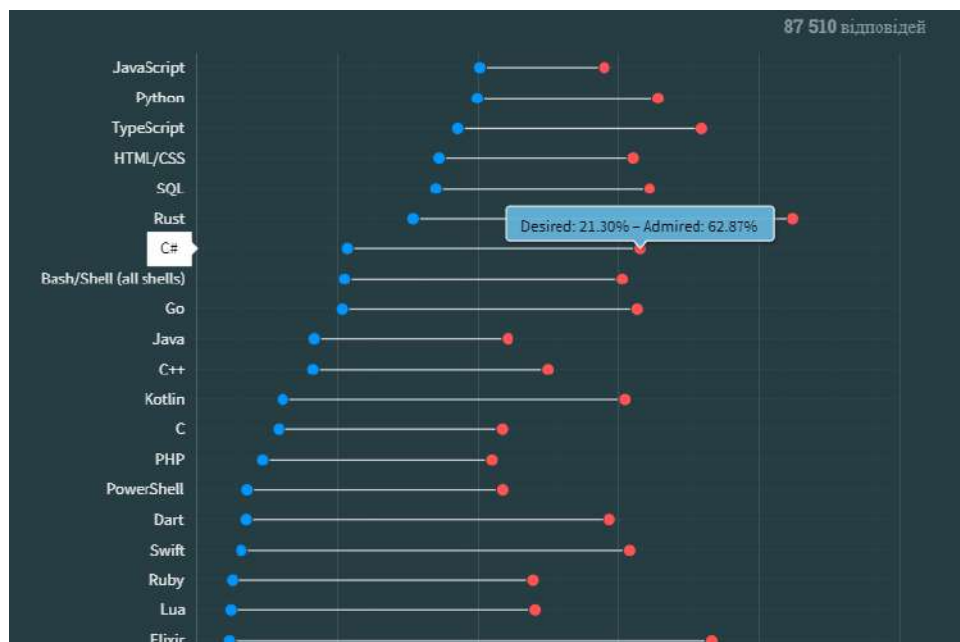


Рисунок 2.4 – Популярність мов програмування за версією Stack Overflow [19]

GitHub Octoverse – це річний звіт та аналіз від GitHub, одного з найбільших веб-платформ для управління проектами на основі системи контролю версій Git. Звіт Octoverse надає інформацію про тенденції, популярність мов програмування, фреймворків та інших технологічних аспектів на платформі GitHub.

З рисунку 2.5 [20] можна побачити з 2020 року С# залишається в п'ятірці популярних мов програмування, витісняючи в цьому топі С++.

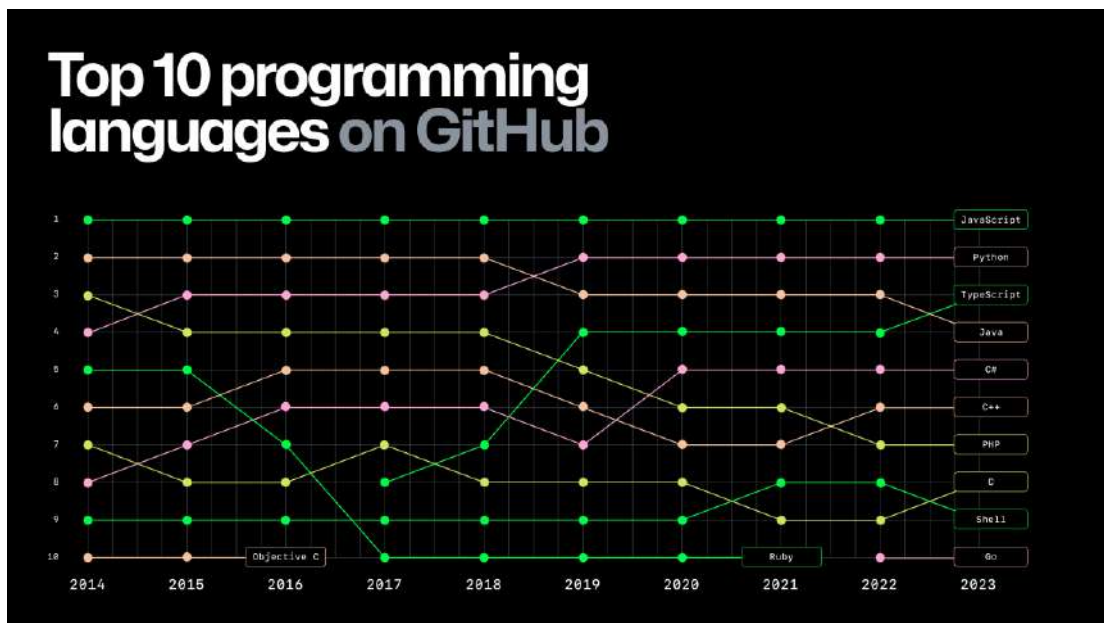


Рисунок 2.5 – Популярність мов за версією GitHub Octoverse [20]

Як мову програмування було обрано С#, оскільки вона ідеально підходить для розробки програмного забезпечення, є популярною серед сучасних розробників і надає широкі можливості для створення зручного та інтуїтивно-зрозумілого інтерфейсу.

Ця мова використовується для створення десктопних програм [21].

Мова програмування С# – це мова, яка об'єднує в собі кілька парадигм, зокрема об'єктно-орієнтовану та компонентно-орієнтовану, і характеризується строгою типизацією. Вона була розроблена для використання на платформі .NET Framework [22].

Об'єктно-орієнтоване програмування (ООП) – це методика програмування, що розглядає програму як набір взаємодіючих об'єктів. Загальний підхід до програмування полягає у відтворенні в мові програмування певних процесів реального світу. Основна перевага об'єктно-орієнтованого підходу полягає в тому, що він дозволяє структуру програми більш точно відтворити ту частину реальності, яку програма моделює [22].

Це одна з парадигм програмування, яке складають три загальні властивості:

- інкапсуляція – це властивість, яка забезпечує об'єднання даних та методів в класі та приховання деталей реалізації від користувача;
- спадкування – це властивість, що дозволяє створювати новий клас на основі вже існуючого класу;
- поліморфізм – це властивість, яка дозволяє одному інтерфейсу отримувати доступ до спільного набору дій для класу. Об'єктно-орієнтована мова програмування – це мова, побудована на принципах об'єктно-орієнтованого програмування, яка об'єднує в собі методи та поля.

Основні переваги об'єктно-орієнтованого програмування включають [22]:

- зрозуміліші програми, оскільки вони краще відображають реальний світ;
- краща структурованість програми;
- більший рівень можливостей для повторного використання коду;
- зручна підтримка програми, що дозволяє внести зміни в процесі експлуатації для адаптації до нових умов;
- зменшення ризиків при проектуванні складних систем.

Мова програмування C# має багато спільного з Java. У 1995 році компанія Sun Microsystems створила Java, базуючись на граматиці C++. Одна з її ключових особливостей полягала в тому, що програма спочатку перетворювалася на байт-код перед тим, як стати машинним кодом. Це робило Java незалежною від апаратної платформи. Крім того, розробники

додали підтримку вебу, щоб дозволити взаємодію з Інтернетом: створювати клієнт-серверні програми, отримувати та обмінюватися даними через мережу [21].

Microsoft була змушена конкурувати з Sun і одночасно працювати з вебom. У Microsoft на той момент було написано мільйони рядків коду в їхніх внутрішніх мовах програмування (C і C++). Тому перехід на Java був неможливим без значних труднощів [21].

Microsoft вирішила скористатися своєю впливовою позицією на ринку, створивши свій варіант Java – мову програмування, яка повністю належить корпорації. Ця нова мова – C#. Вона отримала від мови Java ідеї віртуальної машини (середовище .NET), байт-коду (MSIL) і вищого рівня безпеки вихідного коду програм, а також враховувала досвід використання мови Java для написання програм [23].

Унікальністю C# стала його здатність до більш простої взаємодії з кодом, написаним на інших мовах програмування, що є важливим аспектом для розробки обширних проєктів [21].

Переваги мови програмування C# [24]:

- доступність до великої кількості бібліотек та шаблонів, що дозволяє уникнути зайвого витрачання часу на створення власних шаблонів чи бібліотек;
- строга типізація, яка допомагає уникнути помилок;
- надійність та елегантність – різноманітність синтаксичних конструкцій;
- мова входить у число лідерів на ринку та має великий попит;
- C# створена з метою надати розробникові максимальний контроль над усіма аспектами виконання програми;
- мова продовжує вдосконалюватись, нові функції та можливості створюються набагато раніше, ніж у інших мовах програмування;
- вона є об'єктно-орієнтованою, простою, а водночас потужною.

Саме через велику кількість переваги та популярність у використанні під час розроблення десктопних застосунків було обрано мову програмування C#.

2.1.2 Огляд особливостей обраного середовища розробки

Інтегроване середовище розробки (IDE) — це пакет програмного забезпечення, який об'єднує основні інструменти, необхідні для написання та тестування програмного забезпечення [25].

Компанія JetBrains проводить щорічне дослідження, яке включає в себе рейтинг популярності ІСР серед розробників. Це дослідження має на меті вивчити та зрозуміти тенденції у сфері розробки програмного забезпечення та виявити популярні інструменти серед розробників

З рисунку 2.6 [26] можна побачити, що з 2020 року по сьогоднішній час Visual Studio залишається найбільш популярним IDE при розробці на C#.

Який IDE/редактор ви найчастіше використовуєте для розробки на C#?



Рисунок 2.6 – Рейтинг популярність IDE за версією JetBrains [26]

У таблиці 2.1 було порівняно декілька середовищ розробки для роботи з мовою C#.

Таблиця 2.1 – Порівняння середовищ розробки

Характеристика	Microsoft Visual Studio	JetBrains Rider	Visual Studio Code
Підтримка мови C#	+	-	+
Підтримка WCF	+	Частково	-
Інтегрована розробка	+	+	-
Керування проектами	+	+	+
Рефакторинг	+	+	Частково
Відлагодження	+	+	Частково
Розширені можливості	+	-	+
Підтримка платформи	Windows	Windows/Linux	Windows/Linux/Mac
Вартість	Частково платно	Платно	Безкоштовно

Згідно з таблицею, Microsoft Visual Studio має широкий набір функцій та особливостей, що робить його найкращим варіантом для розробки на мові C# та роботи з Windows Communication Foundation (WCF). Воно підтримує повну функціональність мови C# і надає інтегроване середовище розробки, що спрощує процес створення та керування проектами. Крім того, Microsoft Visual Studio має підтримку WCF, що дозволяє розробникам створювати та розгортати розподілені застосунки з використанням цієї технології. Інші середовища, такі як JetBrains Rider і Visual Studio Code, також мають певні функціональні можливості для роботи з C#, але вони не надають повної підтримки WCF та інтегрованого середовища розробки, що робить Microsoft Visual Studio більш привабливим варіантом для роботи з цими технологіями.

Для розробки програмного продукту було обрано середовище Microsoft Visual Studio, оскільки воно надає багато функцій і можливостей для роботи з мовою програмування C#.

Microsoft Visual Studio – інтегроване середовище розробки (IDE) від компанії Microsoft, яке призначене для створення різноманітних програм, веб-сайтів, десктопних програм, клієнт-серверних програм і мобільних додатків. Воно використовує платформи для розробки програмного забезпечення Microsoft, такі як: Windows API, Windows Forms, Windows Presentation Foundation, Windows Store і Microsoft Silverlight. Visual Studio може бути використаний для написання як власного, так і керованого коду [27].

.NET Framework – це платформа для розробки програмного забезпечення, яка здебільшого функціонує на операційній системі Microsoft Windows. Вона включає в себе обширну бібліотеку класів і підтримує різні мови програмування, що дозволяє здійснювати міжмовну інтеграцію (тобто, будь-яка мова може використовувати код, написаний іншими мовами). Бібліотека .NET доступна для всіх мов програмування, які сумісні з платформою .NET.

.NET Framework – це інноваційна платформа, розроблена Microsoft для створення програм. Ця платформа спроектована таким чином, що дозволяє використовувати її з будь-якою мовою програмування, включаючи C#, C++, Visual Basic, JScript, а також старіші мови [28].

.NET і Visual Studio дозволяють розробляти сучасні бізнес-додатки для Windows, орієнтовані на обробку даних за допомогою платформи Windows Forms, WPF та інших.

Windows Presentation Foundation (WPF) – це революційна підсистема для роботи з графікою, яка була розроблена Microsoft для створення інтерфейсу в програмних продуктах під ОС Windows. Вона входить до складу .NET Framework і надає єдиний стандарт програмування для розробки

потужних додатків для настільних комп'ютерів під управлінням Windows [29].

Переваги WPF [30]:

- WPF пропонує гнучкий підхід до розміщення елементів керування, де вони автоматично адаптуються до вмісту. Це дозволяє створювати інтерфейси, які легко адаптуються для відображення різноманітного контенту та мають відмінну відзвучивість;
- замість простого малювання пікселів, WPF використовує примітиви та об'єкти, що дозволяють створювати більш складні графічні ефекти. Також доступні функції анімації, які можна додавати до елементів інтерфейсу;
- WPF надає розширені можливості для відображення та форматування тексту. Це включає різноманітні стилізації тексту, можливість використання різних шрифтів та розміщення тексту в складних макетах;
- WPF дозволяє вбудовувати аудіо- та відеофайли безпосередньо у користувацький інтерфейс. Це відкриває можливості для створення багатофункціональних додатків з мультимедійним контентом;
- WPF дозволяє стандартизувати форматування елементів інтерфейсу за допомогою стилів та змінювати їх зовнішній вигляд за допомогою шаблонів. Це спрощує розробку та підтримку великих проєктів;
- WPF включає в себе механізм команд, що спрощує взаємодію з програмою через різні елементи керування. Це дозволяє визначати команди централізовано та прив'язувати їх до різних елементів інтерфейсу;
- інтерфейс користувача в WPF може бути описаний у вигляді дескрипторів XML за допомогою мови розмітки XAML. Це робить розробку інтерфейсу більш прозорою та дозволяє відокремити дизайн від логіки програми;
- WPF надає можливість створення програм, що використовують навігаційну модель зі сторінками, подібну до веб-сторінок. Це дозволяє створювати додатки з багаторівневою навігацією та динамічним вмістом.

Microsoft Visual Studio наділений рядом корисних функцій, що спрощують роботу у цьому середовищі. Наприклад [31]:

- хвилясті лінії та швидкі дії. Ця функція позначає помилки або потенційні проблеми у коді під час його введення. Візуальні підказки дозволяють оперативно виправити проблеми, не чекаючи на виникнення помилок під час компіляції або виконання програми. При наведенні вказівника на хвилясту лінію виводяться додаткові відомості про помилку. Крім того, у полі зліва може з'являтися лампочка, яка вказує на наявність пропозицій щодо швидких дій для виправлення помилки;

- очищення коду. Одним натисканням кнопки можна відформатувати код та застосувати до нього виправлення згідно з параметрами стилю коду. Ця функція, доступна лише для коду на C#, допомагає виправити проблеми у коді перед його перевіркою;

- рефакторинг. Включає операції, такі як інтелектуальне перейменування змінних, видалення рядків коду у новий метод та зміна порядку параметрів методів;

- intellisense являє собою набір певних можливостей, що відображає відомості про код безпосередньо у редакторі та в деяких випадках автоматично створює невеликі фрагменти коду. Це вбудований редактор базової документації, що забезпечує доступ до інформації без необхідності виходити за межі середовища;

- codelens допомагає знаходити посилання на код, зміни коду, пов'язані з кодом помилки, робочі елементи, перевірки коду та модульні тести - все це, не виходячи з редактора.

2.1.3 Огляд особливостей обраної системи управління базами даних

Система керування базами даних (СКБД) – це складний комплекс програм та інструментів, які використовуються для створення, оновлення, збереження та управління базами даних. Вона діє як посередник між

користувачем та базою даних, забезпечуючи зручний та ефективний доступ до інформації. СКБД відповідає за збереження та організацію даних, а також забезпечує безпеку та цілісність інформації [32].

Система керування базами даних (СКБД) є комплексом програмних засобів, необхідних для створення, використання та підтримки баз даних. Вона дозволяє об'єднувати дані з різних джерел в одну реляційну базу даних. СКБД забезпечує можливість швидкого та ефективного оновлення даних, створення форм, запитів і звітів, що дозволяють отримувати відповіді на різні питання, здійснювати пошук необхідної інформації, аналізувати дані, а також друкувати звіти, діаграми і поштові наклейки.

Єдина база даних можлива завдяки спеціалізованим програмним продуктам, відомим як СКБД. Основна функція СКБД полягає у створенні та підтримці баз даних в актуальному стані, а також у забезпеченні їх інтеграції з прикладними програмами, що вирішують економічні завдання користувачів.

База даних (БД) – це інструмент для збору, упорядкування та зберігання інформації. Вона може включати дані про людей, магзини, книги та інше, в залежності від тематики БД. Часто бази даних розпочинаються як прості списки в програмах обробки тексту або електронних таблицях, після чого розвиваються в складніші структури для зручного зберігання та управління даними [33].

База даних – це структурована колекція даних, яка має назву і відображає інформацію про об'єкти та їх взаємозв'язки в конкретній сфері. Вона слугує як єдине місце зберігання для даних і використовується різними користувачами, забезпечуючи незалежність від конкретних програм, які їх використовують [32].

До основних можливостей СКБД можна віднести[34]:

- надання зручного інструментарію для створення баз даних за допомогою мови формування даних (DDL - Data Definition Language);

- забезпечення широких можливостей для додавання, оновлення, видалення та отримання даних з бази за допомогою мови маніпулювання даними (DML), часто відомої як мова запитів;
- гарантування контрольованого доступу до даних за допомогою системи захисту, яка забезпечує безпеку від несанкціонованого доступу, системи керування паралельною роботою програм, яка контролює одночасний доступ до бази, та системи відновлення, яка забезпечує відновлення даних до попереднього стану після будь-яких збоїв.

Microsoft SQL Server - це програмне забезпечення для керування базами даних, розроблене Microsoft. Ця система надає широкий набір інструментів для зберігання, обробки і аналізу даних в реляційному форматі [35].

Ось деякі причини, чому Microsoft SQL Server може бути найкращим вибором для цього типу застосунків [36]:

- інтеграція з Visual Studio. Microsoft SQL Server має глибоку інтеграцію з Visual Studio, що спрощує розробку та роботу з базами даних безпосередньо з відповідними інструментами та розширеннями;
- підтримка мови програмування C#. Microsoft SQL Server працює добре з C# і надає багато можливостей для зручної роботи з даними, таких як LINQ to SQL або Entity Framework;
- потужність та функціональність. Microsoft SQL Server має широкі можливості для управління даними, підтримує складні запити, транзакції, збережені процедури та індекси. Він також надає різні інструменти для моніторингу та оптимізації продуктивності бази даних;
- безпека. Microsoft SQL Server надає широкі можливості для забезпечення безпеки даних, включаючи контроль доступу, шифрування, аудит та інші механізми захисту;
- сумісність з WCF. Microsoft SQL Server добре інтегрується з WCF, що дозволяє легко обмінюватися даними між клієнтом та сервером у застосунках, побудованих з використанням WCF.

2.2 Структурна схема програми

Структурна схема – це схема, що відображає організацію та взаємозв'язки компонентів програмного продукту під час його розробки. Ця схема визначається архітектурою програмного забезпечення і зазвичай формується за допомогою методу поетапної деталізації [37].

До компонентів структурної схеми програмної системи можуть належати програми, підсистеми, бази даних, бібліотеки ресурсів та інші аналогічні елементи.

За для забезпечення кращої масштабованості та безпеки даних буде використана клієнт-серверна архітектура реалізована за допомогою WCF.

Windows Communication Foundation (WCF) – це пвний інструмент для розробки програм, орієнтованих на роботу з послугами (клієнт-серверні системи). За допомогою WCF можна передавати дані у формі асинхронних повідомлень від одного вузла служби до іншого, незалежно від їх місцезнаходження [38].

Функціональність WCF включає [38]:

- орієнтація на обслуговування, що дозволяє створювати сервіс-орієнтовані додатки, які використовують веб-сервіси для обміну даними;
- різноманітні шаблони повідомлень для обміну даними у різних паттернах;
- підтримка метаданих служби, яка дозволяє публікувати метадані з використанням стандартів, таких як WSDL, XML Schema і WS-Policy, для автоматичного створення клієнтів;
- використання контрактів даних для зручної роботи з методами, що вимагаються для .NET Framework;
- забезпечення безпеки повідомлень шляхом шифрування і аутентифікації користувачів;

- підтримка різних транспортних протоколів і кодувань для надсилання повідомлень;
- надійний обмін повідомленнями з використанням надійних сеансів і «міцних» повідомлень, які зберігаються в базі даних для відновлення після збоїв.

Фреймворк Windows Communication Foundation (WCF) відкриває безліч можливостей для розробки розподілених систем та програм. Перш за все, він надає потужні інструменти для побудови сервіс-орієнтованих додатків, що дозволяє реалізувати ефективну та масштабовану комунікацію між різними компонентами програми або між різними програмами. Це особливо важливо в сучасному світі, де додатки все частіше стикаються з необхідністю обміну даними через мережу, і де швидкість та безпека цього обміну є критичними аспектами.

Крім того, WCF забезпечує високий рівень надійності, безпеки та масштабованості. Завдяки підтримці різних протоколів, методів кодування та механізмів перенесення, він дозволяє створювати додатки, які працюють надійно навіть у складних умовах мережі. Безпека є ще однією важливою перевагою WCF, оскільки він надає засоби для захисту конфіденційності та автентифікації користувачів, що є критичними аспектами в розподілених системах. У всьому цьому полягає важливість використання Windows Communication Foundation для розробки сучасних додатків та систем.

Структурну схему розроблюємого програмного забезпечення представлено на рисунку 2.7

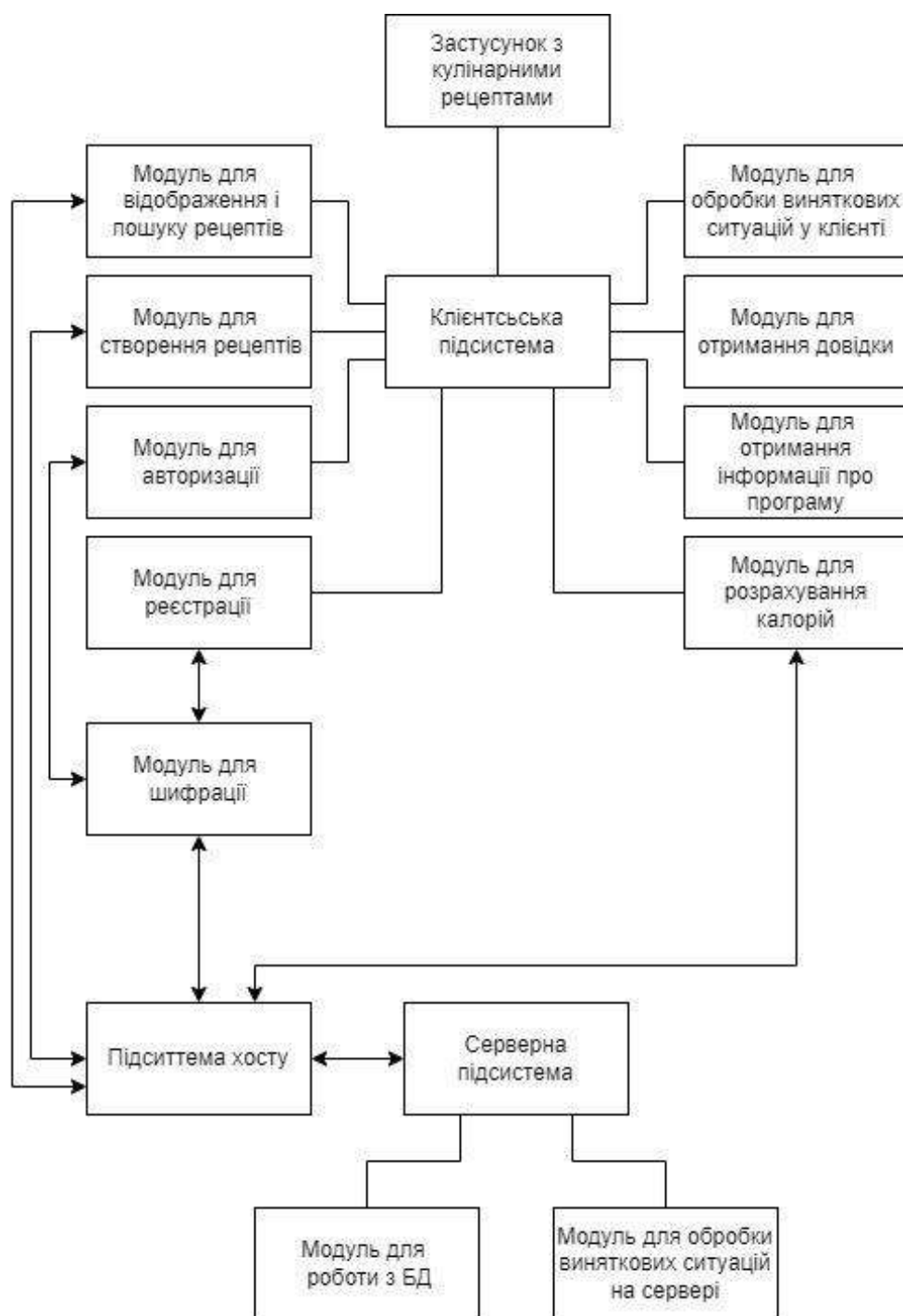


Рисунок 2.7 – Структурна схема проєкту

2.3 Опис модулів розроблюваної системи

У таблиці 2.2 наведено опис модулів реалізованого ПЗ з визначенням їх основних функцій.

Таблиця 2.2 – Опис модулів реалізованого ПЗ з визначенням основних функцій

Модуль	Опис
Відображення і пошуку рецептів	Цей модуль відповідає за відображення рецептів, улюблених рецептів та фільтрування списків відображуваних рецептів за назвою, типом страви, продуктом, популярністю, датою завантаження рецепту
Створення рецептів	Цей модуль відповідає за створення об'єкта рецепту та передання його на сервер для подальшого запису
Авторизації	Цей модуль відповідає за авторизацію існуючих користувачів та надає доступ до індивідуальних налаштувань
Реєстрації	Цей модуль відповідає за реєстрацію нових користувачів та авторизацію існуючих
Шифрація	Цей модуль відповідає за шифрацію введеного користувачем пароля
Обробки виняткових ситуацій у клієнті	Цей модуль відповідає за обробку виняткових ситуацій у клієнтській частині застосунку та сповіщає користувача про проблему, якщо це необхідно.
Отримання довідки	Цей модуль відповідає за надання довідки користувачу
Отримання інформації про програму	Цей модуль відповідає за надання інформації про програму користувачу
Розрахування калорій	Цей модуль відповідає за розрахування калорій для обраних продуктів

Продовження таблиці 2.2

Роботи з БД	Цей модуль відповідає за роботу з базою даних, внесення даних та отримання з неї даних
Обробки виняткових ситуацій на сервері	Цей модуль відповідає за обробку виняткових ситуацій у серверній частині застосунку та робить записи про них.

2.4 Висновки за розділом 2

В результаті було:

- обрано мову програмування C#;
- обрано середовище розробки Microsoft Visual Studio та порівняно з іншими;
- обрано СКБД Microsoft SQL Server;
- створено структурну схему проекту;
- описано модулі розроблюваної системи.

Отже, було розроблена теоретична модель для програмного забезпечення «Ресріє».

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Алгоритм функціонування розробленої програми

Функціональна схема надає опис взаємодії різних компонентів програми та шляхів. Вона вказує на типи даних, що обробляються, та ресурси, які використовуються у цьому процесі. Це допомагає зрозуміти структуру та логіку роботи програми [37]. Функціональну схему розробленого програмного забезпечення представлено на рисунку 3.1.

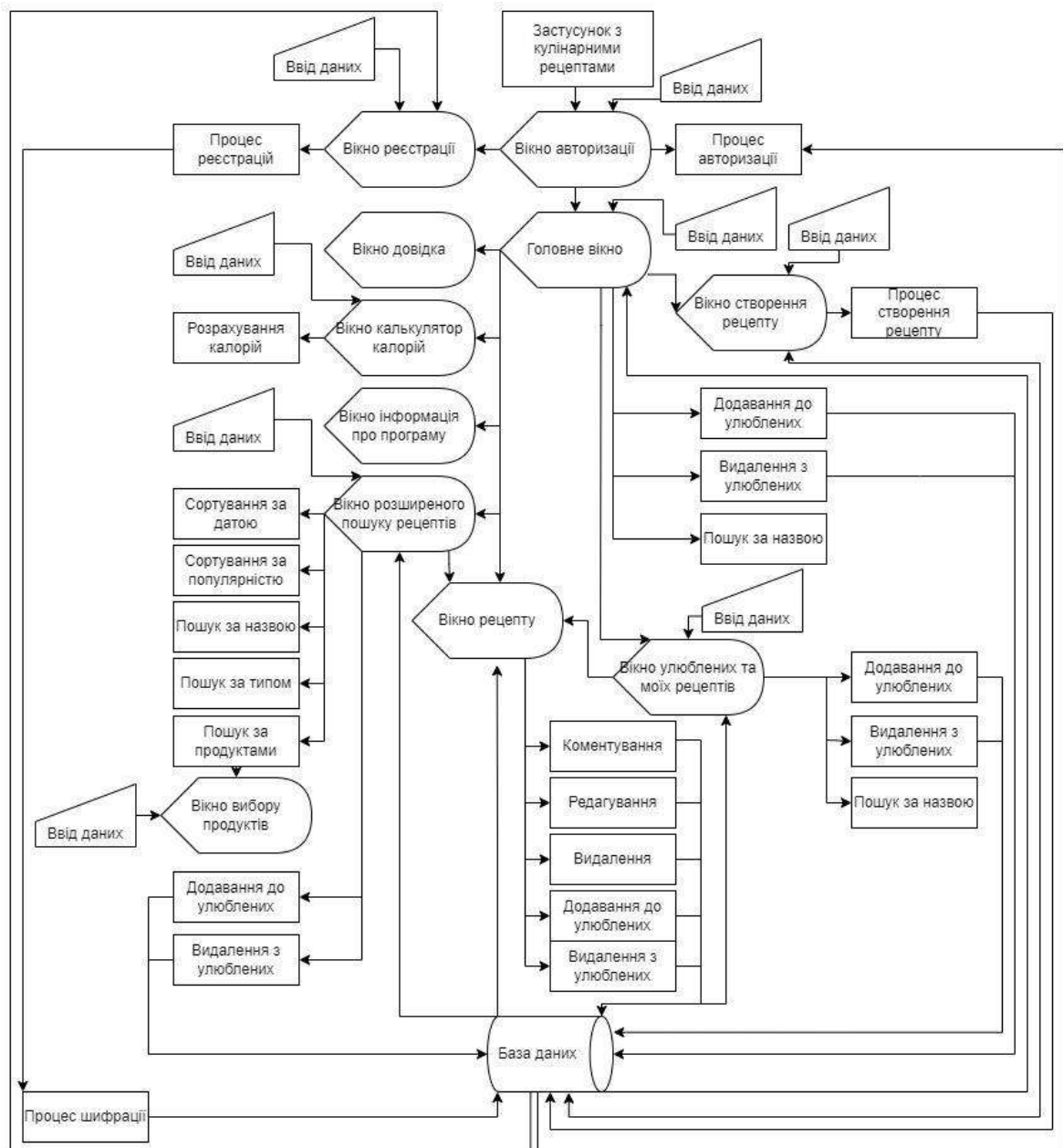


Рисунок 3.1 – Функціональна схема проекту

Як видно зі схеми під час розробки ПЗ «Ресріє», були розроблені наступні алгоритми:

- реєстрація;
- авторизація;
- розрахування калорійності;
- пошук за типом страви;
- сортування за популярністю;
- сортування за датою;
- пошук за продуктами;
- пошук за назвою;
- коментування рецепту;
- редагування рецепту;
- видалення рецепту;
- створення рецепту;
- додавання рецепту до улюблених;
- видалення рецепту з улюблених;
- шифрація.

3.1.1 Алгоритм реєстрації

Призначення: цей алгоритм створено для того, щоб реалізувати можливість створення нових акаунтів користувачами застосунку. В алгоритмі відбуваються перевірки коректності введених логіну та паролю.

Валідація логіну відбувається таким чином щоб не було використано вже існуючий логін чи некоректні символи. Валідація паролю відбувається таким чином щоб перевірити що не було використано некоректні символи та щоб співпадав пароль з його повторним вводом для уникнення можливої помилки користувачем під час вводу даних.

Через необхідність забезпечення безпечного збереження паролів користувачьких акаунтів, алгоритм тісно пов'язан з алгоритмом шифрації. На рисунку 3.2 зображено блок-схема алгоритму.

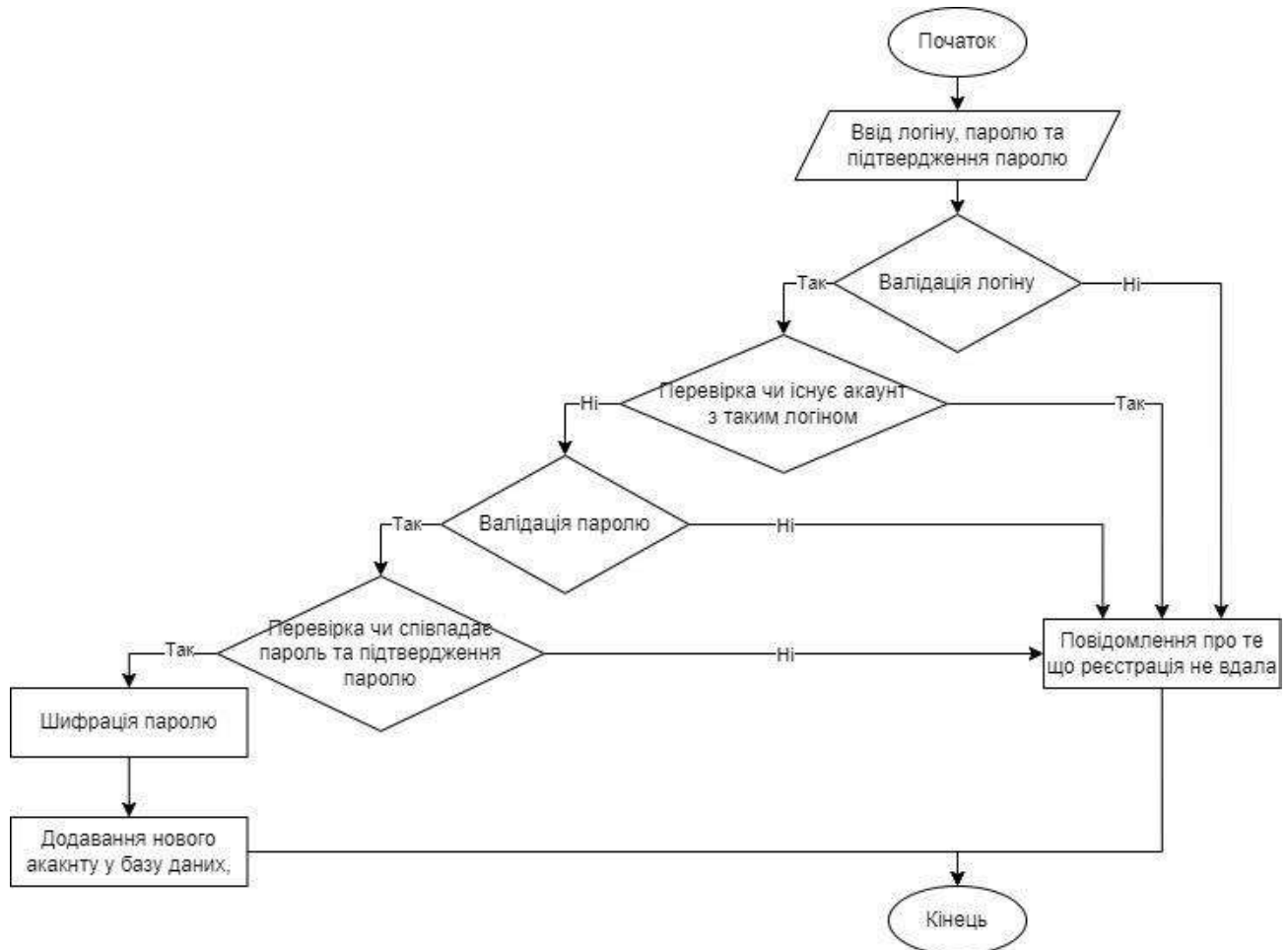


Рисунок 3.2 – Блок-схема для алгоритму реєстрації

3.1.2 Алгоритм авторизації

Призначення: цей алгоритм створено для того, щоб користувачі застосунку мали можливість увійти в систему під раніше створеним акаунтом, для подальшої роботи з усім існуючим функціоналом і отриманням більш якісного досвіду використання застосунку.

Через необхідність забезпечення безпечного використання паролів користувачьких акаунтів, алгоритм тісно пов'язан з алгоритмом шифрації. На рисунку 3.3 зображено блок-схема алгоритму.

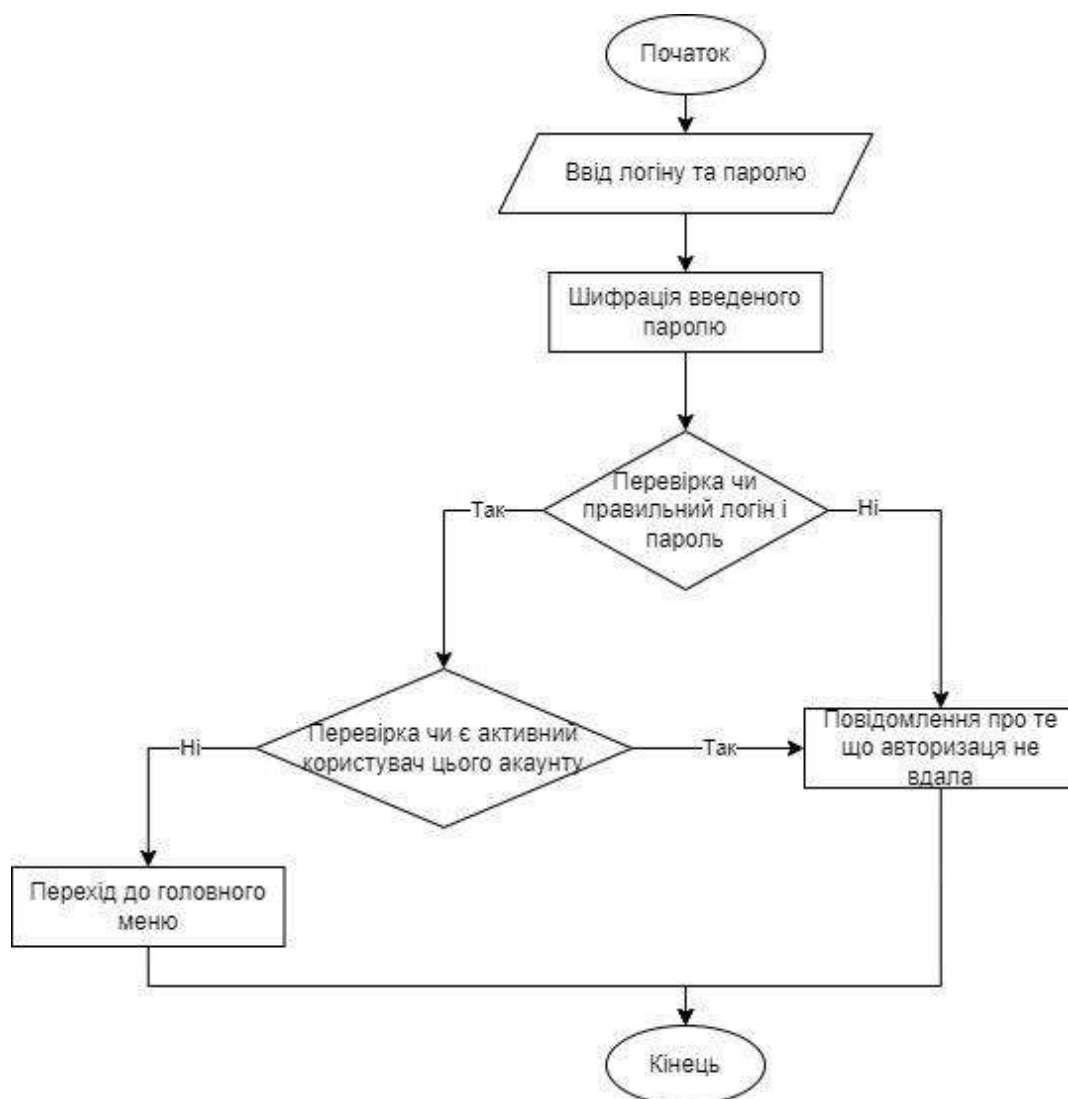


Рисунок 3.3 – Блок-схема для алгоритму авторизації

3.1.3 Алгоритм «Розрахування калорійності»

Призначення: цей алгоритм створено для того, щоб користувачі мали можливість підрахувати калорійність обраних до приготування страв за допомогою калькулятора калорій. На рисунку 3.4 зображено блок-схема алгоритму.

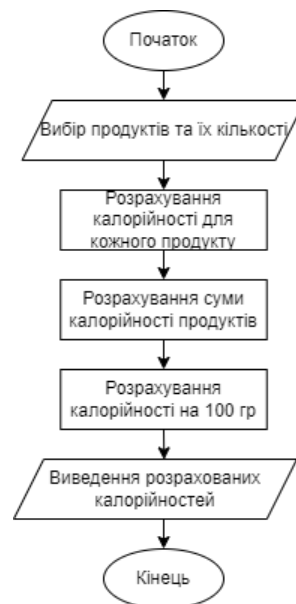


Рисунок 3.4 – Блок-схема для алгоритму «Розрахування калорійності»

3.1.4 Алгоритм сортування за популярністю

Призначення: цей алгоритм створено для того, щоб користувачі мали можливість побачити найпопулярніші рецепти. На рисунку 3.5 зображено блок-схема алгоритму.



Рисунок 3.5 – Блок-схема для сортування за популярністю

3.1.5 Алгоритм сортування за датою

Призначення: цей алгоритм створено для того, щоб користувачі мали можливість побачити нові та старі рецепти в залежності від обраного сортування. На рисунку 3.6 зображено блок-схема алгоритму.



Рисунок 3.6 – Блок-схема для сортування за датою

3.1.6 Алгоритм пошуку за продуктами

Призначення: цей алгоритм створено для того, щоб користувачі мали можливість відфільтрувати перелік рецептів за інгредієнтами, щоб мати можливість приготувати страви з наявними в них продуктами.

На рисунку 3.7 зображено блок-схема алгоритму.

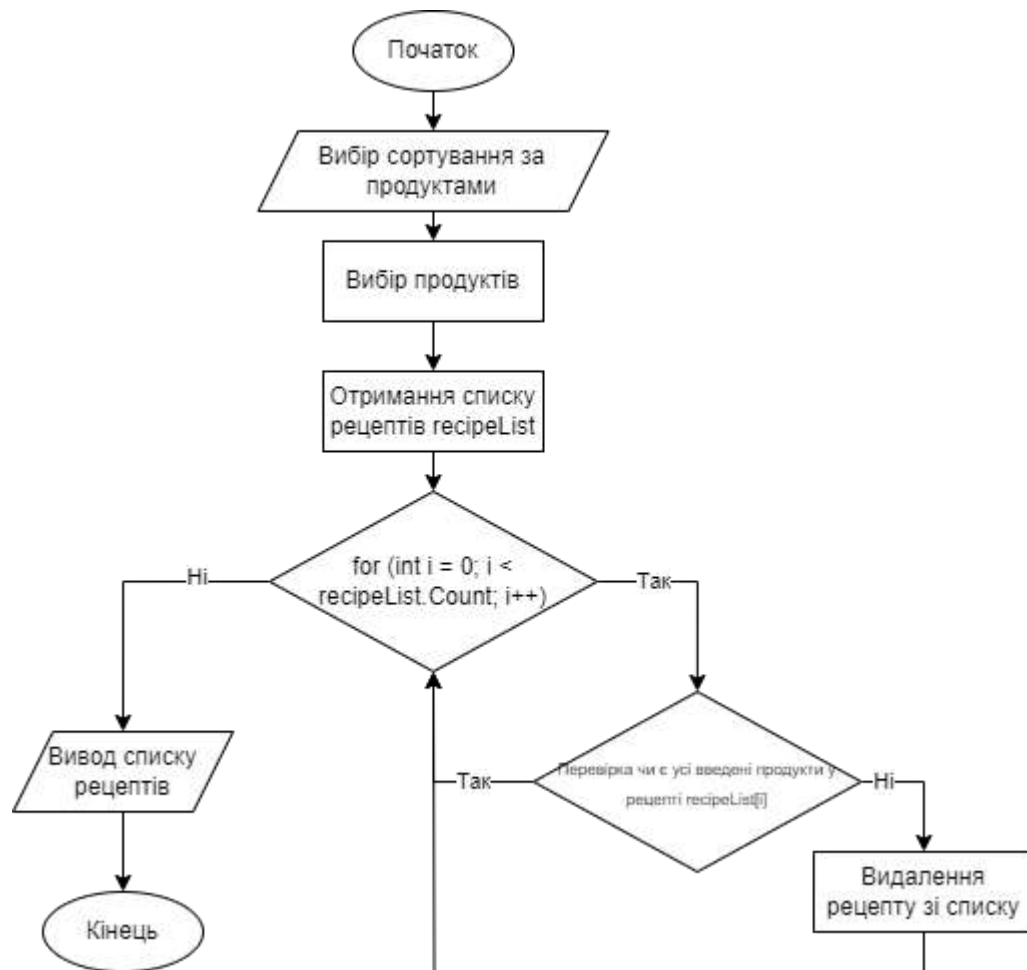


Рисунок 3.7 – Блок-схема для алгоритму пошуку за продуктами

3.1.7 Алгоритм пошуку за назвою

Призначення: цей алгоритм створено для того, щоб користувачі мали можливість знайти рецепт відповідно до його назви за ключовими словами.

На рисунку 3.8 зображено блок-схема алгоритму.



Рисунок 3.8 – Блок-схема для алгоритму «Авторизація»

3.1.8 Алгоритм «Коментування рецепту»

Призначення: цей алгоритм створено для того, щоб користувачі мали можливість коментувати рецепти, щоб мати можливість поділитися своїми емоціями та отримати зворотній зв'язок.

На рисунку 3.9 зображено блок-схема алгоритму.

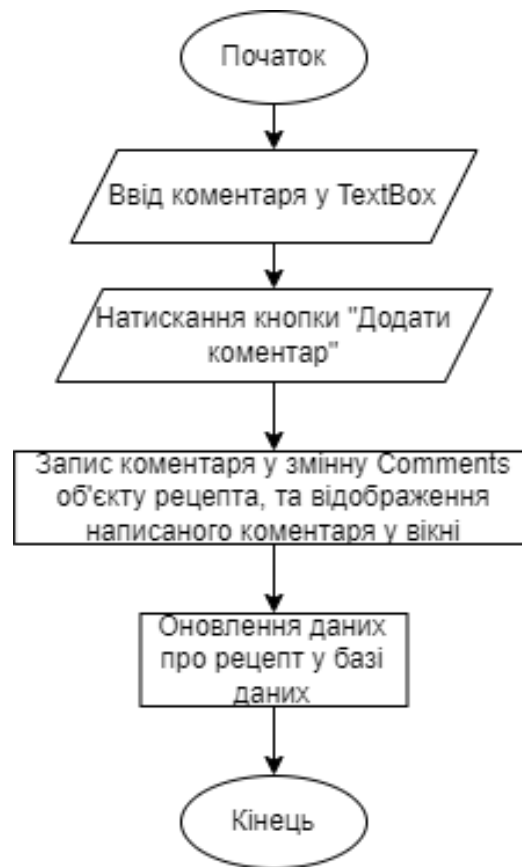


Рисунок 3.9 – Блок-схема для алгоритму «Коментування рецепту»

3.1.9 Алгоритм «Редагування рецепту»

Призначення: цей алгоритм створено для того, щоб користувачі мали можливість відредагувати створені ними рецепти, а саме такі параметри рецепту як назва, опис, тип страви, список продуктів, порядок дій, зображення, час приготування та кількість порцій.

На рисунку 3.10 зображено блок-схема алгоритму.



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

3.1.10 Алгоритм «Видалення рецепту»

Призначення: цей алгоритм створено для того, щоб користувачі мали можливість видаляти не потрібні власні рецепти, які раніше вони створювали. На рисунку 3.11 зображено блок-схема алгоритму.



Рисунок 3.11 – Блок-схема для алгоритму «Видалення рецепту»

3.1.11 Алгоритм «Створення рецепту»

Призначення: цей алгоритм створено для того, щоб користувачі мали створювати власні рецепти зі збереженням їх. На рисунку 3.12 зображено блок-схема алгоритму.



Рисунок 3.12 – Блок-схема для алгоритму «Створення рецепту»

3.1.12 Алгоритм «Додавання рецепту до улюблених»

Призначення: цей алгоритм створено для того, щоб користувачі мали можливість додавати рецепти до улюбленого, щоб мати до них швидкий доступ.

На рисунку 3.13 зображено блок-схема алгоритму



Рисунок 3.13 – Блок-схема для алгоритму «Створення рецепту»

3.1.13 Алгоритм «Видалення рецепту з улюблених»

Призначення: цей алгоритм створено для того, щоб користувачі мали можливість видаляти не потрібні рецепти з «улюблених». На рисунку 3.14 зображено блок-схема алгоритму



Рисунок 3.14 – Блок-схема для алгоритму «Створення рецепту»

3.1.14 Алгоритм «Сортування за популярністю»

Призначення: цей алгоритм створено для того, щоб користувачі мали можливість побачити популярні і непопулярні рецепти в залежності від обраного сортування. На рисунку 3.15 зображено блок-схема алгоритму.



Рисунок 3.15 – Блок-схема для алгоритму «Сортування за популярністю»

3.1.15 Алгоритм шифрації

Призначення: цей алгоритм створено для того, щоб шифрувати паролі які вводять користувачі за допомоги додавання до паролю солі та використання SHA-256 алгоритму. На рисунку 3.16 зображено блок-схема алгоритму.



Рисунок 3.16 – Блок-схема для алгоритму шифрації

Алгоритм SHA-256 відомий своєю надійністю у забезпеченні криптографічного захисту даних, а також здатністю зберегти цілісність і конфіденційність інформації. Важливими особливостями цього шифрування є [39]:

- SHA-256 – це криптографічна хеш-функція, що приймає вхідні дані і створює фіксований за розміром хеш-вихід. Цей вихід унікальний для кожного введення, ускладнюючи зворотне проектування вихідного повідомлення;
- хеш-функція SHA-256 генерує хеш-значення довжиною 256 біт (32 байти), що дозволяє ефективно порівнювати дані та перевіряти цілісність;
- SHA-256 демонструє високу стійкість до зіткнень, що гарантує, що навіть малі зміни вхідних даних призведуть до значно різних хеш-значень;
- ефект лавини викликає значні зміни у хеш-значенні при найменших змінах вхідних даних, ускладнюючи зворотне відтворення;

- SHA-256 – це одностороння функція, що ускладнює обчислення зворотного процесу хешування і забезпечує безпеку для зберігання паролів та цифрових підписів;
- алгоритм завжди створює однакове хеш-значення для однакових вхідних даних, що дозволяє проводити послідовні процеси перевірки та автентифікації;
- SHA-256 забезпечує баланс між безпекою та ефективністю, що робить його підходящим для різних програм;
- його широке застосування охоплює цифрові підписи, зберігання паролів, технологію блокчейн та перевірку цілісності даних;
- SHA-256 є стандартом, схваленим NIST (Національний інститут стандартів і технологій (National Institute of Standards and Technology) в США), що гарантує його високий рівень безпеки та відповідність стандартам;
- алгоритм стійкий до атак попереднього зображення, ускладнюючи визначення вхідних даних, що відповідають певному хеш-значенню.

3.2 Вибір шаблону проектування

Архітектура розробки програмного забезпечення визначає структуру і взаємодію компонентів програми, розподіляючи функціональні обов'язки та організовуючи внутрішні процеси. Цей підхід допомагає забезпечити ефективну роботу програми, розуміння її логіки та спрощення розробки та підтримки коду [40].

Для побудови архітектури ПЗ було вирішено використовувати шаблон проєктування клієнт-сервер, який зображено на рисунку 3.17

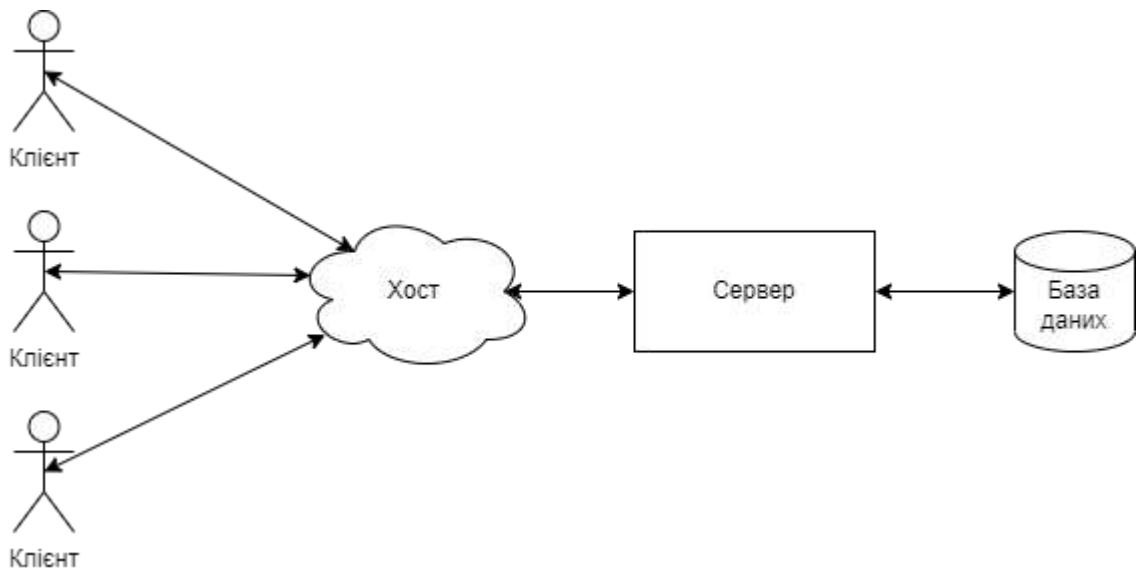


Рисунок 3.17 – Клієнт-серверна архітектура

У рамках підходу «клієнт – сервер», різні компоненти мають визначену роль та взаємодію, що визначається наступними принципами [41]:

- «клієнти» надсилають запити до «сервера» та очікують на відповідь;
- «сервер» приймає запит від «клієнта» та надсилає відповідь назад.

Архітектура програмного забезпечення, що використовує клієнт-серверний підхід, є дуже популярною у тих випадках, коли необхідно контролювати доступ користувачів до ресурсів. Цей підхід сприяє масштабуванню додатка та робить його інтерфейс більш доступним і зрозумілим [42].

За допомогою клієнт-серверної архітектури можна створювати різноманітні додатки та сервіси, починаючи від веб-сайтів та мобільних додатків і закінчуючи корпоративними системами. Кожен клієнт може бути реалізований на різних платформах та пристроях, що дозволяє забезпечити доступ до системи з будь-якого місця та пристрою. Крім того, ця архітектура дозволяє розділити завдання між клієнтами та серверами відповідно до їх функціональних можливостей, що сприяє оптимізації ресурсів та покращує загальну продуктивність системи.

Клієнт-серверна архітектура має кілька переваг, серед яких:

- розподілені обчислення. Застосування розбивається на дві основні компоненти – клієнтську та серверну частину. Це дозволяє розподілити обчислювальні завдання між клієнтом та сервером, що покращує ефективність та продуктивність системи;
- масштабованість. Клієнт-серверна архітектура дозволяє легко масштабувати систему, додавати нові клієнти або сервери для забезпечення зростання навантаження. Це робить архітектуру гнучкою та готовою до збільшення обсягу роботи;
- централізоване керування. Сервер виконує централізоване керування та обробку даних, що дозволяє забезпечити єдиний джерело правдивої інформації та забезпечує консистентність даних у системі;
- забезпечення безпеки. Клієнт-серверна архітектура дозволяє застосовувати різні механізми безпеки, такі як автентифікація та авторизація, шифрування даних, контроль доступу до ресурсів та інші. Це дозволяє забезпечити захист конфіденційної та критичної інформації;
- розширюваність. Клієнт-серверна архітектура дозволяє легко розширювати функціональність застосунку шляхом додавання нових модулів або розширення існуючих модулів;
- зручне управління та технічна підтримка. Розділення клієнта та сервера спрощує управління системою та надання технічної підтримки. Клієнти можуть бути незалежними один від одного, а сервер може забезпечувати централізоване управління та підтримку.

Недоліки такого підходу включають наступне [41]:

- сервер може стати обмеженням щодо продуктивності та єдиним пунктом відмови;
- зміна рішень щодо розташування функціональності (на клієнті або на сервері) після створення системи зазвичай ускладнюється та є витратною.

3.3 Опис модулів розроблюваної системи

Для створених модулів було написані такі класи для клієнтської частини:

- About – реалізує вікно з інформацією про розробників ПЗ;
- Authorization – реалізує вікно з авторизацією;
- CalorieCalculator – реалізує вікно з калькулятором калорій;
- ProductBoxItem – реалізує збереження даних для ListView на вікні з калькулятором калорій;
- CreateRecipe – реалізує вікно для створення рецепту;
- Favorite – реалізує вікно для відображення власних рецептів та донаних до улюблених;
- ImageByte – реалізує конвертацію зображень у масив байтів та навпаки;
- Information – реалізує вікно з інформацією про основні можливості ПЗ;
- MainWindow – реалізує головне вікно ПЗ;
- Products – реалізує додаткове вікно з продуктами для розширеного пошуку;
- Recipe – реалізує вікно з рецептом та детальною інформацією про нього;
- RecipeMessageBox – реалізує вікно для сповіщення про помилку;
- Registration – реалізує вікно для реєстрації;
- Search – реалізує вікно для розширеного пошуку;
- ClientException – реалізує обробку виняткових ситуацій.

Для візуалізації класової ієрархії на клієнтській частині було створено діаграму класів, яку представлено на рисунку 3.18.

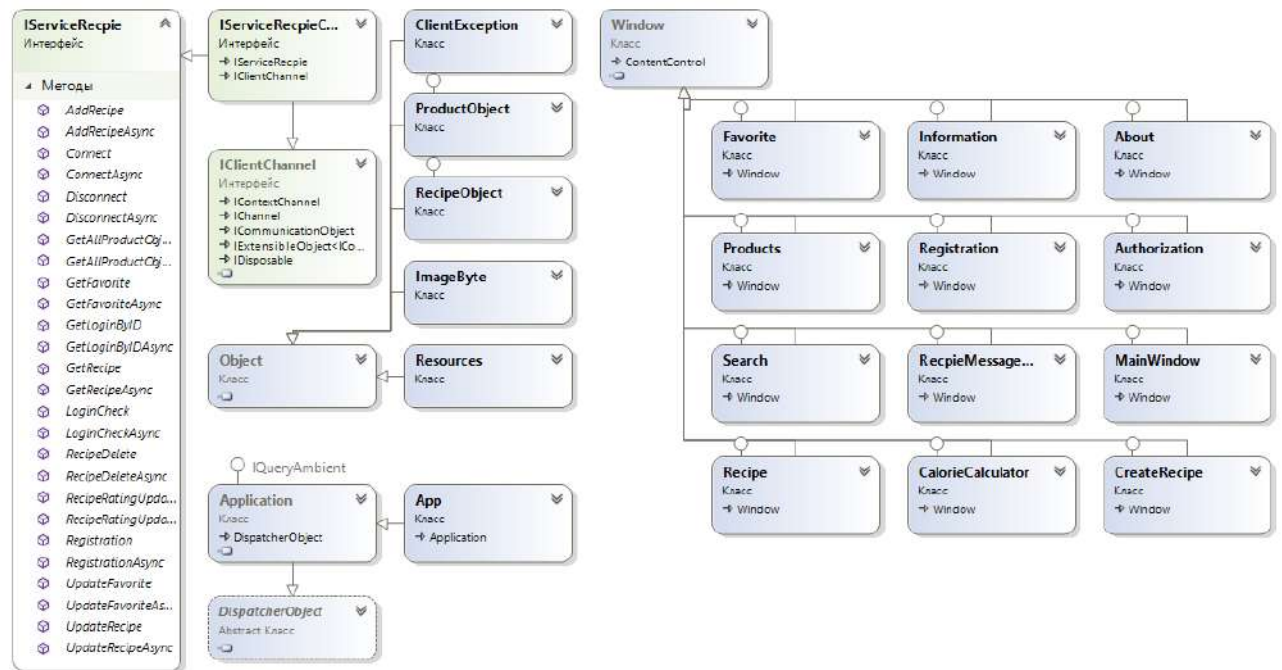


Рисунок 3.18 – Діаграма класів клієнтської частини

Для створених модулів було написані такі класи для серверної частини:

- ProductObject – реалізує об’єкт продукту для передачі даних для вікна «Калькулятор калорій»;
- RecipeObject – реалізує об’єкт рецепт для передачі даних про рецепт до клієнтської частини;
- RecipeDataBase – реалізує роботу з БД;
- ServerUser – реалізує об’єкт користувач для зберігання інформації про активного користувача на сервері;
- ServiceRecipe – реалізує функції інтерфейсу IServiceRecipe.

Для створених модулів було написані такі інтерфейси для серверної частини:

- IServiceRecipe – реалізує інтерфейс для функцій серверу;
- IServerRecipeCallback – реалізує інтерфейс для зворотного зв’язку між сервером та клієнтом;
- RecipeHashing – реалізує шифрацію паролю;
- SHA256Hash – реалізує алгоритм хешування SHA 256.

Для візуалізації класової ієрархії на серверній частині було створено діаграму класів, яку представлено на рисунку 3.19.

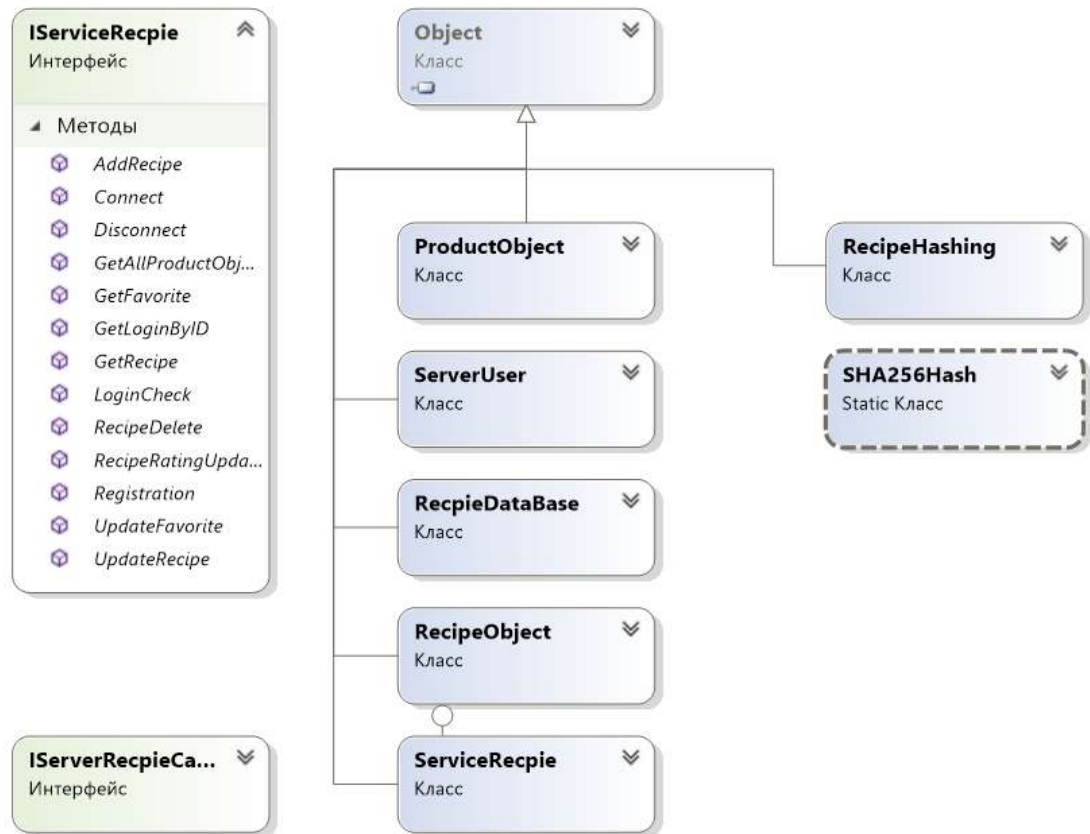


Рисунок 3.19 – Діаграма класів серверної частини

Для створених модулів було написані такі класи для хост-частини:

- Program – реалізує роботу хост-частини.

Для візуалізації класової ієрархії на хост-частині було створено діаграму класів, яку представлено на рисунку 3.20.

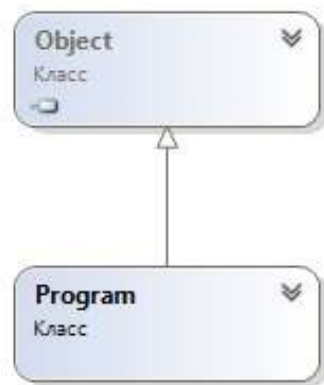


Рисунок 3.20 – Діаграма класів хост-частини

3.3.1 Клас About

Ціль класу полягає у відображенні інформації про розробника ПЗ за допомогою вікна.

Дані та методи класу наведені у таблиці 3.1.

Таблиця 3.1 – Опис полів та методів класу About

Поля та методи класу	Опис
1	2
<i>public:</i>	
public About()	Конструктор, який ініціалізує вікно

3.3.2 Клас Authorization

Ціль класу полягає у відображенні та реалізації процесу авторизації. Дані та методи класу наведені у таблиці 3.2.

Таблиця 3.2 – Опис полів та методів класу Authorization

Поля та методи класу	Опис
1	2
public:	
public Authorization()	Конструктор, який ініціалізує вікно
private:	
ServiceRecpieClient client	Змінна для роботи з сервером
MainWindow mainWindow	Змінна для головного вікна
Registration registration	Зміні для вікна з реєстрацією
private void SingIn_Click(object sender, RoutedEventArgs e)	Реалізація події при натискання кнопки «Увійти»
private void SignUp_Click(object sender, RoutedEventArgs e)	Реалізація події при натискання кнопки «Реєстрація»
private void SingIn_NoLogin_Click(object sender, RoutedEventArgs e)	Реалізація події при натискання кнопки «Увійти без акаунту»
private void Window_KeyDown(object sender, System.Windows.Input.KeyEventArgs e)	Обробка гарячої клавіші «Enter»

3.3.3 Клас CalorieCalculator

Ціль класу полягає у відображенні та реалізації процесу підрахунку калорій страви. Дані та методи класу наведені у таблиці 3.3.

Таблиця 3.3 – Опис полів та методів класу CalorieCalculator

Поля та методи класу	Опис
1	2
public:	
public CalorieCalculator(MainWindow mainWindow)	Конструктор, який ініціалізує вікно

Продовження таблиці 3.3

1	2
<i>private:</i>	
ServiceRecpieClient client	Змінна для роботи з сервером
MainWindow mainWindow	Змінна для головного вікна
List<ProductObject> products	Список продуктів, які можна додати у страву
ProductObject selectedProduct	Змінна в якій зберігається інформація про обраний продукт
void ProductNameBoxLoad()	Завантаження продуктів в ComboBox
private void ProductNameBox_SelectionChanged(object sender, SelectionChangedEventArgs e)	Зміна обраного продукта в ComboBox
void UpdateCaloricContentBox()	Алгоритм підрахування калорій
private void AddButton_Click(object sender, RoutedEventArgs e)	Реалізація кнопки «Додати»
private void DelButton_Click(object sender, RoutedEventArgs e)	Реалізація кнопки «Видалити»
private void BackButton_Click(object sender, RoutedEventArgs e)	Реалізація кнопки «Назад»
private void Window_Closing(object sender, System.ComponentModel.CancelEventArgs e)	Закриття вікна

3.3.4 Клас ProductBoxItem

Ціль класу полягає у збереженні даних для ListView на вікні з калькулятором калорій;. Дані та методи класу наведені у таблиці 3.4.

Таблиця 3.4 – Опис полів та методів класу ProductBoxItem

Поля та методи класу	Опис
1	2
<i>public:</i>	
public string Name	Змінна для збереження назви
public string Amount	Змінна для збереження кількості продукту в грамах
public double Calories	Змінна для збереження калорійності продукту на введену кількість

3.3.5 Клас CreateRecipe

Ціль класу полягає у відображенні та реалізації створення рецепту. Дані та методи класу наведені у таблиці 3.5.

Таблиця 3.5 – Опис полів та методів класу CreateRecipe

Поля та методи класу	Опис
1	2
<i>public:</i>	
public CreateRecipe(int ID, MainWindow mainWindow)	Конструктор, який ініціалізує вікно при відкритті з головного вікна
public CreateRecipe(int ID, Recipe recipeWin, RecipeObject recipe)	Конструктор, який ініціалізує вікно при відкритті з вікна з детальною інформацією про рецепт
<i>private:</i>	
ServiceRecpieClient client	Змінна для роботи з сервером
MainWindow mainWindow	Змінна для головного вікна

Продовження таблиці 3.5

1	2
Recipe recipeWin	Змінна для вікна з детальною інформацією про рецепт
int ID;	Змінна, яка зберігає ID користувача
bool Redact = false;	Прапорець, який показує чи увімкнено режим редагування
RecipeObject recipeObject;	Змінна, яка зберігає дані про рецепт
byte[] Image;	Змінна для збереження зображення
private void Window_Closing(object sender, System.ComponentModel.CancelEventArgs e)	Закриття вікна
private void LoadImageButton_Click(object sender, RoutedEventArgs e)	Завантаження зображення
private void CreateButton_Click(object sender, RoutedEventArgs e)	Створення та збереження рецепту
private void BackButton_Click(object sender, RoutedEventArgs e)	Реалізація кнопки «Назад»

3.3.6 Клас Favorite

Ціль класу полягає у реалізації вікна «Улюблені та мої рецепти».

Дані та методи класу наведені у таблиці 3.6.

Таблиця 3.6 – Опис полів та методів класу Favorite

Поля та методи класу	Опис
1	2
<i>public:</i>	
public Favorite(int ID, MainWindow mainWindow)	Конструктор, який ініціалізує вікно
public void Window_Loaded()	Оновлення інформації про вікно при зміні інформації у іншому вікні
<i>private:</i>	
ServiceRecpieClient client	Змінна для роботи з сервером
MainWindow mainWindow	Змінна для головного вікна
Recipe recipeWin	Змінна для вікна з детальною інформацією про рецепт
int ID	Змінна, яка зберігає ID користувача
RecipeObject recipe1	Змінна, яка зберігає дані про перший рецепт
RecipeObject recipe2;	Змінна, яка зберігає дані про другий рецепт
RecipeObject recipe3	Змінна, яка зберігає дані про третій рецепт
int numOfPageNow	Змінна, яка зберігає номер поточної сторінки
int countOfPage	Змінна, яка зберігає загальну кількість сторінок
List<RecipeObject> recipeList	Список з всіма рецептами
bool load	Прапорець, який позначає активність процесу відображення рецептів

Продовження таблиці 3.6

1	2
private void Window_Loaded(object sender, RoutedEventArgs e)	Оновлення інформації про вікно при завантаженні вікна
private void Window_Closing(object sender, System.ComponentModel.CancelEventArgs e)	Закриття вікна
void UpdateRecipeList()	Оновлення списку рецептів
void SearchFilter()	Відфільтрування списку рецептів
void LoadRecipe()	Відображення рецептів у вікні
void UpdatePageBox()	Оновлення сторінок
private void SearchBox_TextChanged(object sender, TextChangedEventArgs e)	Пошук рецептів за назвою
private void CheckBox1_Checked(object sender, RoutedEventArgs e)	Додавання першого рецепту до улюблених
private void CheckBox2_Checked(object sender, RoutedEventArgs e)	Додавання другого рецепту до улюблених
private void CheckBox3_Checked(object sender, RoutedEventArgs e)	Додавання третього рецепту до улюблених
private void CheckBox1_Unchecked(object sender, RoutedEventArgs e)	Видалення першого рецепту з улюблених

Продовження таблиці 3.6

1	2
private void CheckBox2_Unchecked(object sender, RoutedEventArgs e)	Видалення другого рецепту з улюблених
private void CheckBox3_Unchecked(object sender, RoutedEventArgs e)	Видалення третього рецепту з улюблених
private void FavoriteAndMy Box_DropDownClosed(object sender, EventArgs e)	Зміна режиму відображення власних рецептів чи улюблених
private void NextPage Button_Click(object sender, RoutedEventArgs e)	Перехід на наступну сторінку
private void Previous PageButton_Click(object sender, RoutedEventArgs e)	Перехід на попередню сторінку
private void BackButton_Click(object sender, RoutedEventArgs e)	Реалізація кнопки «Назад»
private void NameBox1_MouseDoubleClick(object sender, System.Windows.Input.MouseButtonEv entArgs e)	Відкриття детальної інформації про перший рецепт
private void NameBox2_MouseDouble Click(object sender, System.Windows. Input.MouseButtonEventArgs e)	Відкриття детальної інформації про другий рецепт

Продовження таблиці 3.6

1	2
private void NameBox3_MouseDoubleClick(object sender, System.Windows.Input.MouseButtonEventArgs e)	Відкриття детальної інформації про третій рецепт

3.3.7 Клас ImageByte

Ціль класу полягає у конвертації зображень у масив байтів та навпаки. Дані та методи класу наведені у таблиці 3.7.

Таблиця 3.7 – Опис полів та методів класу ImageByte

Поля та методи класу	Опис
1	2
public:	
public static byte[] CopyImageToByteArray(Image theImage)	Конвертація зображення в масив байтів
public static Bitmap GetImageFromByteArray(byte[] byteArray)	Конвертація масиву байтів в Bitmap
public static BitmapImage Convert(Bitmap src)	Конвертація Bitmap в BitmapImage
private:	
private static readonly ImageConverter _imageConverter	Конвертор для масиву байтів

3.3.8 Клас Information

Ціль класу полягає у відображенні інформації про основні можливості ПЗ. Дані та методи класу наведені у таблиці 3.8.

Таблиця 3.8 – Опис полів та методів класу Information

Поля та методи класу	Опис
1	2
public:	
public Information()	Конструктор, який ініціалізує вікно

3.3.9 Клас MainWindow

Ціль класу полягає у відображенні та реалізації головного вікна ПЗ. Дані та методи класу наведені у таблиці 3.9.

Таблиця 3.9 – Опис полів та методів класу MainWindow

Поля та методи класу	Опис
1	2
public:	
public MainWindow(int ID, Authorization authorizationWin)	Конструктор, який ініціалізує вікно
public void Window_Loaded()	Оновлення інформації про вікно при зміні інформації у іншому вікні
private:	
Authorization authorizationWin	Змінна вікна з авторизацією
CreateRecipe createRecipeWin	Змінна вікна з створенням рецепту
Search searchWin	Змінна вікна з розширеним пошуком

Продовження таблиці 3.9

1	2
Recipe recipeWin	Змінна вікна з детальною інформацією про рецепт
CalorieCalculator calorieCalculatorWin	Змінна вікна з калькулятором калорій страви
Favorite favoriteWin	Змінна вікна з улюбленими та моїми рецептами
About aboutWin	Змінна вікна з інформацією про розробниками
Information informationWin	Змінна вікна з інформацією про ПЗ
ServiceRecpieClient client	Змінна для роботи з сервером
Recipe recipeWin	Змінна для вікна з детальною інформацією про рецепт
int ID	Змінна, яка зберігає ID користувача
RecipeObject recipe1	Змінна, яка зберігає дані про перший рецепт
RecipeObject recipe2;	Змінна, яка зберігає дані про другий рецепт
RecipeObject recipe3	Змінна, яка зберігає дані про третій рецепт
int numOfPageNow	Змінна, яка зберігає номер поточної сторінки
int countOfPage	Змінна, яка зберігає загальну кількість сторінок
List<RecipeObject> recipeList	Список з всіма рецептами
bool load	Прапорець, який позначає активність процесу відображення рецептів

Продовження таблиці 3.9

1	2
private void Window_Loaded(object sender, RoutedEventArgs e)	Завантаження вікна
private void Window_Closing(object sender, System.ComponentModel.CancelEventArgs e)	Закриття вікна
void UpdateRecipeList()	Оновлення списку рецептів
void SearchFilter()	Відфільтрування списку рецептів
void LoadRecipe()	Відображення рецептів у вікні
void UpdatePageBox()	Оновлення сторінок
private void SearchBox_TextChanged(object sender, TextChangedEventArgs e)	Пошук рецептів за назвою
private void CheckBox1_Checked(object sender, RoutedEventArgs e)	Додавання першого рецепту до улюблених
private void CheckBox2_Checked(object sender, RoutedEventArgs e)	Додавання другого рецепту до улюблених
private void CheckBox3_Checked(object sender, RoutedEventArgs e)	Додавання третього рецепту до улюблених
private void CheckBox1_Unchecked(object sender, RoutedEventArgs e)	Видалення першого рецепту з улюблених

Продовження таблиці 3.9

1	2
private void CheckBox2_Unchecked(object sender, RoutedEventArgs e)	Видалення другого рецепту з улюблених
private void CheckBox3_Unchecked(object sender, RoutedEventArgs e)	Видалення третього рецепту з улюблених
private void NameBox1_MouseDoubleClick(object sender, System.Windows.Input.MouseButtonEv entArgs e)	Відкриття детальної інформації про перший рецепт
private void NameBox2_MouseDoubleClick(object sender, System.Windows.Input.MouseButtonEv entArgs e)	Відкриття детальної інформації про другий рецепт
private void NameBox3_MouseDoubleClick (object sender, System.Windows .Input.MouseButtonEventArgs e)	Відкриття детальної інформації про третій рецепт
private void AddRecipeButton_Click(object sender, RoutedEventArgs e)	Відкриття вікна з створенням рецепту
private void NextPageButton_Click(object sender, RoutedEventArgs e)	Перехід на наступну сторінку

Продовження таблиці 3.9

1	2
private void PreviousPage_Button_Click(object sender, RoutedEventArgs e)	Перехід на попередню сторінку
private void AdvancedSearchButton_Click(object sender, RoutedEventArgs e)	Відкриття вікна з розширеним пошуком
private void ExitButton_Click(object sender, RoutedEventArgs e)	Кнопка назад
private void CalculatorButton_Click(object sender, RoutedEventArgs e)	Відкриття вікна з калькулятором калорій страви
private void Favorite_Click(object sender, RoutedEventArgs e)	Відкриття вікна з улюбленими та моїми рецептами
private void AboutButton_Click(object sender, RoutedEventArgs e)	Відкриття вікна з інформацією розробників
private void InformationButton_Click(object sender, RoutedEventArgs e)	Відкриття вікна з довідкою про ПЗ

3.3.10 Клас Products

Ціль класу полягає у реалізації додаткового вікна з продуктами для розширеного пошуку.

Дані та методи класу наведені у таблиці 3.10.

Таблиця 3.10 – Опис полів та методів класу Products

Поля та методи класу	Опис
1	2
public:	
public Products(Search searchWin)	Конструктор, який ініціалізує вікно
private:	
List<string> productList	Список обраних продуктів
private void Window_Closing(object sender, System.ComponentModel.CancelEventArgs e)	Закриття вікна

3.3.11 Клас Recipe

Ціль класу полягає у відображенні і реалізації вікна з рецептом та детальною інформацією про нього. Дані та методи класу наведені у таблиці 3.11.

Таблиця 3.11 – Опис полів та методів класу Recipe

Поля та методи класу	Опис
1	2
public:	
public RecipeObject recipe	Змінна яка зберігає інформацію про рецепт
public Recipe(RecipeObject recipe, int ID, MainWindow mainWindow)	Конструктор, який ініціалізує вікно при відкритті з головного вікна
public Recipe(RecipeObject recipe, int ID, Search searchWin)	Конструктор, який ініціалізує вікно при відкритті з вікна з розширеним пошуком

Продовження таблиці 3.11

1	2
public Recipe(RecipeObject recipe, int ID, Favorite favoriteWin)	Конструктор, який ініціалізує вікно при відкритті з вікна з улюбленими і моїми рецептами
public void Window_Loaded()	Оновлення інформації про вікно при зміні інформації у іншому вікні
private:	
MainWindow mainWindow	Змінна головного вікна
Search searchWin	Змінна з розширеним пошуком
Favorite favoriteWin	Змінна з моїми та улюбленими рецептами
CreateRecipe createRecipe	Змінна з створенням рецепту
int ID	Змінна, яка зберігає ID користувача
bool load	Прапорець, який позначає активність процесу відображення рецептів
ServiceRecpieClient client	Змінна для роботи з сервером
private void Window_Loaded(object sender, RoutedEventArgs e)	Відкриття вікна
private void Window_Closing(object sender, System.ComponentModel.CancelEventArgs e)	Закриття вікна
private void BackButton_Click(object sender, RoutedEventArgs e)	Повернення до минулого вікна
private void RedactButton_Click(object sender, RoutedEventArgs e)	Редагування рецепту
private void DelButton_Click(object sender, RoutedEventArgs e)	Видалення рецепту

Продовження таблиці 3.11

1	2
private void AddComment Button_Click (object sender, RoutedEventArgs e)	Додавання коментарію
private void CheckBox_Checked(object sender, RoutedEventArgs e)	Додавання рецепту до улюблених
private void CheckBox_Unchecked(object sender, RoutedEventArgs e)	Видалення рецепту з улюблених

3.3.12 Клас RespieMessageBox

Ціль класу полягає у відображенні вікна для сповіщення про помилку. Дані та методи класу наведені у таблиці 3.12.

Таблиця 3.12 – Опис полів та методів класу RespieMessageBox

Поля та методи класу	Опис
1	2
public:	
public RespieMessageBox(string message, string title)	Конструктор, який ініціалізує вікно
private:	
private void OKButton_Click(object sender, RoutedEventArgs e)	Кнопка ОК

3.3.13 Клас Registration

Ціль класу полягає у відображенні та реалізації вікна для реєстрації. Дані та методи класу наведені у таблиці 3.13.

Таблиця 3.13 – Опис полів та методів класу Registration

Поля та методи класу	Опис
1	2
public:	
public Registration(Authorization authorizationWin)	Конструктор, який ініціалізує вікно
private:	
ServiceRecpieClient client	Змінна для роботи з сервером
Authorization authorizationWin	Змінна з авторизацією
bool LoginValidation(string Login)	Перевірка коректності логіну
bool Password Validation (string Password)	Перевірка коректності паролю
private void Window_Closing(object sender, System. Component Model.CancelEventArgs e)	Закриття вікна
private void Registration Button_Click (object sender, RoutedEventArgs e)	Алгоритм реєстрації
private void Back_Click(object sender, RoutedEventArgs e)	Кнопка назад

3.3.14 Клас Search

Ціль класу полягає у відображенні та реалізації вікна для розширеного пошуку. Дані та методи класу наведені у таблиці 3.14.

Таблиця 3.14 – Опис полів та методів класу Search

Поля та методи класу	Опис
1	2
public:	
public void Window_Loaded()	Конструктор, який ініціалізує вікно

Продовження таблиці 3.14

1	2
public Search(int ID, MainWindow mainWindow)	Оновлення інформації про вікно при зміні інформації у іншому вікні
<i>private:</i>	
MainWindow mainWindow;	Змінна з головним вікном
Recipe recipeWin;	Змінна з детальною інформацією про рецепт
Products productsWin;	Змінна з додатковим вікном з продуктами
ServiceRecpieClient client	Змінна для роботи з сервером
List<string> productListFilter	Змінна з списком фільтрованих рецептів
int ID	Змінна, яка зберігає ID користувача
RecipeObject recipe1	Змінна, яка зберігає дані про перший рецепт
RecipeObject recipe2;	Змінна, яка зберігає дані про другий рецепт
RecipeObject recipe3	Змінна, яка зберігає дані про третій рецепт
int numOfPageNow	Змінна, яка зберігає номер поточної сторінки
int countOfPage	Змінна, яка зберігає загальну кількість сторінок
List<RecipeObject> recipeList	Список з всіма рецептами
bool load	Прапорець, який позначає активність процесу відображення рецептів
private void Window_Loaded(object sender, RoutedEventArgs e)	Завантаження вікна

Продовження таблиці 3.14

1	2
private void Window_Closing(object sender, System.ComponentModel.CancelEventArgs e)	Закриття вікна
void UpdateRecipeList()	Оновлення списку рецептів
void SearchFilter()	Відфільтрування списку рецептів
void LoadRecipe()	Відображення рецептів у вікні
void UpdatePageBox()	Оновлення сторінок
private void SearchBox_TextChanged(object sender, TextChangedEventArgs e)	Пошук рецептів за назвою
private void CheckBox1_Checked(object sender, RoutedEventArgs e)	Додавання першого рецепту до улюблених
private void CheckBox2_Checked(object sender, RoutedEventArgs e)	Додавання другого рецепту до улюблених
private void CheckBox3_Checked(object sender, RoutedEventArgs e)	Додавання третього рецепту до улюблених
private void CheckBox1_Unchecked(object sender, RoutedEventArgs e)	Видалення першого рецепту з улюблених
private void CheckBox2_Unchecked(object sender, RoutedEventArgs e)	Видалення другого рецепту з улюблених

Продовження таблиці 3.14

1	2
private void CheckBox3_Unchecked(object sender, RoutedEventArgs e)	Видалення третього рецепту з улюблених
private void NameBox1_MouseDoubleClick(object sender, System.Windows.Input.MouseButtonEv entArgs e)	Відкриття детальної інформації про перший рецепт
private void NameBox2_MouseDoubleClick(object sender, System.Windows.Input.MouseButtonEv entArgs e)	Відкриття детальної інформації про другий рецепт
private void NameBox3_MouseDoubleClick(object sender, System.Windows.Input.MouseButtonEv entArgs e)	Відкриття детальної інформації про третій рецепт
private void TypeBox_DropDownClosed(object sender, EventArgs e)	Фільтрування за типом рецепту
private void RatingBox_DropDownClosed(object sender, EventArgs e)	Фільтрування за рейтингом рецепту
private void DataBox_DropDownClosed(object sender, EventArgs e)	Фільтрування за датою рецепту

Продовження таблиці 3.14

1	2
public void SetProductListFilter(List<string> productList)	Встановлення фільтру за продуктами
private void NextPageButton_Click(object sender, RoutedEventArgs e)	Перехід на наступну сторінку
private void PreviousPageButton_Click(object sender, RoutedEventArgs e)	Перехід на минулу сторінку
private void BackButton_Click(object sender, RoutedEventArgs e)	Кнопка назад
private void ProductButton_Click(object sender, RoutedEventArgs e)	Відкриття додаткового вікна з продуктами

3.3.15 Клас ClientException

Ціль класу полягає у реалізації обробки виняткових ситуацій, які можуть виникати при роботі програмного рішення. Дані та методи класу наведені у таблиці 3.15.

Таблиця 3.15 – Опис полів та методів класу ClientException

Поля та методи класу	Опис
1	2
public:	
public ClientException(int exceptionCode)	Конструктор, який ініціалізує обробку помилки

3.3.16 Клас ProductObject

Ціль класу полягає у збереженні інформації про продукт. Дані та методи класу наведені у таблиці 3.16.

Таблиця 3.16 – Опис полів та методів класу ProductObject

Поля та методи класу	Опис
1	2
<i>public:</i>	
public int ID	Змінна для збереження ID продукту
public string Name	Змінна для збереження назви продукту
public double Calories	Змінна для збереження калорій на 100 грам продукту

3.3.17 Клас RecipeObject

Ціль класу полягає у збереженні інформації про рецепт, тобто всіх існуючих даних в програмі про нього. 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 RecipeObject

Поля та методи класу	Опис
1	2
<i>public:</i>	
public int ID	Змінна для збереження ID рецепту
public int User_ID	Змінна для збереження ID автора рецепту

Продовження таблиці 3.17

1	2
public byte[] Content	Змінна для збереження картинки рецепту
public string Name	Змінна для збереження назви рецепту
public string Type	Змінна для збереження типу страви рецепту
public string Description	Змінна для збереження опису рецепту
public string Products	Змінна для збереження списку продуктів рецепту
public string Recipe_procedure	Змінна для збереження порядку дій рецепту
public string Number_of_servings	Змінна для збереження кількості порцій рецепту
public string Time	Змінна для збереження часу приготування рецепту
public DateTime Date_of_create	Змінна для збереження часу викладення рецепту
public string Comments	Змінна для збереження коментарів до рецепту
public int Rating	Змінна для збереження популярності рецепту

3.3.18 Клас RecipeDataBase

Ціль класу полягає у роботі з БД. Дані та методи класу наведені у таблиці 3.18.

Таблиця 3.18 – Опис полів та методів класу RecipeDataBase

Поля та методи класу	Опис
1	2
public:	
public bool AddServerUser(string Login, string Password)	Додавання акаунтів до БД
public bool AddRecipe(RecipeObject recipe)	Додавання рецептів до БД
public bool LoginExists(string Login)	Інформація про існування логіну
public int AccountExists(string Login, string Password)	Перевірка на існування акаунту з таким логіном та паролем
public string GetLoginByID(int ID)	Отримання логіну користувача за ID
public List<RecipeObject> GetAllRecipe()	Отримання списку всіх рецептів
public List<ProductObject> GetAllProductObject()	Отримання списку всіх продуктів
public string GetFavorite(int ID)	Отримати список улюблених рецептів для конкретного користувача
public bool UpdateFavorite(int ID, string Favorite)	Оновити список улюблених рецептів для конкретного користувача
public bool RecipeRatingUpdate(RecipeObject recipe)	Оновлення рейтингу рецепту
public bool UpdateRecipe(RecipeObject recipe)	Оновити рецепт
public bool RecipeDelete(RecipeObject recipe)	Видалити рецепт
private:	
private string connectionString	Змінна для підключення до БД

3.3.19 Клас ServerUser

Ціль класу полягає у збереженні інформації про активного користувача на сервері. Дані та методи класу наведені у таблиці 3.19.

Таблиця 3.19 – Опис полів та методів класу ServerUser

Поля та методи класу	Опис
1	2
<i>public:</i>	
public int ID	Змінна для збереження ID активного користувача
public string Login	Змінна для збереження логіну активного користувача
public string Password	Змінна для збереження паролю до акаунту активного користувача
public OperationContext operationContext	Змінна для збереження контексту операції.

3.3.20 Клас ServiceRecpie

Ціль класу полягає у реалізації функції інтерфейсу IServiceRecpie. Дані та методи класу наведені у таблиці 3.20.

Таблиця 3.20 – Опис полів та методів класу ServiceRecpie

Поля та методи класу	Опис
1	2
<i>public:</i>	
public int Connect(string Login, string Password)	Підключення користувача до серверу

Продовження таблиці 3.20

1	2
public void Disconnect(int ID)	Відключення користувача від серверу
public bool LoginCheck(string Login)	Перевірка на наявність логіну в БД
public bool Registration(string Login, string Password)	Реєстрація нового користувача
public List<RecipeObject> GetRecipe()	Отримання списку рецептів
public string GetFavorite(int ID)	Отримання списку улюблених рецептів
public bool UpdateFavorite(int ID, string Favorite)	Оновлення списку улюблених рецептів
public bool RecipeRatingUpdate(RecipeObject recipe)	Оновлення рейтингу рецептів
public bool UpdateRecipe(RecipeObject recipe)	Оновлення рецепту
public bool RecipeDelete(RecipeObject recipe)	Видалення рецепту
public string GetLoginByID(int ID)	Отримання логіну користувача за його ID
public bool AddRecipe(RecipeObject recipe)	Додавання рецепту
public List<ProductObject> GetAllProductObject	Отримання списку продуктів
private:	
RecpieDataBase dataBase	Змінні для роботи з БД
List<ServerUser> users	Список активних користувачив

3.3.21 Інтерфейс IServiceRecipe

Ціль інтерфейсу полягає у реалізації інтерфейсу для функцій серверу. Дані та методи інтерфейсу наведені у таблиці 3.21.

Таблиця 3.21 – Опис полів та методів інтерфейсу IServiceRecipe

Поля та методи класу	Опис
1	2
<i>public:</i>	
public int Connect(string Login, string Password)	Підключення користувача до серверу
public void Disconnect(int ID)	Відключення користувача від серверу
public bool LoginCheck(string Login)	Перевірка на наявність логіну в БД
public bool Registration(string Login, string Password)	Реєстрація нового користувача
public List<RecipeObject> GetRecipe()	Отримання списку рецептів
public string GetFavorite(int ID)	Отримання списку улюблених рецептів
public bool UpdateFavorite(int ID, string Favorite)	Оновлення списку улюблених рецептів
public bool RecipeRatingUpdate(RecipeObject recipe)	Оновлення рейтингу рецептів
public bool UpdateRecipe(RecipeObject recipe)	Оновлення рецепту
public bool RecipeDelete(RecipeObject recipe)	Видалення рецепту

Продовження таблиці 3.21

1	2
public string GetLoginByID(int ID)	Отримання логіну користувача за його ID
public bool AddRecipe(RecipeObject recipe)	Додавання рецепту
public List<ProductObject> GetAllProductObject	Отримання списку продуктів

3.3.22 Інтерфейс IServerRecpieCallback

Ціль інтерфейсу полягає у реалізації інтерфейсу для зворотного зв'язку між сервером та клієнтом.

3.3.23 Клас Program

Ціль класу полягає у реалізації роботи хост-частини. Дані та методи класу наведені у таблиці 3.22.

Таблиця 3.22 – Опис полів та методів класу Program

Поля та методи класу	Опис
1	2
<i>private:</i>	
static void Main(string[] args)	Запуск хост-частини

3.3.24 Клас RecpieHashing

Ціль класу полягає у реалізації шифрації паролю за допомоги хеш функції та замішуванні до паролю солі. Дані та методи класу наведені у таблиці 3.23.

Таблиця 3.23 – Опис полів та методів класу RespieHashing

Поля та методи класу	Опис
1	2
<i>public:</i>	
public static string GenerateSalt()	Генерація випадкової солі
public static string HashPassword(string password, string salt)	Шифрування паролю з використання хеш функції та солі

3.3.25 Клас SHA256Hash

Ціль класу полягає у реалізації хеш алгоритму SHA 256. Дані та методи класу наведені у таблиці 3.24.

Таблиця 3.24 – Опис полів та методів класу SHA256Hash

Поля та методи класу	Опис
1	2
<i>public:</i>	
public static string ComputeSHA256(string input)	Метод, який приймає рядок введення і повертає його SHA-256 хеш у вигляді рядка шістнадцяткових символів
<i>private:</i>	
private static uint RotR(uint x, int n)	Метод, який виконує циклічний зсув бітів числа x на n позицій вправо
private static uint Sum0(uint x)	Метод, який виконує операцію, що включає в себе ряд операцій з циклічними зсувами бітів

Продовження таблиці 3.24

1	2
private static uint Sum1(uint x)	Метод, який також виконує операцію з циклічними зсувами, але з іншими параметрами
private static uint Ch(uint x, uint y, uint z)	Метод, що реалізує логічну операцію "змішання" (bitwise operations) між параметрами x, y та z
private static uint Maj(uint x, uint y, uint z)	Метод, що виконує логічну операцію "більшість" (bitwise operations) між параметрами x, y та z
private static readonly uint[] K = new uint[]	Константний масив K, який використовується у функції SHA-256 для виконання кількох раундів обчислень. Ці константи K використовуються разом з іншими операціями, щоб забезпечити стійкість і непередбачуваність алгоритму

3.4 Проектування бази даних

Концептуальна схема – це єдиний логічний опис всіх елементів даних і відносин між ними, логічна структура всієї бази даних. Для кожної бази даних є тільки одна концептуальна схема [43].

Концептуальна схема БД для застосунку «Ресріє» представлена на рисунку 3.21.

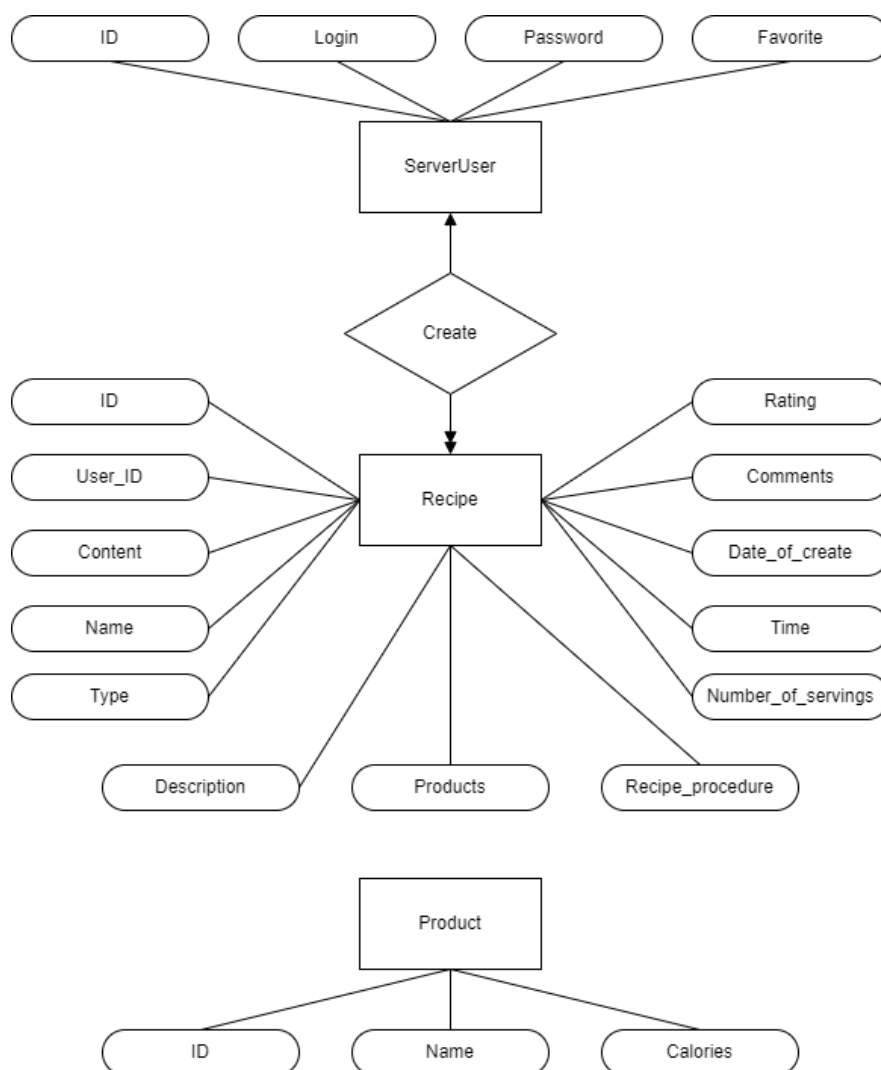


Рисунок 3.21 – Концептуальна модель бази даних

Сутність «Recipe» (ID, User_ID, Content, Name, Type, Description, Products, Recipe_procedure, Number_of_servings, Time, Date_of_create, Comments, Rating) – опис полів сутності та їх типу наведено у таблиці 3.25.

Таблиця 3.25 – Опис полів сутності «Recipe»

№	Назва поля	Тип даних	Розмір
1	ID	int	4 байти
2	User_ID	int	4 байти
3	Content	varbinary(MAX)	2147483,647 байт
4	Name	nvarchar(MAX)	1073741,823 байт
5	Type	nvarchar(MAX)	1073741,823 байт

Продовження таблиці 3.25

6	Description	nvarchar(MAX)	1073741,823 байт
7	Products	nvarchar(MAX)	1073741,823 байт
8	Recipe_procedure	nvarchar(MAX)	1073741,823 байт
9	Number_of_servings	nvarchar(MAX)	1073741,823 байт
10	Time	nvarchar(MAX)	1073741,823 байт
11	Date_of_create	date	3 байти
12	Comments	nvarchar(MAX)	1073741,823 байт
13	Rating	int	4 байти

Сутність «ServerUser» (ID, Login, Password, Favorite) – опис полів сутності та їх типу наведено у таблиці 3.26.

Таблиця 3.26 – Опис полів сутності «ServerUser»

№	Назва поля	Тип даних	Розмір
1	ID	int	4 байти
2	Login	nchar(30)	60 байтів
3	Password	nvarchar (MAX)	1073741,823 байт
4	Favorite	nvarchar(MAX)	1073741,823 байт
5	Salt	nvarchar(MAX)	1073741,823 байт

Сутність «Product» (ID, Name, Calories) – опис полів сутності та їх типу наведено у таблиці 3.27.

Таблиця 3.27 – Опис полів сутності «Product»

№	Назва поля	Тип даних	Розмір
1	ID	int	4 байти
2	Name	nvarchar(MAX)	1073741,823 байт
3	Calories	float	8 байтів

3.5 Основні рішення щодо розробки інтерфейсу

Колір відіграє важливу роль у сприйнятті світу, і він відіграє ключову роль у сприйнятті інтерфейсу. Колір допомагає нам передбачати, розуміти та приймати рішення.

Бачення на вигляд програмного застосунку у користувачів різне. Саме через різні погляди на дизайн, приємну кольорову гаму, було вирішено обрати світлу та приємну для людей рожево-помаранчеву гаму.

Світлі відтінки не позбавлені своєї індивідуальності. Вони можуть бути рожевими, зеленими, бузковими. Людина розрізняє до 100 000 відтінків і 1/3 з них світлі. Говорять що світлі кольори – психологічно комфортні, а особливо теплі відтінки. Також вони поєднуються з багатьма іншими кольорами. Кольорова гама інтерфейсу зображена на рисунку 3.22.

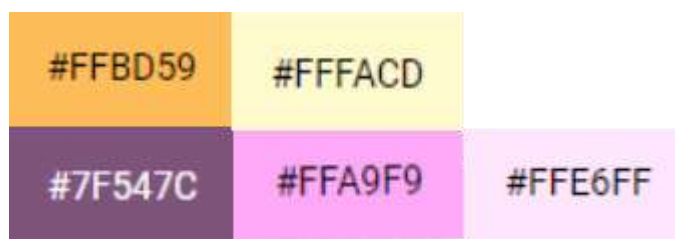


Рисунок 3.22 – Кольорова гама

Поєднання рожевого та помаранчевого приваблює своєю тропічною пишністю. Воно побудована на легкому тепло-холодному контрасті і рідко виходить за ці межі, тому що світлі рожеві відтінки знижують її колорит, а темні насичені роблять його надмірно яскравим. Таке поєднання кольорів нагадує фрукт персик [44].

Жовтий колір привертає увагу, символізує тепло та бадьорість, асоціюється з достатком та активністю. У дизайні його зазвичай використовують для акцентів: як основний колір жовтий може втомлювати.

Для створення інтерфейсу було використано WPF – графічна підсистема (аналог WinForms). Система ресурсів WPF дозволяє визначати

об'єкти в одному місці, а потім повторно використовувати їх в інших частинах розмітки. Для цього було створено файл «ResDictionary».

Також для того, щоб збільшити привабливість інтер'єсу було використано градієнти. На рисунку 3.23 зображено градієнти для елементів інтерфейсу.

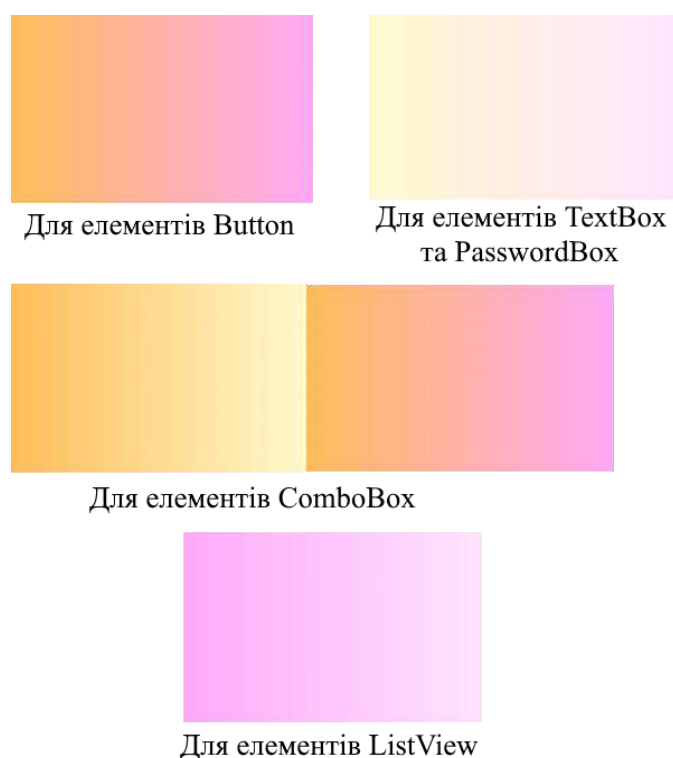


Рисунок 3.23 – Градієнти

Приклад коду встановлення кольору для елементів Button

```
<Setter Property="Background">
  <Setter.Value>
    <LinearGradientBrush StartPoint="0,0" EndPoint="1,1">
      <GradientStop Color="#FFBD59" Offset="0"/>
      <GradientStop Color="#ffa9f9" Offset="1"/>
    </LinearGradientBrush>
  </Setter.Value>
</Setter>
```

Приклад коду встановлення кольору для елементів TextBox

```
<Setter Property="Background">
  <Setter.Value>
    <LinearGradientBrush StartPoint="0,0" EndPoint="1,1">
      <GradientStop Color="#fffacd " Offset="0"/>
      <GradientStop Color="#ffe6ff" Offset="1"/>
    </LinearGradientBrush>
  </Setter.Value>
</Setter>
```

```
</Setter.Value>
</Setter>
```

На рисунку 3.24 зображено інтерфейс для вікна «Авторизація». Фон для цього вікна було намальовано окремо в сервісі «Canva».



Рисунок 3.24 – Вікно «Авторизація»

Закруглені елементи сприймаються легше. Вони виглядають сучасно. Використання закруглених форм почалося з мобільних інтерфейсів і згодом поширилося на програми для персональних комп'ютерів. Відсутність гострих кутів створює відчуття простоти, оптимізму та доступності. Можливо, саме тому багато дизайн-систем впроваджують закруглені елементи і широко застосовують їх для іконок, кнопок і навіть зображень (наприклад, у новому інтерфейсі Windows 11).

В створеному інтерфейсі також були використані закруглені елементи, а саме: Button, TextBox, ListView, ComboBox, PasswordBox. Також замість границь у елементів була створена тінь. Це зроблено для того, щоб інтерфейс виглядав більш плавним [45].

Приклад коду закруглення та створення тіней замість границі для елементів TextBox

```
<Setter Property="Template">
  <Setter.Value>
    <ControlTemplate TargetType="TextBox">
      <Border Background="{TemplateBinding Background}"
        BorderBrush="{TemplateBinding BorderBrush}"
        BorderThickness="{TemplateBinding BorderThickness}"
        CornerRadius="10">
        <Border.Effect>
          <DropShadowEffect Color="#7f547c" Direction="270"
ShadowDepth="2" Opacity="0.7"/>
        </Border.Effect>
        <ScrollViewer x:Name="PART_ContentHost" />
      </Border>
    </ControlTemplate>
  </Setter.Value>
</Setter>
```

На рисунку 3.25 зображено інтерфейс для вікна «Реєстрація». Фон для цього вікна було намальовано окремо в сервісі «Canva».

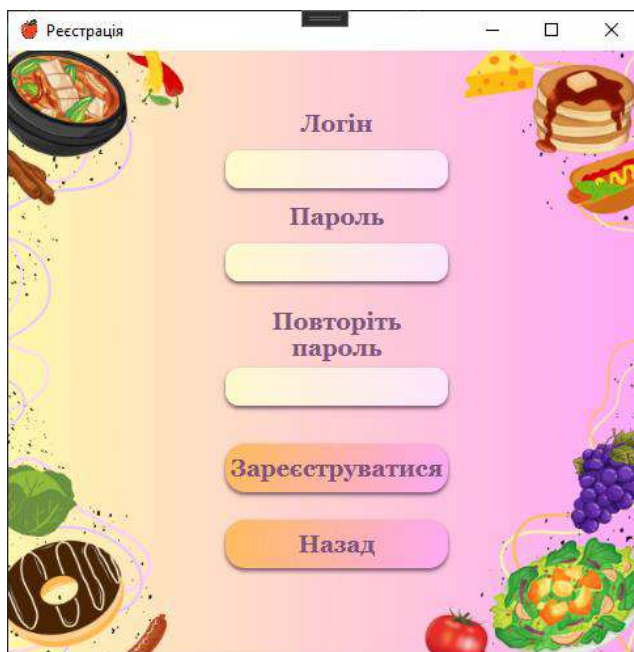


Рисунок 3.25 – Вікно «Реєстрації»

Головне вікно представлено на рисунку 3.26. Фон для цього вікна було намальовано окремо в сервісі «Canva».



Рисунок 3.26 – Головне вікно

Для того, щоб користувачу було зрозуміло, яку кнопку він обрав, було створено зміну кольору кнопки при наведенні.

Приклад коду зміну кольору кнопки при наведенні:

```
<ControlTemplate.Triggers>
  <Trigger Property="IsMouseOver" Value="True">
    <Setter Property="Background" Value="#FFBD59"/>
  </Trigger>
</ControlTemplate.Triggers>
```

Вікно для відображення рецепту, представлено на рисунку 3.27 має схожий фон з головним вікном. Це зроблено для того, щоб користувачам було зручно при відкритті рецепту для детальної інформації.

Рецепт

Автор: admin

Назва: Картопля смажена з грибами

Тип: Друге

Дата викладення: 6.4.2023

Час приготування: 35 хв

Кількість порцій: 4

Опис

З усіх страв домашньої кухні смажена картопля є головною улюбленицею багатьох сімей. Готується вона легко і швидко, а хрустка скоринка і яскравий смак сподобаються абсолютно всім. В кінці осені настає грибний сезон, так що до звичної страви можна додати ароматні смажені гриби.

Список продуктів

Гриби - 150 г.
Картопля - 5-6 шт.
Ріпчаста цибуля велика - 1 шт.
Сіль, перець.
Кріп, зелена цибуля.
Рослинна олія - 5 ст. л.

Порядок дій

Картоплю почистити і нарізати соломкою, гриби залишити цілими, великі розрізати на 4 частини.
У сковороді нагріти олію, покласти картоплю.
Смажити на великому вогні до повної готовності (приблизно 20 хвилин), постійно помішуючи.
На іншій сковороді смажити нарізану півкільцями цибулю до м'якості, потім додати гриби. Смажити 5 хвилин. Посолити і поперчити. Смажити ще 5-7 хвилин.
З'єднати рум'яну картоплю і смажені гриби. Перемішати і обсмажити разом 3 хвилини. Додати рубану зелень. Подавати на стіл в гарячому вигляді.

Коментар

6.6.2023 18:21 Login1: Дуже чудовий рецепт!
6.6.2023 18:21 admin: Дякую за відгук

Додати

Назад

Видалити

Редагувати

Рисунок 3.27 – Вікно для відображення рецепту

Приклад коду для завантаження фону:

```
<Window.Background>
  <ImageBrush ImageSource="Background\main2.png" />
</Window.Background>
```

На рисунку 3.28 зображено інтерфейс для вікна для створення рецепту. Фон для цього вікна було намальовано окремо в сервісі «Canva».

The image shows a software window titled "Створення рецепту" (Recipe Creation). The window has a pink background with a pattern of orange wavy lines. It contains several input fields and buttons:

- Назва** (Name): A text input field.
- Опис** (Description): A large text area.
- Тип страви** (Dish Type): A dropdown menu.
- Завантажити** (Load): A button.
- Список продуктів** (List of products): A large text area.
- Порядок дій** (Order of operations): A large text area.
- Час приготування** (Cooking time): A text input field.
- Кількість порцій** (Number of portions): A text input field.
- Назад** (Back): A button.
- Створити** (Create): A button.

Рисунок 3.28 – Вікно для створення рецепту

Для розширеного пошуку було вирішено створити окреме вікно, яке зображено на рисунку 3.29. Також було створено додаткове вікно, щоб реалізувати пошук за інгредієнтом.

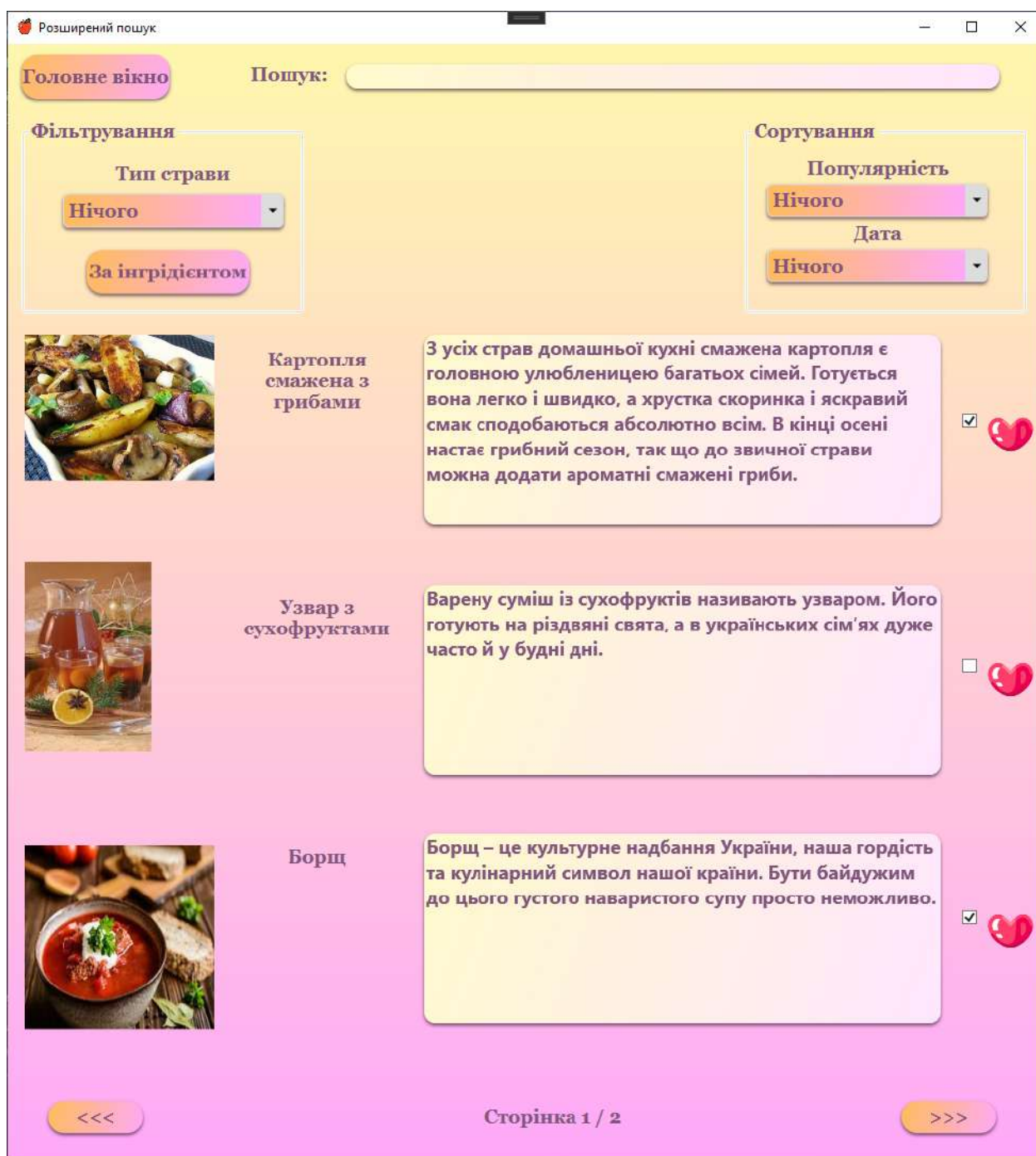


Рисунок 3.29 – Вікно для розширеного пошуку

Для додаткового вікна було розроблено зображення для різних видів продуктів, які представлено на рисунку 3.30

Ще однією важливою можливістю є додавання улюблених рецептів в окреме місце та зберігання власних рецептів в окремому місці. Саме для цього створено вікно «Улюблені та мої рецепти», що представлено на рисунку 3.32.



Рисунок 3.32 – Вікно «Улюблені та мої рецепти»

Також застосунок має довідкові вікна: «Довідка» та «Про програму», які зображено на рисунку 3.33. Фон для вікна «Про програму» було намальовано окремо в сервісі «Canva».

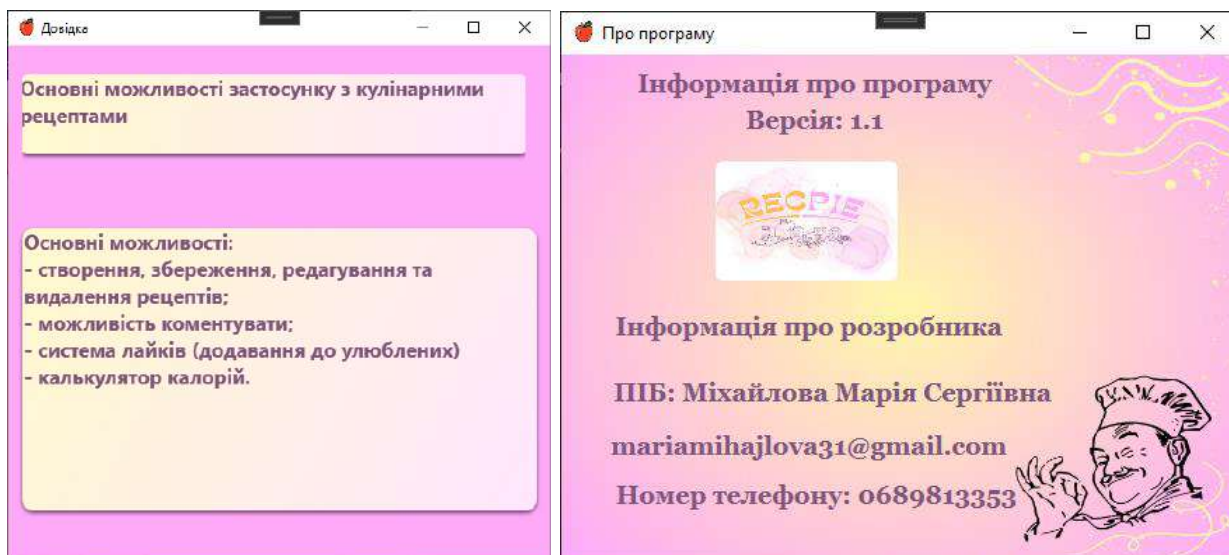


Рисунок 3.33 – Довідкові вікна.

Для виключних ситуації реалізовано вікно помилка. Як приклад показано на рисунку 3.34 сповіщення про помилку, коли сервіс не знайдено.

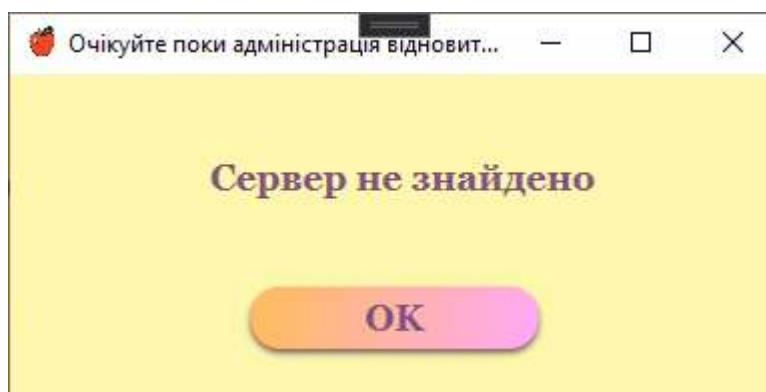


Рисунок 3.34 – Сповіщення про помилку

Як і кожний застосунок, програмне забезпечення «Респіє» має власну іконку, яке зображено на рисунку 3.35



Рисунок 3.35 – Іконка ПЗ «Респіє»

Для застосування «Ресріє» було створено логотип в сервісі «Canva», що показаний на рисунку 3.36



Рисунок 3.35 –Логотип ПЗ «Ресріє»

3.6 Висновки за розділом 3

В результаті було:

- створено функціональну схему ПЗ;
- розроблено та описано основні алгоритми;
- обрано та використано шаблон проєктування – клієнт-сервер;
- створені та описані класи та інтерфейси;
- описано та розроблено БД проєкту;
- розроблено та описано створений інтерфейс.

4 КЕРІВНИЦТВО ПРОГРАМІСТА

В даному розділі розглянуто призначення програми, умови її застосування, функції, характеристика, структура, засоби звертання, вхідні та вихідні дані, а також повідомлення які може відобразити застосунок.

4.1 Призначення та умови застосування програми

4.1.1 Призначення програми

Програма призначена для надання користувачу зручного інтерфейсу користувачу зручного інтерфейсу для організації роботи з рецептами, що дозволяє легко зберігати, організовувати та швидко знаходити рецепти будь-яких страв за допомогою вдосконаленого пошуку.

4.1.2 Функції програми

Функції, що виконує програма:

- реєстрація та авторизація користувачів. Можливість реєстрації нових користувачів і авторизації існуючих;
- шифрування введеного пароля користувача;
- збереження та організація рецептів. Можливість додавати, редагувати, видаляти рецепти та додавати до улюблених;
- пошук та фільтрація рецептів. Можливість швидкого та ефективного пошуку рецептів залежно від різних параметрів, таких як назва, інгредієнти, тип страви. Можливість сортувати рецепти за популярністю та датою;
- рецепти з кроками та інгредієнтами. Додавання до рецепту докладні кроки готування для кожного рецепту, а також перелік необхідних інгредієнтів з відповідними кількостями;

- відображення характеристик рецептів. Надання інформації про час приготування, кількість порцій та інші характеристики рецепту;
- обмін рецептами та спільнота. Можливість ділитися своїми рецептами, залишати коментарі та відгуки;
- калькулятор калорій. Можливість розраховувати калорійність страви.

4.1.3 Умови застосування програми для клієнтської та серверної частини

Для адекватної роботи з програмою, потрібно щоб пристрій, який буде використовуватися, задовольняв таким програмним та технічним вимогам:

- потрібна оперативна пам'ять: мінімум 4 Гб;
- потрібна місце на диску: мінімум 400 Мб;
- операційна система: Windows 10;
- жорсткий диск об'ємом мінімум 128 Гб.

Для компіляції та внесення змін до програми рекомендується середовище Visual Studio 2019 та платформа .NET Framework 4.7.2.

4.2 Характеристика програми

Програма створена мовою C# у середовищі Visual Studio 2019 на платформі .NET Framework 4.7.2 з графічною підсистемою WPF.

Застосунок має головне вікно на якому відображено всі завантажені існуючі рецепти, поле пошуку та кнопки для переходу на інші вікна. Також при подвійному кліку лівою кнопкою миші по назві рецепта, відкривається детальна інформація про нього.

4.2.1 Структура програми

Щоб почати роботу з клієнтською частиною програми, спочатку потрібно запустити хост-частину. Після вдалого запуску хоста користувачі можуть запускати клієнтську частину.

При початку роботи програми клієнтської частини завантажується вікно з авторизацією.

Загальна структура програми складається з: файлів бібліотек, програмного коду, файлів типу *.xaml та *.config. Структура наведена на рисунку 4.1

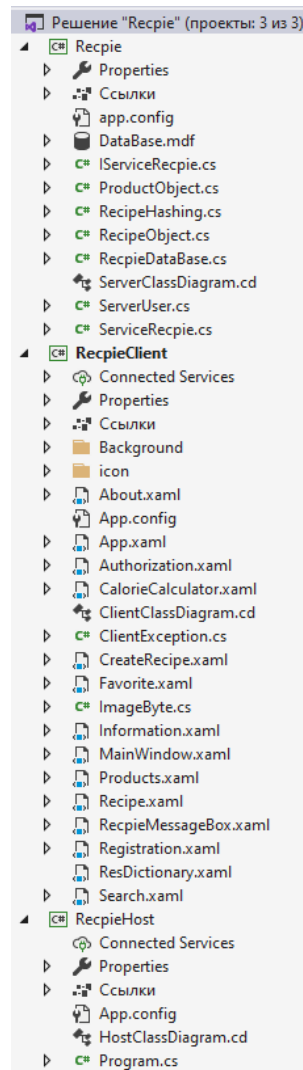


Рисунок 4.1 – Структура програми

4.3 Звертання до програми

Програма має такі методи звертання до неї:

- ліва (рецептами та кнопками) клавіша комп'ютерної миші;
- клавіші клавіатури (ввід назви рецепту, логіну, паролю, інформації про рецепт).

4.4 Вхідні та вихідні дані

4.4.1 Вхідні дані

Програма отримує такі вхідні дані як:

- сигнал про натискання клавіш миші та клавіатури;
- індекс обраного користувачем елементу списку;
- інформація про обрані прапорці продуктів та при додаванні до улюблених;
- БД;
- зображення іконки програми, продуктів, зображення рецептів та фонів інтерфейсу.

4.4.2 Вихідні дані

Програма видає такі вихідні дані як:

- інформація про рецепти;
- спливаючі підказки;
- інформація про програму та розробника;
- інформація про розраховані калорії страв;
- інформація про основний функціонал ПЗ.

4.5 Повідомлення

Програма містить такі повідомлення як:

- «Необхідно авторизуватись» – означає, що даний функціонал недоступний без авторизації;
- «Неправильний логін або пароль» – означає, що при авторизації введено або неіснуючий акаунт, або неправильний пароль;
- «Акаунт створено» – означає, що акаунт успішно створено;
- «Проблема з базою даних» – означає, що акаунт не створено через проблему з базою даних;
- «Паролі не співпадають» – означає, що акаунт не створено через те що підтвердження паролю не співпало з введеним паролем;
- «Пароль некоректний» – означає, що акаунт не створено через те що введений пароль не пройшов валідацію;
- «Логін зайнятий» – означає, що акаунт не створено через те що введений логін був вже зареєстрован;
- «Логін некоректний» – означає, що акаунт не створено через те що введений логін не пройшов валідацію;
- «Сервер не знайдено» – означає, що інформація з сервера не отримана чи відправлена, через неможливість з ним зв'язатись;
- «Рецепт додано» – означає, рецепт створено;
- «Рецепт не додано» – означає, рецепт не створено;
- «Рецепт оновлено» – означає, значення рецепту оновлено на введені дані;
- «Рецепт не оновлено» – означає, рецепт не вдалось оновити.

5 КЕРІВНИЦТВО ОПЕРАТОРА

В даному розділі розглянуто призначення програми, умови її виконання, процес виконання програми та повідомлення для користувача.

5.1 Призначення програми

Програма призначена для надання користувачу зручного інтерфейсу користувачу зручного інтерфейсу для організації роботи з рецептами, що дозволяє легко зберігати, організовувати та швидко знаходити рецепти будь-яких страв за допомогою вдосконаленого пошуку.

5.2 Умови виконання програми

5.2.1 Апаратні вимоги програми

Для адекватної роботи з програмою, потрібно щоб пристрій, який буде використовуватися, задовольняв таким програмним та технічним вимогам:

- потрібна оперативна пам'ять: мінімум 4 Гб;
- потрібна місце на диску: мінімум 400 Мб;
- операційна система: Windows 10;
- жорсткий диск об'ємом мінімум 128 Гб.

5.2.2 Вимоги до операторів

Для комфортної роботи з програмою від користувача потрібні:

- знання та володіння українською мовою;
- навички користування комп'ютерною мишею та клавіатурою;
- ознайомлення з довідкою щодо користування програмою.

5.3 Виконання програми

5.3.1 Запуск програми

Програма запускається за допомогою виконуваного файлу RespieClient.exe та здійснюється подвійним натисканням лівої клавіші комп'ютерної миші по ньому або його ярлику.

5.3.2 Виконання роботи з програмою

При запуску програми користувачу відображається вікно «Авторизації», яке зображено на рисунку 5.1.

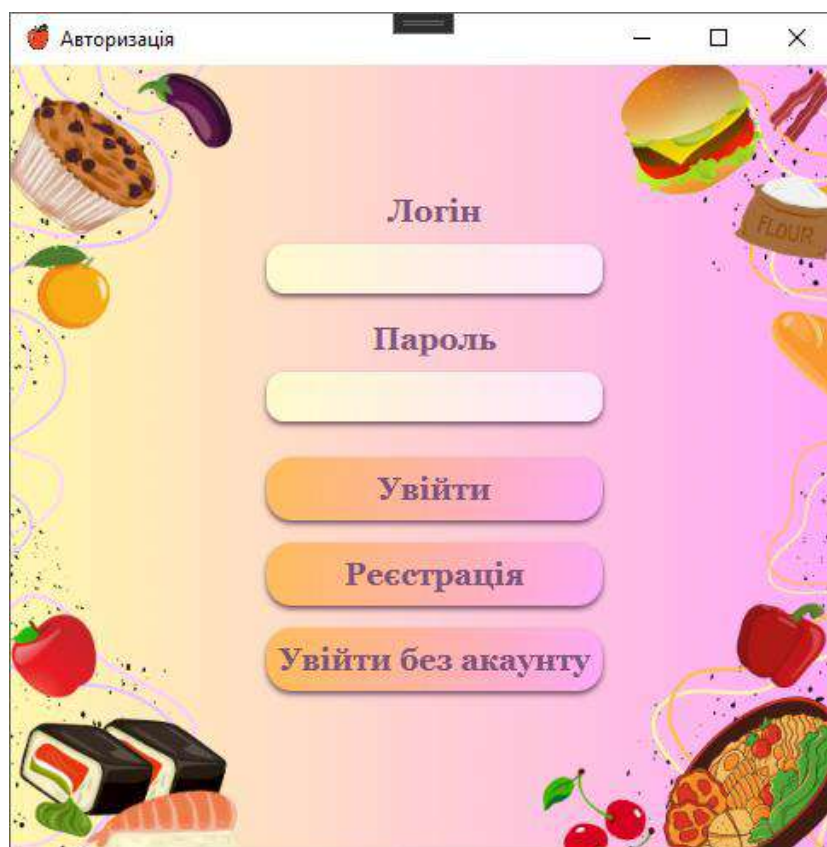


Рисунок 5.1 – Вікно «Авторизація»

Робота зі застосунком можлива без авторизації. Для цього необхідно натиснути кнопку «Увійти без акаунту». Після чого користувач побачить

головне вікно, але функціонал буде обмежений. Не можна: створювати нові рецепти, додавати рецепти до улюбленого, передивлятися свої та улюблені рецепти та додавати коментарі до рецептів.

Для того щоб увійти в свій акаунт користувачу треба ввести логін та пароль, а потім натиснути на кнопку «Увійти».

Якщо у користувача немає акаунту то його можна створити, натиснувши кнопку «Реєстрація». Після чого відкривається вікно, яке представлено на рисунку 5.2.

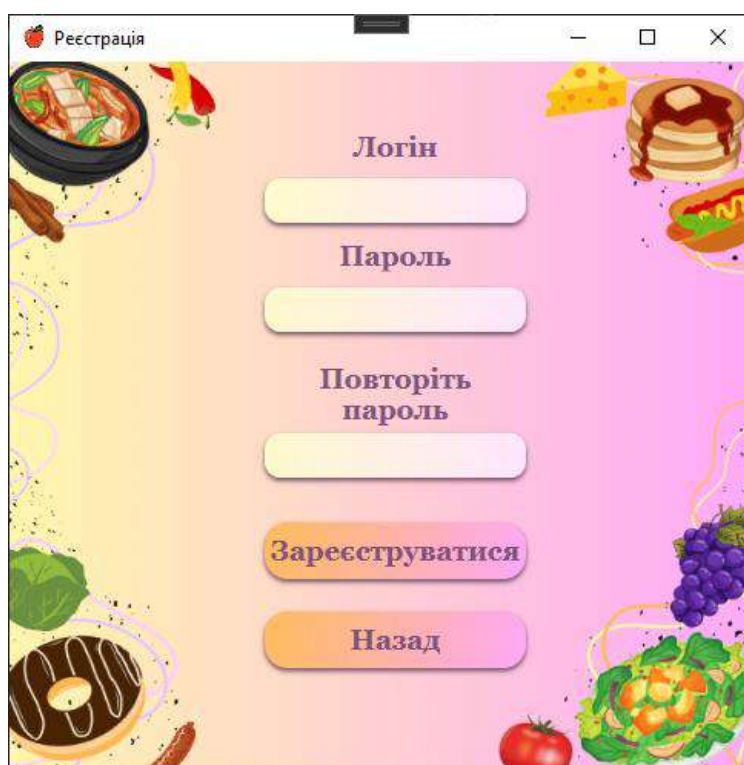
The image shows a software window titled "Реєстрація" (Registration). The window has a light pink background decorated with various food items like a bowl of soup, a pancake, a salad, and a tomato. In the center, there are three text input fields labeled "Логін" (Login), "Пароль" (Password), and "Повторіть пароль" (Repeat password). Below these fields are two buttons: "Зареєструватися" (Register) and "Назад" (Back). The window also features standard window control buttons (minimize, maximize, close) in the top right corner.

Рисунок 5.2 – Вікно «Реєстрація»

У вікні «Реєстрація» користувачу необхідно ввести логін та пароль, а потім пароль ще раз повторити. Якщо паролі не співпадають, то буде сповіщення про помилку, яке зображено на рисунку 5.3

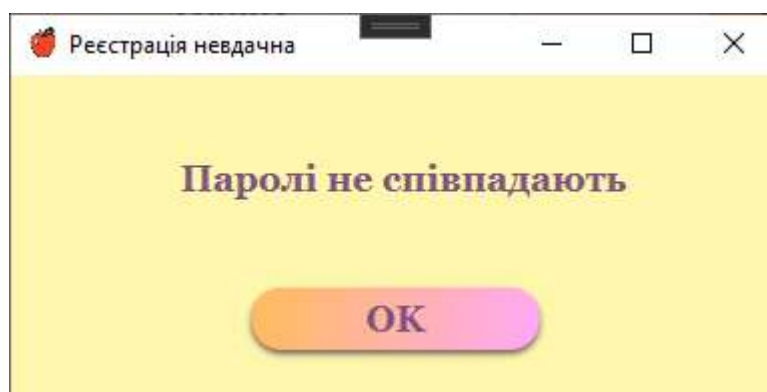


Рисунок 5.3 – Сповіщення про помилку

Після вдалої авторизації, перед користувачем постає головне вікно програми «Ресріє», яке показано на рисунку 5.4.



Рисунок 5.4 – Головне вікно

У червоному прямокутнику на рисунку 5.4 знаходиться основна область вікна. Тут знаходиться коротка інформація про три рецепта. Для переходу на інші сторінки треба натиснути кнопки, які виділено прямокутниками зеленого кольору на рисунку 5.4. Для переходу на інші вікна створені кнопки, які виділено прямокутником синього кольору на рисунку 5.4.

При подвійному кліку лівою кнопкою миші по назві рецепта, відкривається детальна інформація про нього, що представлено як вікно на рисунку 5.5.

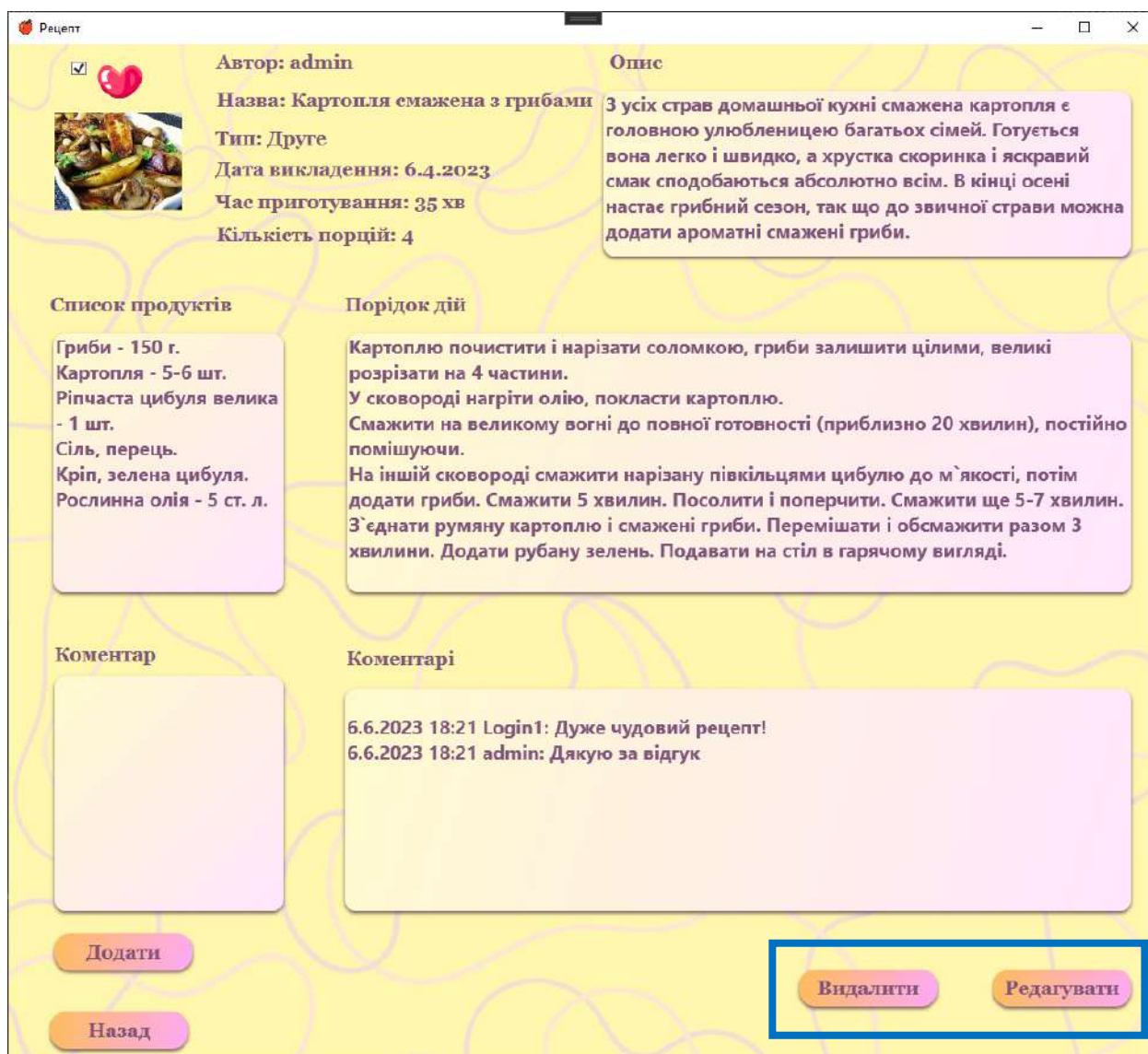


Рисунок 5.5 – Вікно для відображення рецепту

У вікні на рисунку 5.5 відображаються кнопки у синьому прямокутнику тільки тоді, коли користувач є автором цього рецепту. Завдяки цим кнопкам користувач може вносити зміни до рецепту або видаляти його зовсім.

Також для користувача є окреме вікно, де знаходиться його власні рецепти та улюблене. Це вікно показано на рисунку 5.6.



Рисунок 5.6 – Вікно «Улюблені та мої рецепти»

Це вікно створено окремо для того, щоб користувачу було зручно знайти свої та улюблені рецепти. Щоб змінити тип відображаємих рецептів треба обрати інший елемент у спливаючому списку, який виділено прямокутником синього кольору на рисунку 5.6.

Для створення та редагування є окреме вікно, що показано на рисунку 5.7. У цьому вікні користувач може надавати всю інформацію про рецепт.

Рисунок 5.7 – Вікно для створення рецепту

Для покращення пошуку рецептів за вподобанням користувачів існує окреме вікно з розширеним пошуком, яке зображено на рисунку 5.8.

Прямокутником жовтого кольору на рисунку 5.8 виділено область, де можна застосувати сортування за популярністю та датою. Прямокутником блакитного кольору на рисунку 5.8 виділено поле для пошуку за назвою (ця функція доступна і в інших вікна з зображенням декількох рецептів). Прямокутником синього кольору на рисунку 5.8 виділено область, де можна застосувати фільтрування за типом страви.



Рисунок 5.8 – Вікно для розширеного пошуку

Прямокутником фіолетового кольору на рисунку 5.8 виділено кнопку за допомогою якої можна фільтрувати рецепти за інгредієнтами. Після натиснення на неї відкривається додаткове вікно, яке представлено на рисунку 5.9.



Рисунок 5.9 – Вікно з продуктами

На цьому вікні користувачу треба обрати необхідні продукти. Якщо користувачу не зрозумілі якісь зображення продуктів, то створені спливаючі підказки. Після фільтрування та сортування на вікні, що зображено на рисунку 5.8 буде відображено необхідні рецепти. Область, де це буде знаходитися, виділена прямокутником червоного кольору на рисунку 5.8.

Якщо користувач хоче підрахувати калорійність якоїсь страви, то він може скористатися вікно «Калькулятор калорій». Воно зображено на рисунку 5.10.

Для введення необхідних продуктів, користувачу необхідно обрати назву продукту та вказати його кількість в грамах. Ця можливість виділена прямокутником синього кольору на рисунку 5.10. Потім треба додати цей продукт за допомогою кнопки. Кнопки для взаємодії з продуктами показано

прямокутником фіолетового кольору на рисунку 5.10 Після чого продукт додається у список, який виділено червоним прямокутником на рисунку 5.10.

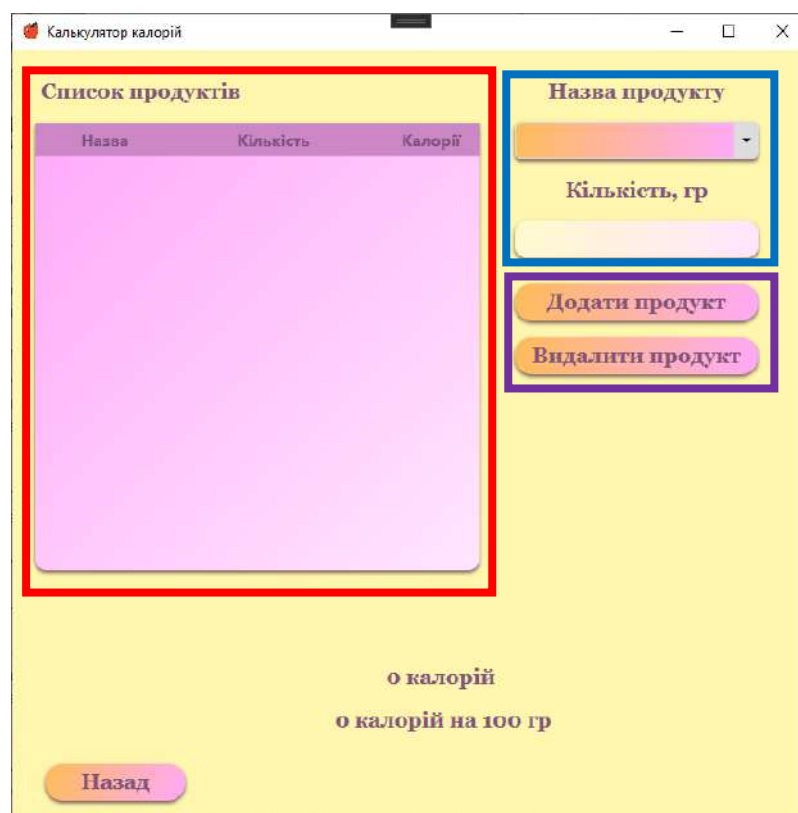


Рисунок 5.10 – Вікно «Калькулятор калорій»

Для додаткової інформації про розробника чи про саму програму створені довідкові вікна, які представлено на рисунку 5.11.

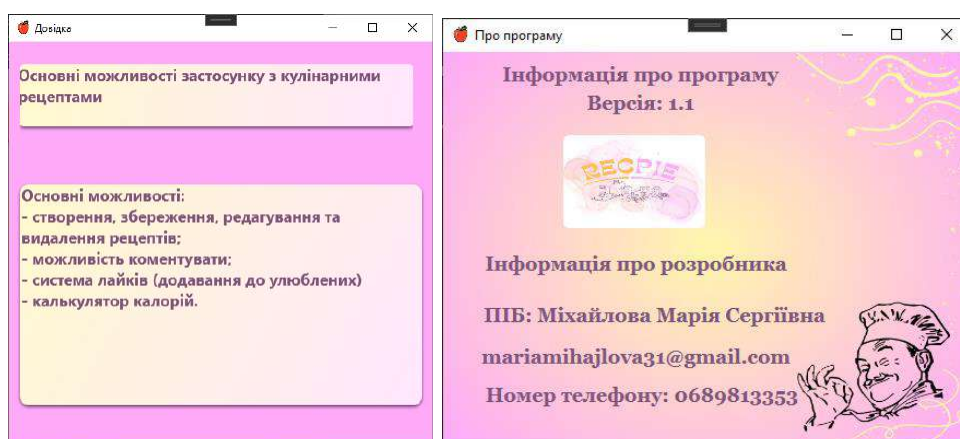


Рисунок 5.11 – Довідкові вікна.

5.4 Повідомлення оператору

Програма містить такі повідомлення як:

- «Неправильний логін або пароль» – означає, що при авторизації введено або неіснуючий акаунт, або неправильний пароль;
- «Акаунт створено» – означає, що акаунт успішно створено;
- «Проблема з базою даних» – означає, що акаунт не створено через проблему з базою даних;
- «Паролі не співпадають» – означає, що акаунт не створено через те що підтвердження паролю не співпало з введеним паролем;
- «Пароль некоректний» – означає, що акаунт не створено через те що введений пароль не пройшов валідацію;
- «Логін зайнятий» – означає, що акаунт не створено через те що введений логін був вже зареєстрован;
- «Логін некоректний» – означає, що акаунт не створено через те що введений логін не пройшов валідацію;
- «Сервер не знайдено» – означає, що інформація з сервера не отримана чи відправлена, через неможливість з ним зв'язатись;
- «Рецепт додано» – означає, рецепт створено;
- «Рецепт не додано» – означає, рецепт не створено;
- «Рецепт оновлено» – означає, значення рецепту оновлено на введені дані;
- «Рецепт не оновлено» – означає, рецепт не вдалось оновити.

6 ТЕСТУВАННЯ І РЕЗУЛЬТАТИ

6.1 Опис методики проведення тестування

Тестування буде проведене за таким сценарієм:

а) реєстрація акаунту:

- 1) з некоректним логіном;
- 2) з некоректним паролем;
- 3) з існуючим логіном;
- 4) з неправильно повтореним паролем;
- 5) правильний логін та пароль;
- 6) перевірка коректного збереження паролю у БД;

б) авторизація в застосунок:

- 1) невведений логін чи пароль;
- 2) неіснуючий акаунт;
- 3) неправильний пароль;
- 4) правильний логін та пароль;
- 5) перевірка коректного шифрування введеного паролю;

в) головне вікно:

- 1) відображення рецептів;
- 2) перехід по сторінкам;
- 3) відкриття рецептів;
- 4) пошук за назвою;
- 5) додавання та видалення з улюблених;
- 6) вихід з акаунту;
- 7) перехід до інших вікон;

г) вікно з рецептом:

- 1) відображення рецепту;
- 2) відображення кнопок видалення та редагування для автора;
- 3) відображення кнопок видалення та редагування для не

автора;

- 4) видалення рецепту;
- 5) перехід до редагування рецепту;
- 6) коментування рецепту;
- д) вікно для створення рецепту:
 - 1) завантаження зображення;
 - 2) створення рецепту;
 - 3) збереження відредагованого рецепту;
- е) вікно «Мої та улюблені»:
 - 1) відображення моїх рецептів;
 - 2) відображення улюблених рецептів;
 - 3) перехід по сторінкам;
 - 4) відкриття рецептів;
 - 5) пошук за назвою;
 - 6) додавання та видалення з улюблених;
- ж) вікно з розширеним пошуком:
 - 1) відображення рецептів;
 - 2) перехід по сторінкам;
 - 3) відкриття рецептів;
 - 4) пошук за назвою;
 - 5) додавання та видалення з улюблених;
 - 6) фільтрування за типом;
 - 7) фільтрування за продуктом;
 - 8) сортування за датою;
 - 9) сортування за популярністю;
- з) вікно «Калькулятор калорій»:
 - 1) можливість обрати продукт;
 - 2) введення некоректної кількості продукту;
 - 3) введення коректної кількості продукту;
 - 4) видалення продукту;
 - 5) розрахування кількості калорій.

Тестування буде проводитися на ПК, з характеристиками:

- оперативна пам'ять 16 Гб;
- операційна система Windows 10;
- жорсткий диск об'ємом 1 Тб.

6.2 Результати використання розробленого програмного забезпечення

6.2.1 Реєстрація акаунту

В таблиці 6.1 наведено тестування вікна «Реєстрація».

Таблиця 6.1 – Тестування реєстрації

Назва	Результат	Успішність
Некоректний логін	Реєстрація невдала	+
Некоректний пароль	Реєстрація невдала	+
Існуючий логін	Реєстрація невдала	+
Неправильно повторений пароль	Реєстрація невдала	+
Правильний логін та пароль	Реєстрація вдала	+
Перевірка коректного збереження паролю у БД	Пароль зашифровано	+

6.2.2 Авторизація

В таблиці 6.2 наведено тестування вікна «Авторизація».

Таблиця 6.2 – Тестування авторизації

Назва	Результат	Успішність
Невведений логін чи пароль	Авторизація невдала	+
Неіснуючий акаунт	Авторизація невдала	+
Неправильний пароль	Авторизація невдала	+
Правильний логін та пароль	Авторизація вдала	+
Перевірка коректного шифрування введеного паролю	Шифрація виконується правильно	+

6.2.3 Головне вікно

В таблиці 6.3 наведено тестування головного вікна.

Таблиця 6.3 – Тестування головного вікна

Назва	Результат	Успішність
Відображення рецептів	Коректне відображення	+
Перехід по сторінкам	Коректне перехід	+
Відкриття рецептів	Коректне відкриття	+
Пошук за назвою	Коректний пошук	+
Додавання та видалення з улюблених	Коректне додавання та видалення	+
Вихід з акаунту	Коректний вихід	+
Перехід до інших вікон	Коректний перехід	+

6.2.4 Вікно з рецептом

В таблиці 6.4 наведено тестування вікна з рецептом.

Таблиця 6.4 – Тестування вікна з рецептом

Назва	Результат	Успішність
Відображення рецепту	Коректне відображення	+
Відображення кнопок видалення та редагування для автора	Кнопки відображаються	+
Відображення кнопок видалення та редагування для не автора	Кнопки не відображаються	+
Видалення рецепту	Коректне видалення	+
Перехід до редагування рецепту	Коректний перехід	+
Коментування рецепту	Коректне додавання коментарів	+

6.2.5 Вікно створення рецепту

В таблиці 6.5 наведено тестування вікна для створення рецепту.

Таблиця 6.5 – Тестування вікна для створення рецепту

Назва	Результат	Успішність
Завантаження зображення	Коректне	+
Створення рецепту	Коректне	+
Збереження відредагованого рецепту	Коректне	+

6.2.6 Вікно «Мої та улюблені»

В таблиці 6.6 наведено тестування вікна «Мої та улюблені».

Таблиця 6.6 – Тестування вікна «Мої та улюблені»

Назва	Результат	Успішність
Відображення моїх рецептів	Коректне відображення	+
Відображення улюблених рецептів	Коректне відображення	+
Перехід по сторінкам	Коректний перехід	+
Відкриття рецептів	Коректне відкриття	+
Пошук за назвою	Коректний пошук	+
Додавання та видалення з улюблених	Коректне додавання та видалення	+

6.2.7 Вікно з розширеним пошуком

В таблиці 6.7 наведено тестування вікна з розширеним пошуком.

Таблиця 6.7 – Тестування вікна з розширеним пошуком

Назва	Результат	Успішність
Відображення рецептів	Коректне відображення	+
Перехід по сторінкам	Коректний перехід	+
Відкриття рецептів	Коректне відкриття	+
Пошук за назвою	Коректний пошук	+
Додавання та видалення з улюблених	Коректне додавання та видалення	+
Фільтрування за типом	Коректне фільтрування	+
Фільтрування за продуктом	Коректне фільтрування	+
Сортування за датою	Коректне сортування	+

Продовження таблиці 6.7

Сортування популярністю	за	Коректне сортування	+
----------------------------	----	---------------------	---

6.2.8 Вікно «Калькулятор калорій»

В таблиці 6.8 наведено тестування вікна «Калькулятор калорій».

Таблиця 6.8 – Тестування вікна «Калькулятор калорій»

Назва	Результат	Успішність
Можливість обрати продукт	Коректне	+
Введення некоректної кількості продукту	Коректний	+
Введення коректної кількості продукту	Коректне	+
Видалення продукту	Коректний	+
Розрахування кількості калорій	Коректне	+

ВИСНОВКИ

Під час виконання дипломної роботи було використано: мову програмування C#, середовище розробки Microsoft Visual Studio 2019 з графічною підсистемою WPF, СКБД Microsoft SQL Server та клієнт-серверну архітектуру. Було проведено аналіз предметної області, описано існуючі методи вирішення завдання та методи шифрування, порівняно та визначено переваги та недоліки аналогів, а саме: 1000.menu, Smachno.ua та Allrecipes.

Основним завданням дипломної роботи стала розробка програмного рішення «Respie», що дозволило закріпити теоретичні знання і здобути практичні навички розробки програмного продукту з використанням методів шифрування. В першому розділі було проведено аналіз предметної області та обрано прототип проєкту. В другому розділі було обрано програмні засоби, що були використані в програмному рішенні. В третьому розділі проведено опис реалізації програми, наведені використані класи, їх поля та методи, описано інтерфейс та створену БД. В четвертому та п'ятому було надано інструкції щодо використання програмного продукту. В шостому розділі описану тестування та надано чек-лист тестів.

В створеному проєкті було реалізовано механізм роботи з рецептами, розроблено різні операції з ними, а саме: створення, зберігання, видалення, додавання до улюблених, коментування рецептів. Також в програмі створено дружній користувацький інтерфейс, який характеризується зрозумілістю та багатофункціональністю, і не потребує специфічних навичок у користувачів. Для забезпечення високого рівня надійності та безпека виконано шифрування паролів користувачів за допомогою хеш-функції.

Отже, робота над дипломною роботою дозволила на практиці отримати необхідні навички та знання для розробки програмного забезпечення з дотриманням всіх вимог та потреб користувачів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Остроушко О. Часова семантика минулого в сучасному кулінарному дискурсі / Оксана Остроушко // Філологічні студії: Науковий вісник Криворізького державного педагогічного університету. – 2019. – №20. – С. 73-86.
2. Антоненко О. В. Криптографічні методи перетворення інформації / О. В. Антоненко. – Бердянськ: БДПУ, 2015. – 180 с.
3. Корченко О. Г. Прикладна криптологія : системи шифрування / О. Г. Корченко, В. П. Сіденко, Ю. О. Дрейс. – Житомир: Державний університет телекомунікацій (ДУТ), 2014. – 448 с.
4. Website [Electronic resource]. – Access mode: <https://www.techopedia.com/definition/5411/website>.
5. Мобільні застосунки: їх види та особливості [Електронний ресурс]. – Режим доступу: <https://highload.today/uk/mobilni-zastosunki-yih-vidi-ta-osoblivosti/>.
6. Десктоп застосунки [Електронний ресурс]. – Режим доступу: <https://noname.digital/services/desktop-applications/>.
7. Becoming Allrecipes [Electronic resource]. – Access mode: <https://headerbidding.co/becoming-allrecipes/>.
8. Allrecipes [Electronic resource]. – Access mode: <https://www.allrecipes.com/>.
9. Про проєкт Smachno.ua [Електронний ресурс]. – Режим доступу: <https://smachno.ua/ua/o-proekte/>.
10. Smachno.ua [Електронний ресурс]. – Режим доступу: <https://smachno.ua/ua/>.
11. Домашні рецепти приготування страв з фото [Електронний ресурс]. – Режим доступу: <https://1000.menu/>.

12. Шифрування: Типи та алгоритми. Що це і який тип шифрування кращий? [Електронний ресурс]. – Режим доступу: <https://hostkoss.com/b/uk/encryption-types-algorithms/>.
13. Тихий Н.С. Захист інформації на основі криптографічних хеш-функцій. У кн.: Aktuální vymoženosti vědy -2018: тези доп. XIV-ї міжнар. наук.-практ. конф., 22-30 червня 2018 р., Praha / «Education and Science». – Praha: Zdenák Černák 2018. – 68 с.
14. Криптографічний хеш [Електронний ресурс]. – Режим доступу: https://ukrayinska.libretexts.org/Криптографічний_хеш.
15. What Is A Programming Language? [Electronic resource]. – Access mode: <https://codeinstitute.net/global/blog/what-is-a-programming-language/>.
16. TIOBE Programming Community Index Definition [Electronic resource]. – Access mode: https://www.tiobe.com/tiobe-index/programminglanguages_definition/
17. TIOBE Index for December 2023 [Electronic resource]. – Access mode: <https://www.tiobe.com/tiobe-index/>.
18. The C# Programming Language [Electronic resource]. – Access mode: <https://www.tiobe.com/tiobe-index/csharp/>.
19. Programming, scripting, and markup languages [Electronic resource]. – Access mode: – Режим доступу: <https://survey.stackoverflow.co/2023/#section-admired-and-desired-programming-scripting-and-markup-languages>.
20. The state of the Octoverse 2023 [Electronic resource]. – Access mode: <https://github.blog/2023-11-08-the-state-of-open-source-and-ai/#the-most-popular-programming-languages>
21. С #: що це за мова та де її використовують [Електронний ресурс]. – Режим доступу: <https://robotdreams.cc/uk/blog/284-s-hto-eto-za-yazyk-i-gde-ego-ispolzuuyut>.
22. Коноваленко І. В. Платформа .NET та мова програмування С# 8.0: навчальний посібник / І. В. Коноваленко, П. О. Марущак. – Тернопіль: ФОП Паляниця В. А., 2020. – 320 с.

23. Історія виникнення [Електронний ресурс]. – Режим доступу: <https://programming-languages8.webnode.com.ua/l/tse-stattya-z-foto2/>.
24. 9 причин вивчити мову С# [Електронний ресурс]. – Режим доступу: <https://foxminded.ua/9-prichin-vivchiti-movu-c/>.
25. Integrated development environment (IDE) [Electronic resource]. – Access mode: <https://www.techtarget.com/searchsoftwarequality/definition/>.
26. С# [Електронний ресурс]. – Режим доступу: <https://www.jetbrains.com/idea/devecosystem-2023/csharp/>.
27. Neel K. C# 5.0 and the .NET 4.5 Platform for Professionals / Christian Neel. – М: Williams, 2013. – 1440 p.
28. Benjamin Perkins C# 6 Programming with Visual Studio 2015 / Benjamin Perkins, Jacob Vibe Hammer, Jon D. Reid – Indianapolis: John Wiley & Sons, 2016. – 840p.
29. How to Use Windows Presentation Foundation [Electronic resource]. – Access mode: <https://reintech.io/blog/how-to-use-windows-presentation-foundation>.
30. Вища освіта – студентська наука – сучасне суспільство: напрями розвитку // матеріали V Всеукр. студ наук.-практ. конф., Київ, 21 грудня 2021 р.: тези доповідей / [ред. кол.: Г.І. Калінічева (відпов. ред.) та ін.]. – К.: МНТУ, 2021. – С. 325.
31. Welcome to the Visual Studio IDE [Electronic resource]. – Access mode: <https://learn.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide>.
32. Толстохатко В. А. Бази даних: проектування та використання для обліку нерухомого майна / В. А. Толстохатко, О. Є. Поморцева, І. М. Патракеєв. – Харків: ХНУМГ, 2014. – 174 с.
33. Основні відомості про бази даних [Електронний ресурс]. – Режим доступу: <https://support.microsoft.com/uk-ua/office/основні-відомості-про-бази-даних>.
34. Microsoft SQL Server [Electronic resource]. – Access mode: <https://www.microsoft.com/en-us/sql-server/sql-server-2019>.

35. SQL Server [Електронний ресурс]. – Режим доступу: https://powerautomate.microsoft.com/uk-ua/connectors/details/shared_sql/.
36. Jeff Prosise Programming Microsoft .NET / Jeff Prosise – Microsoft Press Redmond: Washington, 2002. – 800 p.
37. Гученко М. І. Методичні вказівки щодо виконання лабораторних робіт з навчальної дисципліни «Керування розробкою програмних систем» / М. І. Гученко, П. П. Костенко, Н. Л. Сохін. – Кременчук: КрНУ ім. Михайла Остроградського, 2018. – 26 с.
38. Ладієва Л. Р. Компоненти програмної інженерії Архітектура програмного забезпечення: навчальний посібник / Л. Р. Ладієва. – Київ: КПІ ім. Ігоря Сікорського, 2023. – 88 с.
39. Що таке алгоритм SHA 256? Функції та програми [Електронний ресурс]. – Режим доступу: <https://www.upgrad.com/blog/sha-256-algorithm/>.
40. Архітектура програмного забезпечення [Електронний ресурс]. – Режим доступу: <https://wezom.com.ua/ua/blog/arhitektura-programmnogo-obespecheniya>
41. Найважливіші архітектурні шаблони, які необхідно знати [Електронний ресурс]. – Режим доступу: <https://devzone.org.ua/post/nayvazhlivishi-arkhitekturni-shabloni>.
42. Richards M. Software Architecture Patterns / M. Richards. – Sebastopol, CA: O'Reilly Media, 2015. – 47 с.
43. Ланська С. С. Конспект лекцій з дисципліни «Бази даних» / С. С. Ланська. – Дніпро: ДНУ ім. Олеся Гончара, 2017. – 356 с.
44. Психологія кольорів у веб-дизайні: як вибрати кольірну палітру, щоб підвищити конверсію сайтів [Електронний ресурс]. – Режим доступу: <https://genius.space/lab/psihologiya-koloriv-u-veb-dizajni>.
45. 7 правил створення гарних інтерфейсів (Частина 1) [Електронний ресурс]. – Режим доступу: <https://medium.com/uxuiua/7-правил-створення-гарних-інтерфейсів>.

ДОДАТОК А
Технічне завдання

A.1 Вступ

Без комп'ютерних технологій не можливо уявити сьогодення. Зі впровадженням різних інновацій та удосконалень в інформаційній системі, все більшої уваги приділяється дослідженню всіх сфер життя людини.

У сучасному світі, де цифрова безпека стає важливим аспектом нашого онлайн-життя, розробка програмного забезпечення для ефективного та безпечного управління особистими даними стає нагальною задачею. Кулінарні знання та гастрономічні досягнення займають особливе місце в повсякденному житті, тому все більше людей шукають способи організації та зберігання своїх улюблених кулінарних рецептів.

A.1.1 Назва програми

Назва програми: «Respie».

A.1.2 Призначення та область застосування

Застосунок з кулінарними рецептами призначений для всіх, хто цікавиться готуванням, експериментує зі смаками та прагне до організованого зберігання та обміну своїми улюбленими рецептами. Він знаходить своє застосування як серед досвідчених кулінарів, так і серед новачків у готуванні, надаючи їм зручний інструмент для кулінарного творчості та вдосконалення навичок.

Область застосування: приватні особи.

A.2 Підстави для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Дослідження та програмна реалізація клієнт-серверної

системи кулінарних рецептів з безпечним зберіганням паролів користувачів», затверджене наказом Національного університету «Запорізька політехніка» № 168 від 24 квітня 2024 р.

А.3 Призначення розробки

Розробка застосунку з кулінарними рецептами має на меті створення функціонального та інтуїтивно зрозумілого інструменту, який надасть користувачам зручність, організованість та доступність в управлінні своїми кулінарними рецептами. Основне призначення розробки полягає у наступному:

- збереження та категоризація рецептів. Застосунок надає можливість користувачам зберігати свої улюблені кулінарні рецепти в базі даних. Користувачі зможуть організувати рецепти за типами, популярністю та за датою, що сприятиме зручному пошуку та швидкому доступу до необхідної інформації;
- пошук та рекомендації нових рецептів. Застосунок надає можливість користувачам відкривати нові кулінарні ідеї та рецепти;
- забезпечення безпеки. Застосунок шифрує паролі користувачів які зберігає;
- обмін та спільнота. Розроблений застосунок сприяє обміну рецептами та ідеями між користувачами. Користувачі можуть у коментарях ділитися своїми враженнями від рецептами зі спільнотою, відкривати нові кулінарні тенденції та отримувати поради стосовно приготування страв.

А.4 Вимоги до програми

А.4.1 Вимоги до функціональних характеристик

Вимоги до функціональних характеристик для застосунку з кулінарними рецептами:

- реєстрація та авторизація користувачів. Застосунок повинен мати можливість реєстрації нових користувачів і авторизації існуючих. Це дозволить зберігати та відновлювати персональні дані, а також забезпечить доступ до індивідуальних налаштувань;
- шифрування паролів. Застосунок повинен шифрувати збережені паролі від акаунтів користувачів, за для забезпечення безпеки облікових записів;
- збереження та організація рецептів. Застосунок повинен надавати можливість додавати, редагувати, видаляти рецепти та додавати до улюблених;
- пошук та фільтрація рецептів. Застосунок повинен мати можливість швидкого та ефективного пошуку рецептів залежно від різних параметрів, таких як назва, інгредієнти, тип страви. Користувачі повинні мати можливість сортувати рецепти за популярністю та датою;
- рецепти з кроками та інгредієнтами. Застосунок повинен дозволяти додавати докладні кроки готування для кожного рецепту, а також перелік необхідних інгредієнтів з відповідними кількостями. Це допоможе користувачам зрозуміти процес готування та виконати його успішно;
- відображення характеристик рецептів. Застосунок повинен надавати користувачам інформацію про час приготування, кількість порцій та інші характеристики рецепту. Це дозволить користувачам швидко оцінити відповідність рецепту їхнім потребам та обмеженням;
- обмін рецептами та спільнота. Користувачі повинні мати можливість ділитися своїми рецептами, залишати коментарі та відгуки;
- калькулятор калорій. Застосунок повинен давати можливість користувачам розраховувати калорійність страви.

А.4.2 Вимоги до надійності

Вимоги до надійності для застосунку з кулінарними рецептами:

- стабільність та безперебійна робота. Застосунок повинен функціонувати стабільно та безперебійно, забезпечуючи користувачам постійний доступ до їхніх кулінарних рецептів;
- захист даних та конфіденційність. Застосунок повинен гарантувати захист особистої інформації користувачів;
- стійкість до навантаження. Застосунок повинен бути здатним витримувати достатню кількість користувачів;
- тестування та усунення помилок. Застосунок повинен бути підданий тестуванню з метою виявлення та виправлення можливих помилок.

A.4.3 Вимоги до технічних та програмних засобів

Для адекватної роботи з програмою, потрібно щоб пристрій, який буде використовуватися, задовольняв таким програмним та технічним вимогам:

- потрібна оперативна пам'ять: мінімум 4 Гб;
- потрібна місце на диску: мінімум 400 Мб;
- операційна система: Windows 10;
- жорсткий диск об'ємом мінімум 128 Гб.

A.5 Вимоги до програмної документації

Склад програмної документації повинен включати в себе:

- технічне завдання розроблене відповідно до аналізу потреб користувачів та предметної області;
- опис програми;
- пояснювальна записка.

A.6 Стадії і етапи розробки

A.6.1 Стадії розробки

Стадії розробки:

- аналіз вимог та формування концепції;
- розробка технічного завдання;
- проектування програмного забезпечення;
- розробки програмного забезпечення;
- тестування програмного забезпечення;
- завершення розробки та оформлення програмної документації.

A.6.2 Етапи розробки та зміст роботи на кожному етапі

Етапи розробки:

- аналіз предметної області – огляд актуальності теми розробки, області застосування, основних понять;
- аналіз існуючих аналогів – огляд існуючих методів вирішення проблем в предметній області, їх порівняння;
- аналіз існуючих методів шифрування паролів;
- визначення вимог до програмного забезпечення – затвердження функцій які повинно виконувати програмне забезпечення;
- визначення вимог до надійності – затвердження необхідного рівня надійності програмного забезпечення;
- визначення вимог до технічних та програмних засобів – затвердження мінімальних вимог для стабільного використання програмного забезпечення у системі;
- визначення вимог до програмної документації – затвердження складу програмної документації;
- проектування структури клієнтської частини – створення прототипу інтерфейсу;

- проектування структури серверної частини – визначення основних алгоритмів, які буде виконувати ця частина;
- проектування структури таблиць бази даних – визначення основних сутностей;
- розробка клієнтської частини – створення форм інтерфейсу та їх налаштування;
- розробка хоста – створення хост частини;
- розробка серверної частини – розробка основних алгоритмів програмного застосунку;
- розробка таблиці бази даних та їх наповнення – створення таблиць для усіх сутностей та заповнення їх необхідними даними;
- об'єднання частин проєкту – налаштування з'єднання усіх розроблених частин між собою;
- тестування програмного забезпечення – перевірка коректності виконання всіх затверджених функцій та відповідності до вимог надійності;
- виправлення знайдених проблем – виправлення помилок зафіксованих під час тестування;
- оформлення опису програми – опис загальних відомостей, логічної структури, алгоритмів, використаних технічних засобів, початку роботи з програмним застосунком та вхідних та вихідних даних системи;
- завершення розробки пояснювальної записки – оформлення усіх пунктів пояснювальної записки, додавання до неї необхідних додатків.

ДОДАТОК Б
Опис програми

Б.1 Загальні відомості

Застосунок «Ресріє» є інноваційним та простим у використанні рішенням для всіх любителів кулінарії. Він створений з метою надати користувачам зручний доступ до різноманітних кулінарних рецептів та сприяти їхньому творчому процесу приготування страв.

Мета використання застосунку «Ресріє» полягає в тому, щоб забезпечити користувачам легкий та швидкий пошук рецептів, додавання власних рецептів, організацію та збереження улюблених рецептів, а також спільноту, де можна ділитися своїми кулінарними витворами та отримувати відгуки та поради від інших користувачів.

Застосунок «Ресріє» адресований широкому колу користувачів – від початківців-кухарів до досвідчених гурманів, які цінують смачну та різноманітну їжу. Він стане незамінним помічником у плануванні страв на кожен день, особливі події та святкові заходи.

Б.2 Функціональне призначення

Функції, що виконує програма:

- реєстрація та авторизація користувачів. Можливість реєстрації нових користувачів і авторизації існуючих;
- шифрування паролів. Можливість шифрувати збережені паролі від акаунтів користувачів, за для забезпечення безпеки облікових записів;
- збереження та організація рецептів. Можливість додавати, редагувати, видаляти рецепти та додавати до улюблених;
- пошук та фільтрація рецептів. Можливість швидкого та ефективного пошуку рецептів залежно від різних параметрів, таких як назва, інгредієнти, тип страви. Можливість сортувати рецепти за популярністю та датою;

- рецепти з кроками та інгредієнтами. Додавання до рецепту докладні кроки готування для кожного рецепту, а також перелік необхідних інгредієнтів з відповідними кількостями;
- відображення характеристик рецептів. Надання інформації про час приготування, кількість порцій та інші характеристики рецепту;
- обмін рецептами та спільнота. Можливість ділитися своїми рецептами, залишати коментарі та відгуки;
- калькулятор калорій. Можливість розраховувати калорійність страви.

Б.3 Опис логічної структури програми

Логічна структура ПЗ показна за допомогою структурної схеми на рисунку Б.1

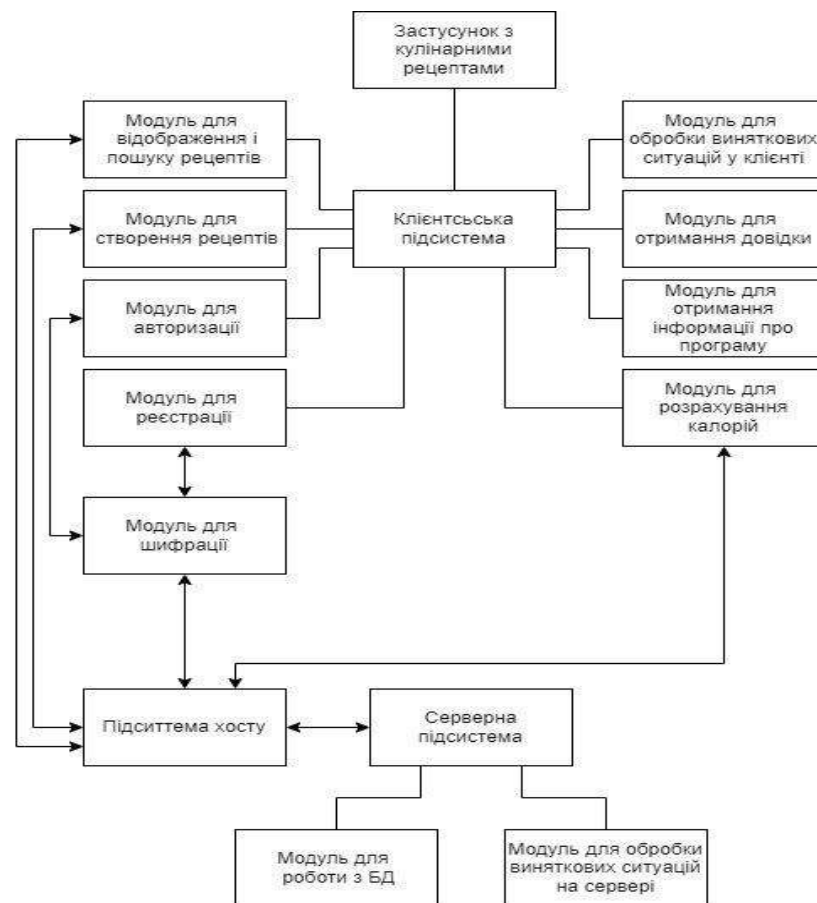


Рисунок Б.1 – Структурна схема проекту

Б.3.1 Алгоритм реєстрації

Призначення: цей алгоритм створено для того, щоб реалізувати можливість створювати нові акаунти.

Б.3.2 Алгоритм авторизації

Призначення: цей алгоритм створено для того, щоб користувачі мали можливість увійти в систему під раніше створеним акаунтом, для подальшої роботи.

Б.3.3 Алгоритм «Розрахування калорійності»

Призначення: цей алгоритм створено для того, щоб користувачі мали можливість підрахувати калорійність обраних до приготування страв за допомогою калькулятора калорій.

Б.3.4 Алгоритм сортування за популярністю

Призначення: цей алгоритм створено для того, щоб користувачі мали можливість побачити найпопулярніші рецепти.

Б.3.5 Алгоритм сортування за датою

Призначення: цей алгоритм створено для того, щоб користувачі мали можливість побачити нові та старі рецепти в залежності від обраного сортування.

Б.3.6 Алгоритм пошуку за продуктами

Призначення: цей алгоритм створено для того, щоб користувачі мали можливість відфільтрувати перелік рецептів за інгредієнтами, щоб мати можливість приготувати страви з наявними в них продуктами.

Б.3.7 Алгоритм пошуку за назвою

Призначення: цей алгоритм створено для того, щоб користувачі мали можливість знайти рецепт відповідно до його назви за ключовими словами.

Б.3.8 Алгоритм «Коментування рецепту»

Призначення: цей алгоритм створено для того, щоб користувачі мали можливість коментувати рецепти, щоб мати можливість поділитися своїми емоціями та отримати зворотній зв'язок.

Б.3.9 Алгоритм «Редагування рецепту»

Призначення: цей алгоритм створено для того, щоб користувачі мали можливість змінити опис створених своїх рецептів.

На рисунку 3.10 зображено блок-схема алгоритму.

Б.3.10 Алгоритм «Видалення рецепту»

Призначення: цей алгоритм створено для того, щоб користувачі мали можливість видаляти не потрібні власні рецепти, які раніше вони створювали.

Б.3.11 Алгоритм «Створення рецепту»

Призначення: цей алгоритм створено для того, щоб користувачі мали створювати власні рецепти зі збереженням їх.

Б.3.12 Алгоритм «Додавання рецепту до улюблених»

Призначення: цей алгоритм створено для того, щоб користувачі мали можливість додавати рецепти до улюбленого, щоб мати до них швидкий доступ.

Б.3.13 Алгоритм «Видалення рецепту з улюблених»

Призначення: цей алгоритм створено для того, щоб користувачі мали можливість видаляти не потрібні рецепти з «улюблених».

Б.3.14 Алгоритм «Сортування за популярністю»

Призначення: цей алгоритм створено для того, щоб користувачі мали можливість побачити популярні і непопулярні рецепти в залежності від обраного сортування.

Б.3.2 Алгоритм шифрації

Призначення: цей алгоритм створено алгоритм створено для того, щоб шифрувати паролі які вводять користувачі за допомоги додавання до паролю солі та використання SHA-256 алгоритму.

Б.4 Використані технічні засоби

Під час виконання дипломної роботи було використано мову програмування C#, середовище розробки Microsoft Visual Studio 2019 з графічною підсистемою WPF, СКБД Microsoft SQL Server та клієнт-серверну архітектуру. Саме через велику кількість переваги та популярність у використанні під час розроблення десктопних застосунків було обрано ці програмні засоби.

Умови застосування програми для клієнтської та серверної частини. Для адекватної роботи з програмою, потрібно щоб пристрій, який буде використовуватися, задовольняв таким програмним та технічним вимогам:

- потрібна оперативна пам'ять: мінімум 4 Гб;
- потрібна місце на диску: мінімум 400 Мб;
- операційна система: Windows 10;
- жорсткий диск об'ємом мінімум 128 Гб.

Для компіляції та внесення змін до програми рекомендується середовище Visual Studio 2019 та платформа .NET Framework 4.7.2.

Б.5 Виклик і завантаження

Програма запускається за допомогою виконуваного файлу RespieClient.exe та здійснюється подвійним натисканням лівої клавіші комп'ютерної миші по ньому або його ярлику.

Б.6 Початкові дані

Програма отримує такі вхідні дані як:

- сигнал про натискання клавіш миші та клавіатури;
- індекс обраного користувачем елементу списку;

- інформація про обрані прапорці продуктів та при додаванні до улюблених;

- БД;

- зображення іконки програми, продуктів, зображення рецептів та фонів інтерфейсу.

Програма видає такі вихідні дані як:

- інформація про рецепти;

- спливаючі підказки;

- інформація про програму та розробника;

- інформація про розраховані калорії страв;

- інформація про основний функціонал ПЗ.

ДОДАТОК В
Текст програми

В.1 Файл RecipeHashing.cs

```

using System;
using System.Collections.Generic;
using System.Text;

namespace Recipe
{
    class RecipeHashing
    {
        // Генерація випадкової солі
        public static string GenerateSalt()
        {
            // Генеруємо випадковий рядок для солі
            string chars = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890";
            StringBuilder salt = new StringBuilder();
            Random random = new Random();
            int saltLength = 16; // Довжина солі
            for (int i = 0; i < saltLength; i++)
            {
                int index = random.Next(chars.Length);
                salt.Append(chars[index]);
            }
            return salt.ToString();
        }

        // Хешування паролю з використанням солі
        public static string HashPassword(string password, string salt)
        {
            // Переводимо в байти
            byte[] saltBytes = Encoding.UTF8.GetBytes(salt);
            byte[] passwordBytes = Encoding.UTF8.GetBytes(password);

            // Додаємо сіль до паролю
            byte[] saltedPassword = new byte[saltBytes.Length + passwordBytes.Length];
            Buffer.BlockCopy(saltBytes, 0, saltedPassword, 0, saltBytes.Length);
            Buffer.BlockCopy(passwordBytes, 0, saltedPassword, saltBytes.Length, passwordBytes.Length);

            // Обчислюємо хеш
            string hash = SHA256Hash.ComputeSHA256(BitConverter.ToString(saltedPassword).Replace("-", ""));
            return hash.ToLower();
        }

        // Повертаємо хеш
        return hash;
    }
}

// Реалізація SHA-256
public static class SHA256Hash
{
    public static string ComputeSHA256(string input)
    {
        // Ініціалізуємо початкові значення хеша
        uint[] h = new uint[]

```

```

{
    0x6a09e667, 0xbb67ae85, 0x3c6ef372, 0xa54ff53a,
    0x510e527f, 0x9b05688c, 0x1f83d9ab, 0x5be0cd19
};

// Перетворюємо рядок в байти UTF-8
byte[] bytes = Encoding.UTF8.GetBytes(input);

// Додаємо довжину повідомлення в бітах
ulong bitLength = (ulong)bytes.Length * 8;
List<byte> paddedBytes = new List<byte>(bytes);
paddedBytes.Add(0x80); // Додаємо один біт '1' і решту нулів
while ((paddedBytes.Count * 8) % 512 != 448)
{
    paddedBytes.Add(0x00);
}
paddedBytes.AddRange(BitConverter.GetBytes(bitLength));

// Розбиваємо дані на блоки по 512 біт
int blockCount = paddedBytes.Count / 64;
for (int i = 0; i < blockCount; i++)
{
    byte[] block = new byte[64];
    Array.Copy(paddedBytes.ToArray(), i * 64, block, 0, 64);

    // Розширюємо блоки в слова (32-бітні)
    uint[] words = new uint[64];
    for (int j = 0; j < 16; j++)
    {
        words[j] = BitConverter.ToUInt32(block, j * 4);
    }
    for (int j = 16; j < 64; j++)
    {
        uint s0 = RotR(words[j - 15], 7) ^ RotR(words[j - 15],
18) ^ (words[j - 15] >> 3);
        uint s1 = RotR(words[j - 2], 17) ^ RotR(words[j - 2], 19)
^ (words[j - 2] >> 10);
        words[j] = words[j - 16] + s0 + words[j - 7] + s1;
    }

    // Ініціалізуємо тимчасові змінні
    uint a = h[0];
    uint b = h[1];
    uint c = h[2];
    uint d = h[3];
    uint e = h[4];
    uint f = h[5];
    uint g = h[6];
    uint hVal = h[7];

    // Виконуємо основні операції SHA-256
    for (int j = 0; j < 64; j++)
    {
        uint temp1 = hVal + Sum1(e) + Ch(e, f, g) + K[j] +
words[j];

        uint temp2 = Sum0(a) + Maj(a, b, c);
        hVal = g;
        g = f;
        f = e;
        e = d + temp1;
        d = c;
        c = b;
    }
}

```

```

        b = a;
        a = temp1 + temp2;
    }

    // Оновлюємо значення хеша
    h[0] += a;
    h[1] += b;
    h[2] += c;
    h[3] += d;
    h[4] += e;
    h[5] += f;
    h[6] += g;
    h[7] += hVal;
}

// Перетворюємо 32-бітні слова хеша в байти і формуємо
результуючий хеш
byte[] hashBytes = new byte[32];
for (int i = 0; i < 8; i++)
{
    BitConverter.GetBytes(h[i]).CopyTo(hashBytes, i * 4);
}

// Перетворюємо байти хеша в рядок в шістнадцятковому форматі
StringBuilder hashString = new StringBuilder();
foreach (byte b in hashBytes)
{
    hashString.Append(b.ToString("x2"));
}

return hashString.ToString();
}

private static uint RotR(uint x, int n)
{
    return (x >> n) | (x << (32 - n));
}

private static uint Sum0(uint x)
{
    return RotR(x, 2) ^ RotR(x, 13) ^ RotR(x, 22);
}

private static uint Sum1(uint x)
{
    return RotR(x, 6) ^ RotR(x, 11) ^ RotR(x, 25);
}

private static uint Ch(uint x, uint y, uint z)
{
    return (x & y) ^ (~x & z);
}

private static uint Maj(uint x, uint y, uint z)
{
    return (x & y) ^ (x & z) ^ (y & z);
}

private static readonly uint[] K = new uint[]
{
    0x428a2f98, 0x71374491, 0xb5c0fbcf, 0xe9b5dba5,
    0x3956c25b, 0x59f111f1, 0x923f82a4, 0xab1c5ed5,

```

```

0xd807aa98, 0x12835b01, 0x243185be, 0x550c7dc3,
0x72be5d74, 0x80deb1fe, 0x9bdc06a7, 0xc19bf174,
0xe49b69c1, 0xefbe4786, 0x0fc19dc6, 0x240ca1cc,
0x2de92c6f, 0x4a7484aa, 0x5cb0a9dc, 0x76f988da,
0x983e5152, 0xa831c66d, 0xb00327c8, 0xbf597fc7,
0xc6e00bf3, 0xd5a79147, 0x06ca6351, 0x14292967,
0x27b70a85, 0x2e1b2138, 0x4d2c6dfc, 0x53380d13,
0x650a7354, 0x766a0abb, 0x81c2c92e, 0x92722c85,
0xa2bfe8a1, 0xa81a664b, 0xc24b8b70, 0xc76c51a3,
0xd192e819, 0xd6990624, 0xf40e3585, 0x106aa070,
0x19a4c116, 0x1e376c08, 0x2748774c, 0x34b0bcb5,
0x391c0cb3, 0x4ed8aa4a, 0x5b9cca4f, 0x682e6ff3,
0x748f82ee, 0x78a5636f, 0x84c87814, 0x8cc70208,
0x90beffffa, 0xa4506ceb, 0xbef9a3f7, 0xc67178f2
};
}
}

```

B.2 Файл RecpieDataBase.cs

```

using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;

namespace Recpie
{
    class RecpieDataBase
    {
        private string connectionString = @"Data
Source=(LocalDB)\MSSQLLocalDB;" +

@"AttachDbFilename=C:\Users\Мария\Desktop\студент(\Диплом\Recpie\Recpie\DataB
ase.mdf;" +

                                @"Integrated Security=True"; //
        рядок підключення до БД

        public bool AddServerUser(string Login, string Password) // додати
        акаунт
        {
            try
            {

                string Salt = RecipeHashing.GenerateSalt(); //
                генеруємо сіль
                Password = RecipeHashing.HashPassword(Password, Salt); //
                хешуємо пароль з сіллю

                // робимо запит до бд
                SqlConnection Connection = new
                SqlConnection(connectionString);
                SqlCommand Command = new SqlCommand("INSERT dbo.ServerUser
(Login, Password, Favorite, Salt) VALUES ('" + Login + "', '" + Password +
"'', '" + "" + "', '" + Salt + "')", Connection);
                Connection.Open();
                Command.ExecuteNonQuery();
            }
            catch { }
        }
    }
}

```

```

        Connection.Close();
        return true;
    }
    catch
    {
        return false;
        //Exceptions e = new Exceptions();
        //e.ErrorMessage(3);
    }
}
public bool AddRecipe(RecipeObject recipe)
{
    try
    {
        using (SqlConnection Connection = new
SqlConnection(connectionString))
        {
            recipe.Comments = "";
            recipe.Rating = 0;
            Connection.Open();
            SqlCommand Command = new SqlCommand();
            Command.Connection = Connection;
            Command.CommandText = @"INSERT INTO dbo.Recipe VALUES
(@User_ID, @Content, N'" + recipe.Name + "', N'" + recipe.Type + "', N'" +
recipe.Description + "', N'" + recipe.Products + "', N'" +
recipe.Recipe_procedure + "', N'" + recipe.Number_of_servings + "', N'" +
recipe.Time + "', N'" + recipe.Date_of_create + "', N'" + recipe.Comments +
"', N'" + recipe.Rating + "')";
            Command.Parameters.Add("@User_ID", SqlDbType.Int);
            Command.Parameters.Add("@Content", SqlDbType.Image,
1000000);
            Command.Parameters["@User_ID"].Value = recipe.User_ID;
            Command.Parameters["@Content"].Value = recipe.Content;
            Command.ExecuteNonQuery();
            return true;
        }
    }
    catch
    {
        return false;
    }
}

public bool LoginExists(string Login) // дізнатися про існування
логіну (існує - true, не існує - false)
{
    try
    {
        // робимо запит до бд
        SqlDataAdapter Adapter = new SqlDataAdapter();
        DataTable Table = new DataTable();
        SqlConnection Connection = new
SqlConnection(connectionString);
        SqlCommand Command = new SqlCommand("SELECT ID FROM
dbo.ServerUser WHERE Login = '" + Login + "'", Connection);
        Adapter.SelectCommand = Command;
        Adapter.Fill(Table);
    }
}

```

```

        if (Table.Rows.Count == 0) // якщо таких логінів немає у
таблиці
            return false;
        else
            return true;
    }
    catch
    {
        //Exceptions e = new Exceptions();
        //e.ErrorMessage(2);
        return false;
    }
}
public int AccountExists(string Login, string Password) // чи є
акаунт з таким логіном та паролем
{
    try
    {
        // робимо запит на отримання солі для цього логіну
        SqlDataAdapter Adapter = new SqlDataAdapter();
        DataTable Table = new DataTable();
        SqlConnection Connection = new
SqlConnection(connectionString);
        SqlCommand Command = new SqlCommand("SELECT ALL * FROM
dbo.ServerUser WHERE Login = '" + Login + "';", Connection);
        Adapter.SelectCommand = Command;
        Adapter.Fill(Table);
        if (Table.Rows.Count == 0) // якщо такого логіну немає
            return 0;
        else
        {
            string Salt = Table.Rows[0].Field<string>("Salt");
            Password = RecipeHashing.HashPassword(Password, Salt); //
хешуємо пароль з сіллю

            // робимо запит до бд
            Adapter = new SqlDataAdapter();
            Table = new DataTable();
            Connection = new SqlConnection(connectionString);
            Command = new SqlCommand("SELECT ALL * FROM
dbo.ServerUser WHERE Login = '" + Login + "' AND Password = '" + Password +
"'; ", Connection);
            Adapter.SelectCommand = Command;
            Adapter.Fill(Table);
            if (Table.Rows.Count == 0) // якщо такого логіну з
паролем немає
                return 0;
            else
            {
                return Table.Rows[0].Field<int>("ID");
            }
        }
    }
    catch
    {
        //Exceptions e = new Exceptions();
        //e.ErrorMessage(2);

```

```

        return 0;
    }
}

public string GetLoginByID(int ID)
{
    // робимо запит до бд
    SqlDataAdapter Adapter = new SqlDataAdapter();
    DataTable Table = new DataTable();
    SqlConnection Connection = new SqlConnection(connectionString);
    SqlCommand Command = new SqlCommand("SELECT ALL * FROM
dbo.ServerUser WHERE ID = '" + ID + "';", Connection);
    Adapter.SelectCommand = Command;
    Adapter.Fill(Table);
    if (Table.Rows.Count == 0)
        return null;
    else
    {
        return Table.Rows[0].Field<string>("Login");
    }
}

public List<RecipeObject> GetAllRecipe()
{
    try
    {
        SqlDataAdapter Adapter = new SqlDataAdapter();
        DataTable Table = new DataTable();
        SqlConnection Connection = new
SqlConnection(connectionString);
        SqlCommand Command = new SqlCommand("SELECT ALL * FROM
dbo.Recipe;", Connection);
        Adapter.SelectCommand = Command;
        Adapter.Fill(Table);
        List<RecipeObject> recipeList = new List<RecipeObject>();
        for (int i = 0; i < Table.Rows.Count; i++)
        {
            RecipeObject recipe = new RecipeObject();
            recipe.ID = Table.Rows[i].Field<int>("ID");
            recipe.User_ID = Table.Rows[i].Field<int>("User_ID");
            recipe.Content = (byte[])Table.Rows[i][2];
            recipe.Name = Table.Rows[i].Field<string>("Name");
            recipe.Type = Table.Rows[i].Field<string>("Type");
            recipe.Description =
Table.Rows[i].Field<string>("Description");
            recipe.Products =
Table.Rows[i].Field<string>("Products");
            recipe.Recipe_procedure =
Table.Rows[i].Field<string>("Recipe_procedure");
            recipe.Number_of_servings =
Table.Rows[i].Field<string>("Number_of_servings");
            recipe.Time = Table.Rows[i].Field<string>("Time");
            recipe.Date_of_create =
Table.Rows[i].Field<DateTime>("Date_of_create");
            recipe.Comments =
Table.Rows[i].Field<string>("Comments");
            recipe.Rating = Table.Rows[i].Field<int>("Rating");
            recipeList.Add(recipe);
        }
    }
}

```

```

        }
        return recipeList;
    }
    catch
    {
        //Exceptions e = new Exceptions();
        //e.ErrorMessage(2);
        return null;
    }
}
public List<ProductObject> GetAllProductObject()
{
    try
    {
        SqlDataAdapter Adapter = new SqlDataAdapter();
        DataTable Table = new DataTable();
        SqlConnection Connection = new
SqlConnection(connectionString);
        SqlCommand Command = new SqlCommand("SELECT ALL * FROM
dbo.Product;", Connection);
        Adapter.SelectCommand = Command;
        Adapter.Fill(Table);
        List<ProductObject> productList = new List<ProductObject>();
        for (int i = 0; i < Table.Rows.Count; i++)
        {
            ProductObject product = new ProductObject();
            product.ID = Table.Rows[i].Field<int>("ID");
            product.Name = Table.Rows[i].Field<string>("Name");
            product.Calories =
Table.Rows[i].Field<double>("Calories");
            productList.Add(product);
        }
        return productList;
    }
    catch
    {
        //Exceptions e = new Exceptions();
        //e.ErrorMessage(2);
        return null;
    }
}
public string GetFavorite(int ID)
{
    // робимо запит до бд
    SqlDataAdapter Adapter = new SqlDataAdapter();
    DataTable Table = new DataTable();
    SqlConnection Connection = new SqlConnection(connectionString);
    SqlCommand Command = new SqlCommand("SELECT Favorite FROM
dbo.ServerUser WHERE ID = '" + ID + "';", Connection);
    Adapter.SelectCommand = Command;
    Adapter.Fill(Table);
    if (Table.Rows.Count == 0)
        return null;
    else
    {
        return Table.Rows[0].Field<string>("Favorite");
    }
}

```

```

    }

    public bool UpdateFavorite(int ID, string Favorite)
    {
        try
        {
            SqlConnection Connection = new
SqlConnection(connectionString);
            SqlCommand Command = new SqlCommand(@"UPDATE dbo.ServerUser
SET Favorite = '" + Favorite + "' " + "WHERE ID = '" + ID + "';",
Connection); ;
            // + recipe.Procedure + "', '" + recipe.Number_of_servings + '");",
Connection);

            Connection.Open();
            Command.ExecuteNonQuery();
            Connection.Close();
            return true;
        }
        catch
        {
            return false;
        }
    }

    public bool RecipeRatingUpdate(RecipeObject recipe)
    {
        try
        {
            SqlConnection Connection = new
SqlConnection(connectionString);
            SqlCommand Command = new SqlCommand(@"UPDATE dbo.Recipe SET
Rating = '" + recipe.Rating + "' " + "WHERE ID = '" + recipe.ID + "';",
Connection); ;
            // + recipe.Procedure + "', '" + recipe.Number_of_servings + '");",
Connection);

            Connection.Open();
            Command.ExecuteNonQuery();
            Connection.Close();
            return true;
        }
        catch
        {
            return false;
        }
    }

    public bool UpdateRecipe(RecipeObject recipe)
    {
        try
        {
            using (SqlConnection Connection = new
SqlConnection(connectionString))
            {
                Connection.Open();
                SqlCommand Command = new SqlCommand();
                Command.Connection = Connection;
                Command.CommandText = @"UPDATE dbo.Recipe SET Content =
@Content, Name = N'" + recipe.Name

```

```

+ "'", Type = N'" + recipe.Type
+ "'", Description = N'" + recipe.Description
+ "'", Products = N'" + recipe.Products
+ "'", Recipe_procedure = N'" + recipe.Recipe_procedure
+ "'", Number_of_servings = N'" + recipe.Number_of_servings
+ "'", Time = N'" + recipe.Time
+ "'", Comments = N'" + recipe.Comments
+ "' WHERE ID = '" + recipe.ID + "';";
        Command.Parameters.Add("@Content", SqlDbType.Image,
1000000);
        Command.Parameters["@Content"].Value = recipe.Content;
        Command.ExecuteNonQuery();
        return true;
    }
}
catch
{
    return false;
}
}

public bool RecipeDelete(RecipeObject recipe)
{
    try
    {
        SqlConnection Connection = new
SqlConnection(connectionString);
        SqlCommand Command = new SqlCommand(@"DELETE FROM dbo.Recipe
WHERE ID = '" + recipe.ID + "';", Connection); ;
        // + recipe.Procedure + "'", '" + recipe.Number_of_servings + "'",
Connection);
        Connection.Open();
        Command.ExecuteNonQuery();
        Connection.Close();
        return true;
    }
    catch
    {
        return false;
    }
}
}
}

```

В.3 Файл ResourceDictionary.xaml

```

<ResourceDictionary
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">

    <!-- Налаштування кнопки -->
    <Style TargetType="Button">
        <Setter Property="FontFamily" Value="Georgia" />
        <Setter Property="Background">
            <Setter.Value>
                <LinearGradientBrush StartPoint="0,0" EndPoint="1,1">
                    <GradientStop Color="#FFBD59" Offset="0"/>
                    <GradientStop Color="#ffa9f9" Offset="1"/>
                </LinearGradientBrush>
            </Setter.Value>
        </Setter>
        <Setter Property="Foreground" Value="#7f547c" />
        <Setter Property="FontSize" Value="18" />
        <Setter Property="FontWeight" Value="Bold" />

        <!-- Закруглення -->
        <Setter Property="Template">
            <Setter.Value>
                <ControlTemplate TargetType="Button">
                    <Border Background="{TemplateBinding Background}"
                        CornerRadius="15">
                        <Border.Effect>
                            <DropShadowEffect Color="#7f547c" Direction="270"
ShadowDepth="2" Opacity="0.7"/>
                        </Border.Effect>
                        <ContentPresenter HorizontalAlignment="Center"
VerticalAlignment="Center"/>
                    </Border>
                    <ControlTemplate.Triggers>
                        <Trigger Property="IsMouseOver" Value="True">
                            <Setter Property="Background" Value="#FFBD59"/>
                        </Trigger>
                    </ControlTemplate.Triggers>
                </ControlTemplate>
            </Setter.Value>
        </Setter>
    </Style>

    <!-- Налаштування піпису -->
    <Style TargetType="Label">
        <Setter Property="FontFamily" Value="Georgia" />
        <Setter Property="Foreground" Value="#7f547c" />
        <Setter Property="FontSize" Value="18" />
        <Setter Property="FontWeight" Value="Bold" />
        <Setter Property="HorizontalContentAlignment" Value="Center" />

        <Setter Property="Template">
            <Setter.Value>
                <ControlTemplate TargetType="Label">

```

```

        <TextBlock Text="{TemplateBinding Content}"
TextWrapping="Wrap" TextAlignment="Center"/>
    </ControlTemplate>
</Setter.Value>
</Setter>
</Style>

<!-- Налаштування текстових полів -->
<Style TargetType="TextBox">
    <Setter Property="Foreground" Value="#7f547c" />
    <Setter Property="FontSize" Value="18" />
    <Setter Property="FontWeight" Value="Bold" />
    <Setter Property="Background">
        <Setter.Value>
            <LinearGradientBrush StartPoint="0,0" EndPoint="1,1">
                <GradientStop Color="#fffacd" Offset="0"/>
                <GradientStop Color="#ffe6ff" Offset="1"/>
            </LinearGradientBrush>
        </Setter.Value>
    </Setter>
    <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="#7f547c" Direction="270"
ShadowDepth="2" Opacity="0.7"/>
                    </Border.Effect>
                    <ScrollViewer x:Name="PART_ContentHost" />
                </Border>
            </ControlTemplate>
        </Setter.Value>
    </Setter>
</Style>

<!-- Налаштування парольових полів -->
<Style TargetType="PasswordBox">

    <Setter Property="Foreground" Value="#7f547c" />
    <Setter Property="FontSize" Value="18" />
    <Setter Property="FontWeight" Value="Bold" />
    <Setter Property="Background">
        <Setter.Value>
            <LinearGradientBrush StartPoint="0,0" EndPoint="1,1">
                <GradientStop Color="#fffacd" Offset="0"/>
                <GradientStop Color="#ffe6ff" Offset="1"/>
            </LinearGradientBrush>
        </Setter.Value>
    </Setter>
    <Setter Property="BorderThickness" Value="0" />
    <!-- Закруглення -->
    <Setter Property="Template">

```



```

        <DropShadowEffect
Color="#7f547c" Direction="270" ShadowDepth="2" Opacity="0.7" />
        </Border.Effect>
    </Border>
    <Border Grid.Column="0" Margin="1"
BorderBrush="#7f547c" BorderThickness="0,0,0,0" CornerRadius="5,0,0,5">
        <Border.Background>
            <LinearGradientBrush
StartPoint="0,0" EndPoint="1,1">
                <GradientStop Offset="0"
Color="#FFBD59"/>
                <GradientStop Offset="1"
Color="#ffa9f9"/>
            </LinearGradientBrush>
        </Border.Background>
    </Border>
    <Path x:Name="Arrow" Grid.Column="1"
Fill="{TemplateBinding Foreground}" HorizontalAlignment="Center"
VerticalAlignment="Center" Data="M 0 0 L 4 4 L 8 0 Z"/>
    </Grid>
</ControlTemplate>
</ToggleButton.Template>
</ToggleButton>
    <ContentPresenter x:Name="ContentPresenter"
ContentTemplate="{TemplateBinding SelectionBoxItemTemplate}"
Content="{TemplateBinding SelectionBoxItem}" HorizontalAlignment="Left"
Margin="5, 0, 0, 0" VerticalAlignment="Center"/>
    <Popup x:Name="Popup" Placement="Bottom"
IsOpen="{TemplateBinding IsDropDownOpen}" AllowsTransparency="True"
Focusable="False" PopupAnimation="Slide">
        <Grid x:Name="DropDown"
SnapsToDevicePixels="True" MinWidth="{TemplateBinding ActualWidth}"
MaxHeight="{TemplateBinding MaxDropDownHeight}">
            <Border x:Name="DropDownBorder"
Background="#FFBD59" BorderThickness="1" BorderBrush="LightGray"
CornerRadius="5"/>
            <ScrollViewer Margin="4"
SnapsToDevicePixels="True">
                <ItemsPresenter />
            </ScrollViewer>
        </Grid>
    </Popup>
</Grid>
</ControlTemplate>
</Setter.Value>
</Setter>
</Style>

<!-- Налаштування листв'ю -->

<Style TargetType="ListView">
    <Setter Property="Foreground" Value="#7f547c" />
    <Setter Property="FontSize" Value="18" />
    <Setter Property="FontWeight" Value="Bold" />
    <Setter Property="Background">
        <Setter.Value>
            <LinearGradientBrush StartPoint="0,0" EndPoint="1,1">

```

```

        <GradientStop Color="#ffa9f9" Offset="0"/>
        <GradientStop Color="#ffe6ff" Offset="1"/>
    </LinearGradientBrush>
    </Setter.Value>
</Setter>
<Setter Property="Template">
    <Setter.Value>
        <ControlTemplate TargetType="ListView">
            <Border Background="{TemplateBinding Background}"
                BorderThickness="0"
                CornerRadius="10">
                <Border.Effect>
                    <DropShadowEffect Color="#7f547c"
                        Direction="270"
                        ShadowDepth="2"
                        Opacity="0.7"/>
                </Border.Effect>
                <Grid>
                    <Grid.RowDefinitions>
                        <RowDefinition Height="Auto"/>
                        <RowDefinition Height="*" />
                    </Grid.RowDefinitions>
                    <GridViewHeaderRowPresenter Grid.Row="0"
Columns="{Binding Path=View.Columns, RelativeSource={RelativeSource
TemplatedParent}}"/>
                    <ScrollViewer Grid.Row="1"
HorizontalScrollBarVisibility="Disabled" VerticalScrollBarVisibility="Auto">
                        <ItemsPresenter />
                    </ScrollViewer>
                </Grid>
            </Border>
        </ControlTemplate>
    </Setter.Value>
</Setter>
<Style.Triggers>
    <Trigger Property="IsMouseOver" Value="True">
        <Setter Property="Background">
            <Setter.Value>
                <LinearGradientBrush StartPoint="0,0" EndPoint="1,1">
                    <GradientStop Color="#cc87c7" Offset="0"/>
                    <GradientStop Color="#ffe6ff" Offset="1"/>
                </LinearGradientBrush>
            </Setter.Value>
        </Setter>
    </Trigger>
</Style.Triggers>
</Style>

<Style TargetType="GridViewColumnHeader">
    <Setter Property="Background" Value="#cc87c7"/>
    <Setter Property="Foreground" Value="#7f547c"/>
    <Setter Property="BorderThickness" Value="0"/>
    <Setter Property="Padding" Value="5"/>
    <Setter Property="FontSize" Value="14"/>
    <Setter Property="Template">
        <Setter.Value>

```

```

        <ControlTemplate TargetType="GridViewColumnHeader">
            <Border Background="{TemplateBinding Background}"
                BorderBrush="{TemplateBinding BorderBrush}"
                BorderThickness="{TemplateBinding BorderThickness}"
                Padding="{TemplateBinding Padding}"
                SnapsToDevicePixels="True">
                <ContentPresenter HorizontalAlignment="Center"
VerticalAlignment="Center" />
            </Border>
        </ControlTemplate>
    </Setter.Value>
</Setter>
<Style.Triggers>
    <Trigger Property="IsMouseOver" Value="True">
        <Setter Property="Background" Value="#ffa9f9" />
    </Trigger>
</Style.Triggers>
</Style>

<Style TargetType="ListViewItem">
    <Setter Property="Foreground" Value="#7f547c" />
    <Setter Property="FontSize" Value="18" />
    <Setter Property="FontWeight" Value="Bold" />
    <Setter Property="Background">
        <Setter.Value>
            <LinearGradientBrush StartPoint="0,0" EndPoint="1,1">
                <GradientStop Color="#ffa9f9" Offset="0"/>
                <GradientStop Color="#ffe6ff" Offset="1"/>
            </LinearGradientBrush>
        </Setter.Value>
    </Setter>
    <Style.Triggers>
        <Trigger Property="IsSelected" Value="True">
            <Setter Property="Background" Value="#cc87c7" />
        </Trigger>
        <Trigger Property="IsMouseOver" Value="True">
            <Setter Property="Background" Value="#cc87c7" />
        </Trigger>
    </Style.Triggers>
</Style>

<!-- Налаштування GroupBox-->
<Style TargetType="GroupBox">
    <Setter Property="FontFamily" Value="Georgia" />
    <Setter Property="Foreground" Value="#7f547c" />
    <Setter Property="FontSize" Value="18" />
    <Setter Property="FontWeight" Value="Bold" />
    <Setter Property="HorizontalContentAlignment" Value="Center" />
</Style>

</ResourceDictionary>

```

В.4 Файл Authorization.xaml.cs

```

using RecpieClient.ServiceRecpie;
using System.Windows;
using System.Windows.Input;

namespace RecpieClient
{
    public partial class Authorization : Window
    {
        // Змінні для роботи з сервером
        ServiceRecpieClient client = new ServiceRecpieClient();

        // Змінні вікон
        MainWindow mainWindow;
        Registration registration;

        // Конструктор
        public Authorization()
        {
            InitializeComponent();
        }

        // Кнопки
        // Авторизуватись
        private void SingIn_Click(object sender, RoutedEventArgs e)
        {
            try
            {
                int ID = client.Connect(Login.Text, Password.Password);
                if (ID != 0)
                {
                    mainWindow = new MainWindow(ID, this);
                    mainWindow.Show();
                    this.Hide();
                }
                else
                {
                    string message = "Неправильний логін або пароль";
                    string title = "Авторизація невдачна";
                    RecpieMessageBox messageBox = new
RecpieMessageBox(message, title);
                    messageBox.ShowDialog();
                }
                Login.Text = "";
                Password.Password = "";
            }
            catch
            {
                ClientException exception = new ClientException(404);
            }
        }

        // Зареєструватись
        private void SignUp_Click(object sender, RoutedEventArgs e)
        {
            registration = new Registration(this);
            registration.Show();
            this.Hide();
            Login.Text = "";
        }
    }
}

```

```

        Password.Password = "";
    }
    // Увійти без акаунту
    private void SingIn_NoLogin_Click(object sender, RoutedEventArgs e)
    {
        mainWindow = new MainWindow(0, this);
        mainWindow.Show();
        this.Hide();
    }

    private void Window_KeyDown(object sender,
System.Windows.Input.KeyEventArgs e)
    {
        if (e.Key == Key.Enter)
        {
            try
            {
                int ID = client.Connect(Login.Text, Password.Password);
                if (ID != 0)
                {
                    mainWindow = new MainWindow(ID, this);
                    mainWindow.Show();
                    this.Hide();
                }
                else
                {
                    string message = "Неправильний логін або пароль";
                    string title = "Авторизація невдачна";
                    RecpieMessageBox messageBox = new
RecpieMessageBox(message, title);
                    messageBox.ShowDialog();
                }
                Login.Text = "";
                Password.Password = "";
            }
            catch
            {
                ClientException exception = new ClientException(404);
            }
        }
    }
}
}
}
}

```

В.5 Файл MainWindow.xaml.cs

```

using RecpieClient.ServiceRecpie;
using System.Collections.Generic;
using System.Windows;
using System.Windows.Controls;

namespace RecpieClient
{
    public partial class MainWindow : Window
    {
        // Змінні для роботи з сервером
        ServiceRecpieClient client = new ServiceRecpieClient();
    }
}

```

```

int ID;

// Змінні вікон
Authorization authorizationWin;
CreateRecipe createRecipeWin;
Search searchWin;
Recipe recipeWin;
CalorieCalculator calorieCalculatorWin;
Favorite favoriteWin;
About aboutWin;
Information informationWin;

// Змінні
bool load; // Змінна для того щоб не змінювалися статуси улюблених
рецептів, при завантаженні їх на форму
int numOfPageNow = 1; // Поточна сторінка
int countOfPage; // Кількість сторінок
List<RecipeObject> recipeList; // Список рецептів
RecipeObject recipe1; // Рецепт що відображається у першому наборі
елементів
RecipeObject recipe2; // Рецепт що відображається у другому наборі
елементів
RecipeObject recipe3; // Рецепт що відображається у третьому наборі
елементів

// Конструктор
public MainWindow(int ID, Authorization authorizationWin)
{
    InitializeComponent();
    this.ID = ID;
    this.authorizationWin = authorizationWin;
}

// Функції вікна
// При завантаженні вікна
private void Window_Loaded(object sender, RoutedEventArgs e)
{
    Window_Loaded();
}
// При закритті вікна
private void Window_Closing(object sender,
System.ComponentModel.CancelEventArgs e)
{
    try
    {
        client.Disconnect(ID);
        authorizationWin.Show();
    }
    catch
    {
        ClientException exception = new ClientException(404);
    }
}

// Функції
// Оновлення змісту вікна, яке викликається з інших вікон
public void Window_Loaded()

```

```

{
    UpdateRecipeList();
    LoadRecipe();
    UpdatePageBox();
}
/// Відображення рецептів
// Оновити список рецептів і кількість сторінок
void UpdateRecipeList()
{
    try
    {
        recipeList = new List<RecipeObject>(client.GetRecipe());
    }
    catch
    {
        ClientException exception = new ClientException(404);
    }
    SearchFilter();
    countOfPage = recipeList.Count / 3;
    if (recipeList.Count % 3 > 0)
        countOfPage++;
}
// Фільтрація і сортування списку рецептів
void SearchFilter()
{
    if ((SearchBox.Text != "") && (SearchBox.Text != null))
    {
        for (int i = 0; i < recipeList.Count; i++)
        {
            for (int j = 0; j < SearchBox.Text.Length; j++)
            {
                if (SearchBox.Text.Length >
recipeList[i].Name.Length)
                {
                    recipeList.RemoveAt(i);
                    i--;
                    j = SearchBox.Text.Length;
                }
                else
                {
                    if (char.ToLower(SearchBox.Text[j]) !=
char.ToLower(recipeList[i].Name[j]))
                    {
                        recipeList.RemoveAt(i);
                        i--;
                        j = SearchBox.Text.Length;
                    }
                }
            }
        }
    }
}
// Відображення рецептів у вікні
void LoadRecipe()
{
    try
    {

```

```

load = true;
string favoriteRecipe = client.GetFavorite(ID);

if ((numOfPageNow - 1) * 3 < recipeList.Count)
{
    ImageBox1.Visibility = Visibility.Visible;
    NameBox1.Visibility = Visibility.Visible;
    DescriptionBox1.Visibility = Visibility.Visible;
    CheckBox1.Visibility = Visibility.Visible;
    ImageBox1.Source =
ImageByte.Convert(ImageByte.GetImageFromByteArray(recipeList[(numOfPageNow -
1) * 3].Content));
    NameBox1.Content = recipeList[(numOfPageNow - 1) *
3].Name;
    DescriptionBox1.Text = recipeList[(numOfPageNow - 1) *
3].Description;
    recipe1 = recipeList[(numOfPageNow - 1) * 3];
    if (favoriteRecipe.IndexOf("|" + recipe1.ID + "|") == -1)
        CheckBox1.IsChecked = false;
    else
        CheckBox1.IsChecked = true;
}
else
{
    ImageBox1.Visibility = Visibility.Hidden;
    NameBox1.Visibility = Visibility.Hidden;
    DescriptionBox1.Visibility = Visibility.Hidden;
    CheckBox1.Visibility = Visibility.Hidden;
}
if ((numOfPageNow - 1) * 3 + 1 < recipeList.Count)
{
    ImageBox2.Visibility = Visibility.Visible;
    NameBox2.Visibility = Visibility.Visible;
    DescriptionBox2.Visibility = Visibility.Visible;
    CheckBox2.Visibility = Visibility.Visible;
    ImageBox2.Source =
ImageByte.Convert(ImageByte.GetImageFromByteArray(recipeList[(numOfPageNow -
1) * 3 + 1].Content));
    NameBox2.Content = recipeList[(numOfPageNow - 1) * 3 +
1].Name;
    DescriptionBox2.Text = recipeList[(numOfPageNow - 1) * 3
+ 1].Description;
    recipe2 = recipeList[(numOfPageNow - 1) * 3 + 1];
    if (favoriteRecipe.IndexOf("|" + recipe2.ID + "|") == -1)
        CheckBox2.IsChecked = false;
    else
        CheckBox2.IsChecked = true;
}
else
{
    ImageBox2.Visibility = Visibility.Hidden;
    NameBox2.Visibility = Visibility.Hidden;
    DescriptionBox2.Visibility = Visibility.Hidden;
    CheckBox2.Visibility = Visibility.Hidden;
}
if ((numOfPageNow - 1) * 3 + 2 < recipeList.Count)
{

```

```

        ImageBox3.Visibility = Visibility.Visible;
        NameBox3.Visibility = Visibility.Visible;
        DescriptionBox3.Visibility = Visibility.Visible;
        CheckBox3.Visibility = Visibility.Visible;
        ImageBox3.Source =
ImageByte.Convert(ImageByte.GetImageFromByteArray(recipeList[(numOfPageNow -
1) * 3 + 2].Content));
        NameBox3.Content = recipeList[(numOfPageNow - 1) * 3 +
2].Name;
        DescriptionBox3.Text = recipeList[(numOfPageNow - 1) * 3
+ 2].Description;
        recipe3 = recipeList[(numOfPageNow - 1) * 3 + 2];
        if (favoriteRecipe.IndexOf("|" + recipe3.ID + "|") == -1)
            CheckBox3.IsChecked = false;
        else
            CheckBox3.IsChecked = true;
    }
    else
    {
        ImageBox3.Visibility = Visibility.Hidden;
        NameBox3.Visibility = Visibility.Hidden;
        DescriptionBox3.Visibility = Visibility.Hidden;
        CheckBox3.Visibility = Visibility.Hidden;
    }
    load = false;

    if (ID == 0)
    {
        CheckBox1.Visibility = Visibility.Hidden;
        CheckBox2.Visibility = Visibility.Hidden;
        CheckBox3.Visibility = Visibility.Hidden;
    }
}
catch
{
    load = false;
    ClientException exception = new ClientException(404);
}
}
// Оновлення відображуємого номеру сторінки
void UpdatePageBox()
{
    if (countOfPage != 0)
        PageBox.Content = "Сторінка " + numOfPageNow + " / " +
countOfPage;
    else
        PageBox.Content = "Сторінка 0 / 0";
}
// Оновлення вікна, при зміні тексту у строці пошуку
private void SearchBox_TextChanged(object sender,
TextChangedEventArgs e)
{
    UpdateRecipeList();
    LoadRecipe();
    UpdatePageBox();
}

```

```

// Кнопки
// Створити рецепт
private void AddRecipeButton_Click(object sender, RoutedEventArgs e)
{
    if (ID == 0)
    {
        string message = "Необхідно авторизуватись";
        string title = "Операція недоступна";
        RecpieMessageBox messageBox = new RecpieMessageBox(message,
title);
        messageBox.ShowDialog();
    }
    else {
        createRecipeWin = new CreateRecipe(ID, this);
        createRecipeWin.Show();
        this.Hide();
    }
}
// Наступна стрінка
private void NextPageButton_Click(object sender, RoutedEventArgs e)
{
    UpdateRecipeList();
    if (numOfPageNow + 1 <= countOfPage)
        numOfPageNow++;
    LoadRecipe();
    UpdatePageBox();
}
// Минула стрінка
private void PreviousPageButton_Click(object sender, RoutedEventArgs
e)
{
    UpdateRecipeList();
    if (numOfPageNow - 1 > 0)
        numOfPageNow--;
    LoadRecipe();
    UpdatePageBox();
}
// Розширений пошук
private void AdvancedSearchButton_Click(object sender,
RoutedEventArgs e)
{
    searchWin = new Search(ID, this);
    searchWin.Show();
    this.Hide();
}
// Вихід
private void ExitButton_Click(object sender, RoutedEventArgs e)
{
    this.Close();
}
// Калькулятор калорій
private void CalculatorButton_Click(object sender, RoutedEventArgs e)
{
    calorieCalculatorWin = new CalorieCalculator(this);
    calorieCalculatorWin.Show();
    this.Hide();
}

```

```

// Улюблені і мої рецепти
private void Favorite_Click(object sender, RoutedEventArgs e)
{
    if (ID == 0)
    {
        string message = "Необхідно авторизуватись";
        string title = "Операція недоступна";
        RecpieMessageBox messageBox = new RecpieMessageBox(message,
title);
        messageBox.ShowDialog();
    }
    else
    {
        favoriteWin = new Favorite(ID, this);
        favoriteWin.Show();
        this.Hide();
    }
}
// Довідка
private void AboutButton_Click(object sender, RoutedEventArgs e)
{
    aboutWin = new About();
    aboutWin.ShowDialog();
}
// Інфоргмація про про програму
private void InformationButton_Click(object sender, RoutedEventArgs
e)
{
    informationWin = new Information();
    informationWin.ShowDialog();
}

// Відкриття рецепта
private void NameBox1_MouseDoubleClick(object sender,
System.Windows.Input.MouseButtonEventArgs e)
{
    recipeWin = new Recipe(recipe1, ID, this);
    recipeWin.Show();
    this.Hide();
}
private void NameBox2_MouseDoubleClick(object sender,
System.Windows.Input.MouseButtonEventArgs e)
{
    recipeWin = new Recipe(recipe2, ID, this);
    recipeWin.Show();
    this.Hide();
}
private void NameBox3_MouseDoubleClick(object sender,
System.Windows.Input.MouseButtonEventArgs e)
{
    recipeWin = new Recipe(recipe3, ID, this);
    recipeWin.Show();
    this.Hide();
}

// Зміна статусу рецепта
private void CheckBox1_Checked(object sender, RoutedEventArgs e)

```

```

{
    try
    {
        if (load == false)
        {
            string favoriteRecipe = client.GetFavorite(ID);
            favoriteRecipe += "|" + recipe1.ID + "|";
            client.UpdateFavorite(ID, favoriteRecipe);
            recipe1.Rating++;
            client.RecipeRatingUpdate(recipe1);
        }
    }
    catch
    {
        ClientException exception = new ClientException(404);
    }
}
private void CheckBox2_Checked(object sender, RoutedEventArgs e)
{
    try
    {
        if (load == false)
        {
            string favoriteRecipe = client.GetFavorite(ID);
            favoriteRecipe += "|" + recipe2.ID + "|";
            client.UpdateFavorite(ID, favoriteRecipe);
            recipe2.Rating++;
            client.RecipeRatingUpdate(recipe2);
        }
    }
    catch
    {
        ClientException exception = new ClientException(404);
    }
}
private void CheckBox3_Checked(object sender, RoutedEventArgs e)
{
    try
    {
        if (load == false)
        {
            string favoriteRecipe = client.GetFavorite(ID);
            favoriteRecipe += "|" + recipe3.ID + "|";
            client.UpdateFavorite(ID, favoriteRecipe);
            recipe3.Rating++;
            client.RecipeRatingUpdate(recipe3);
        }
    }
    catch
    {
        ClientException exception = new ClientException(404);
    }
}
private void CheckBox1_Unchecked(object sender, RoutedEventArgs e)
{
    try
    {

```

```

        if (load == false)
        {
            string favoriteRecipe = client.GetFavorite(ID);
            favoriteRecipe = favoriteRecipe.Replace("|" + recipe1.ID
+ "|", "");

            client.UpdateFavorite(ID, favoriteRecipe);
            recipe1.Rating--;
            client.RecipeRatingUpdate(recipe1);
        }
    }
    catch
    {
        ClientException exception = new ClientException(404);
    }
}
private void CheckBox2_Unchecked(object sender, RoutedEventArgs e)
{
    try
    {
        if (load == false)
        {
            string favoriteRecipe = client.GetFavorite(ID);
            favoriteRecipe = favoriteRecipe.Replace("|" + recipe2.ID
+ "|", "");

            client.UpdateFavorite(ID, favoriteRecipe);
            recipe2.Rating--;
            client.RecipeRatingUpdate(recipe2);
        }
    }
    catch
    {
        ClientException exception = new ClientException(404);
    }
}
private void CheckBox3_Unchecked(object sender, RoutedEventArgs e)
{
    try
    {
        if (load == false)
        {
            string favoriteRecipe = client.GetFavorite(ID);
            favoriteRecipe = favoriteRecipe.Replace("|" + recipe3.ID
+ "|", "");

            client.UpdateFavorite(ID, favoriteRecipe);
            recipe3.Rating--;
            client.RecipeRatingUpdate(recipe3);
        }
    }
    catch
    {
        ClientException exception = new ClientException(404);
    }
}
}
}

```

ДОДАТОК Г
Слайди презентації

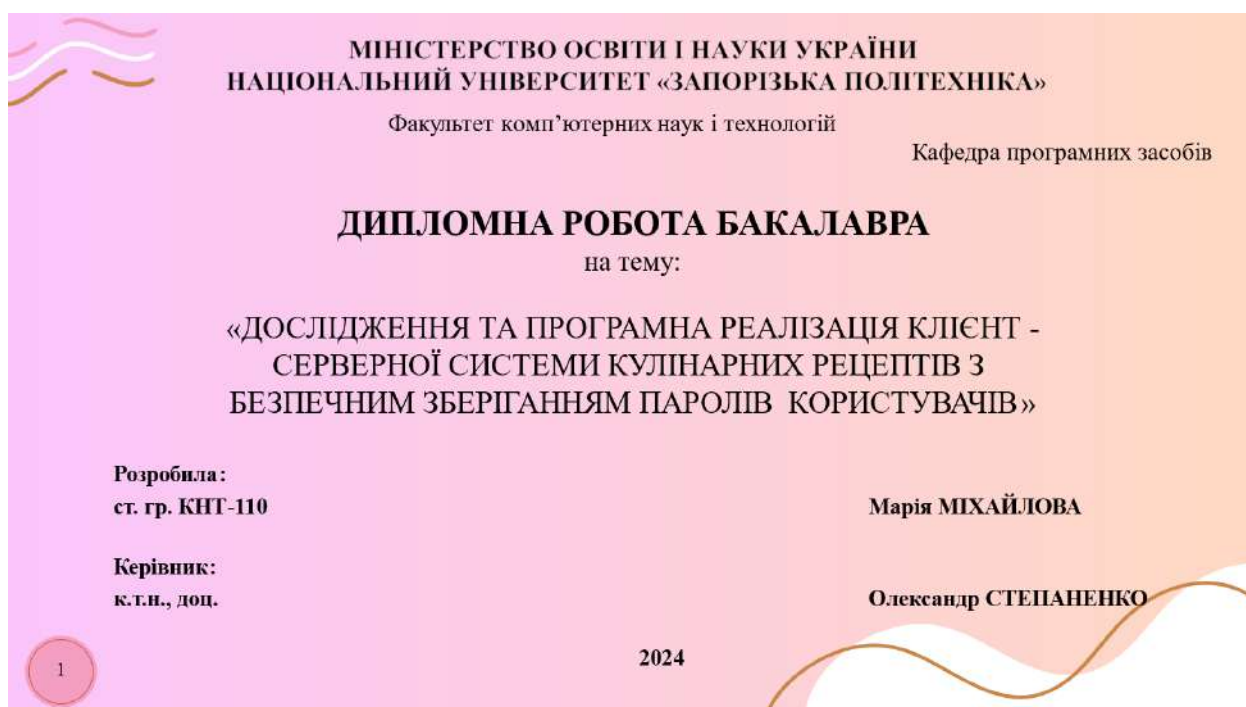


Рисунок Г.1 – Перший слайд



Рисунок Г.2 –Другий слайд

Об'єкт розробки

Об'єктом розробки є застосунок «Ресріє» для кулінарних рецептів. Цей об'єкт включає в себе всі компоненти та модулі, необхідні для реалізації функціональності застосунку. Він складається з клієнтської частини (десктопний інтерфейс) та серверної частини, які взаємодіють між собою за допомогою клієнт-серверної архітектури.

3

Рисунок Г.3 –Третій слайд

ЗАДАЧІ РОЗРОБКИ

- Розробка модуля реєстрації та авторизації користувачів.
- Розробка модуля шифрації пароля користувача з використанням хеш-функції.
- Збереження та організація рецептів.
- Пошук та фільтрація рецептів.
- Відображення характеристик рецептів.
- Відображення характеристик рецептів.

4

Рисунок Г.4 –Четвертий слайд

Порівняння існуючих аналогів

Характеристика	Recpie	1000.menu	Smachno.ua	Allrecipes
Ресстрація та авторизація користувачів	+	+	-	+
Збереження та організація рецептів	+	+	-	+
Пошук та фільтрація рецептів	+	+	+	+
Рецепти з кроками та інгредієнтами	+	+	+	+
Відображення характеристик рецептів	+	+	+	+
Обмін рецептами та спільнота	+	-	+	+
Калькулятор калорій	+	-	+	+
Україномовний інтерфейс	+	-	+	-
Зручний інтерфейс	+	+	Частково	Частково
Шифрування паролів	+	+	-	+

5

Рисунок Г.5 –П'ятий слайд

Порівняння мов програмування

Ознака	JavaScript	Python	Java	C#
Веб-розробка	Основна мова	Популярна мова	Рідна підтримка	Рідна підтримка
Серверна підтримка	Застосовується (Node.js)	Застосовується	Застосовується (Spring)	Застосовується (ASP.NET)
Синтаксис	C-подібний	Простий та зрозумілий	Java-подібний	C-подібний
Швидкість	Висока швидкість	Середня швидкість	Висока швидкість	Висока швидкість
Масштабованість	Добре масштабується	Добре масштабується	Добре масштабується	Добре масштабується
Витрати	Безкоштовна	Безкоштовна	Безкоштовна	Частково платна
Величина спільноти	Велика	Велика	Велика	Велика

6

Рисунок Г.6 – Шостий слайд

Функції системи

- реєстрація та авторизація користувачів. Можливість реєстрації нових користувачів і авторизації існуючих;
- збереження та організація рецептів. Можливість додавати, редагувати, видаляти рецепти та додавати до улюблених;
- пошук та фільтрація рецептів. Можливість швидкого та ефективного пошуку рецептів залежно від різних параметрів, таких як назва, інгредієнти, тип страви. Можливість сортувати рецепти за популярністю та датою;
- рецепти з кроками та інгредієнтами. Додавання до рецепту докладні кроки готування для кожного рецепту, а також перелік необхідних інгредієнтів з відповідними кількостями;
- відображення характеристик рецептів. Надання інформації про час приготування, кількість порцій та інші характеристики рецепту;
- обмін рецептами та спільнота. Можливість ділитися своїми рецептами, залишати коментарі та відгуки;
- калькулятор калорій. Можливість розраховувати калорійність страви.

9

Рисунок Г.9 – Дев'ятий слайд

Проектування системи



Діаграма класів клієнтської частини

10

Рисунок Г.10 – Десятий слайд

Приклади інтерфейсу

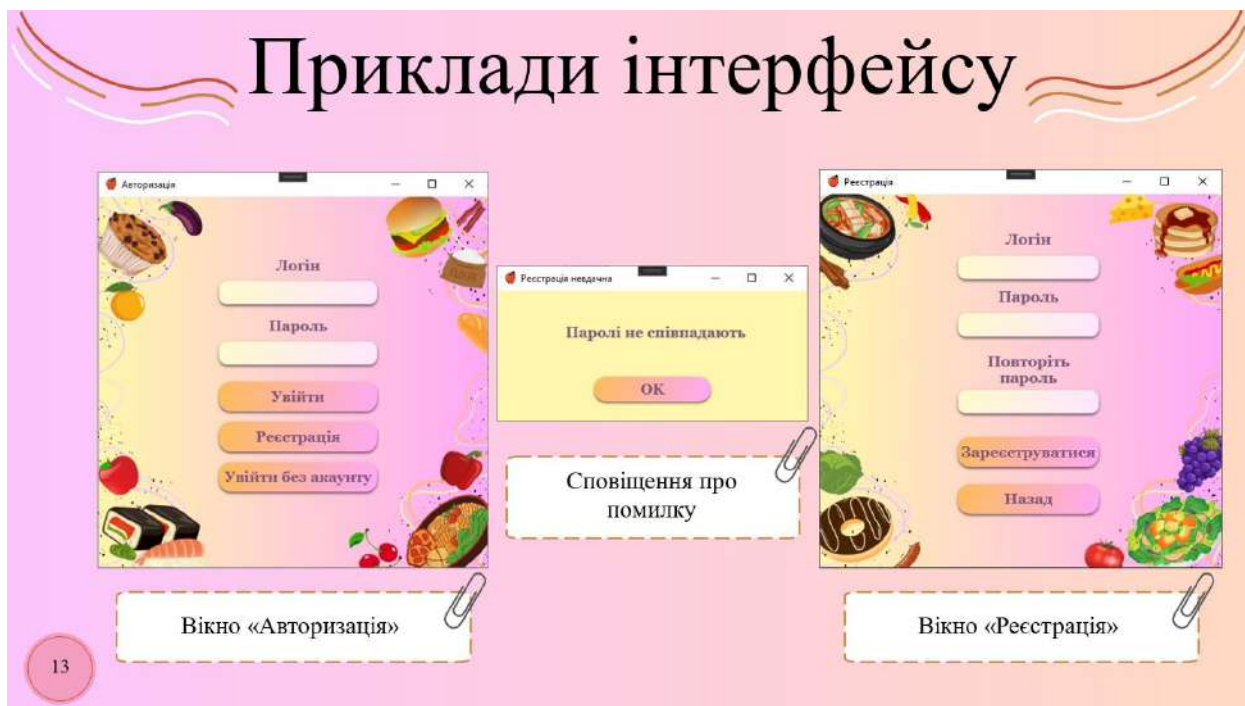


Рисунок Г.13 – Тринадцятий слайд

Приклади інтерфейсу



Рисунок Г.14 – Чотирнадцятий слайд

Приклади інтерфейсу



Вікно «Улюблені та мої рецепти»



Вікно для створення рецепту

Рисунок Г.15 – П'ятнадцятий слайд

Приклади інтерфейсу



Вікно з розширеним пошуком



Додаткове вікно для пошуку за інгредієнтом

Рисунок Г.16 – Шістнадцятий слайд

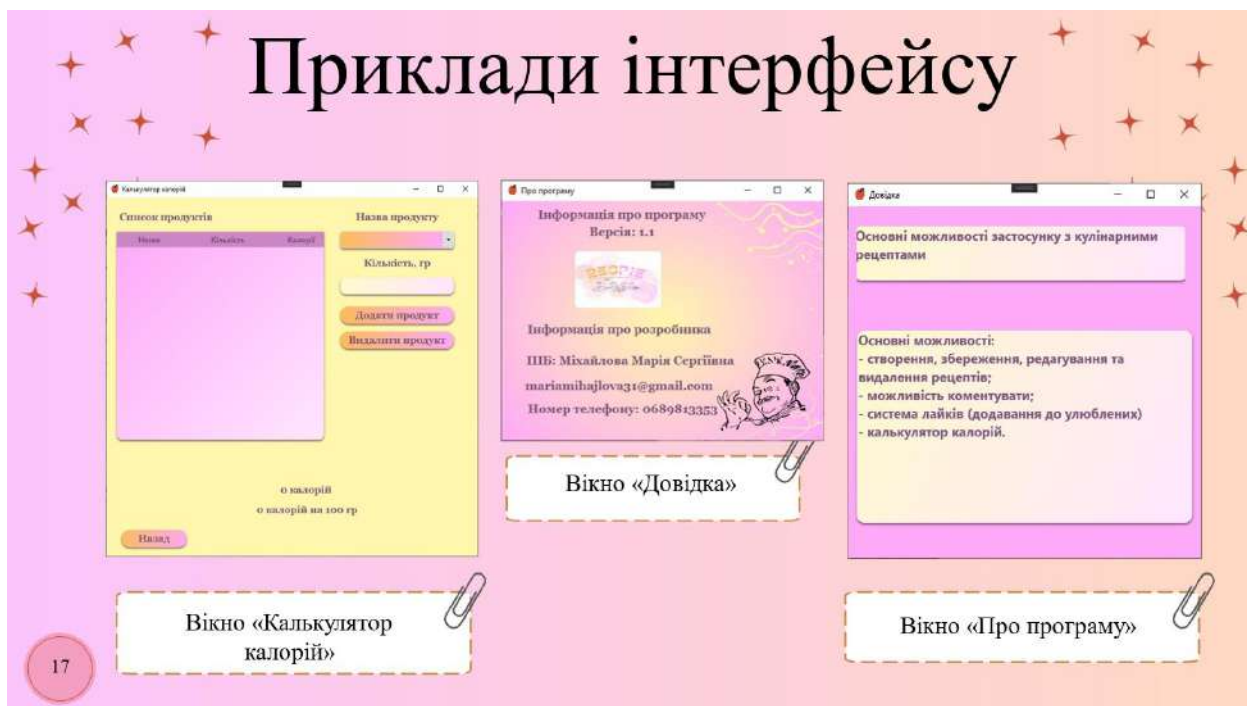


Рисунок Г.17 – Сімнадцятий слайд

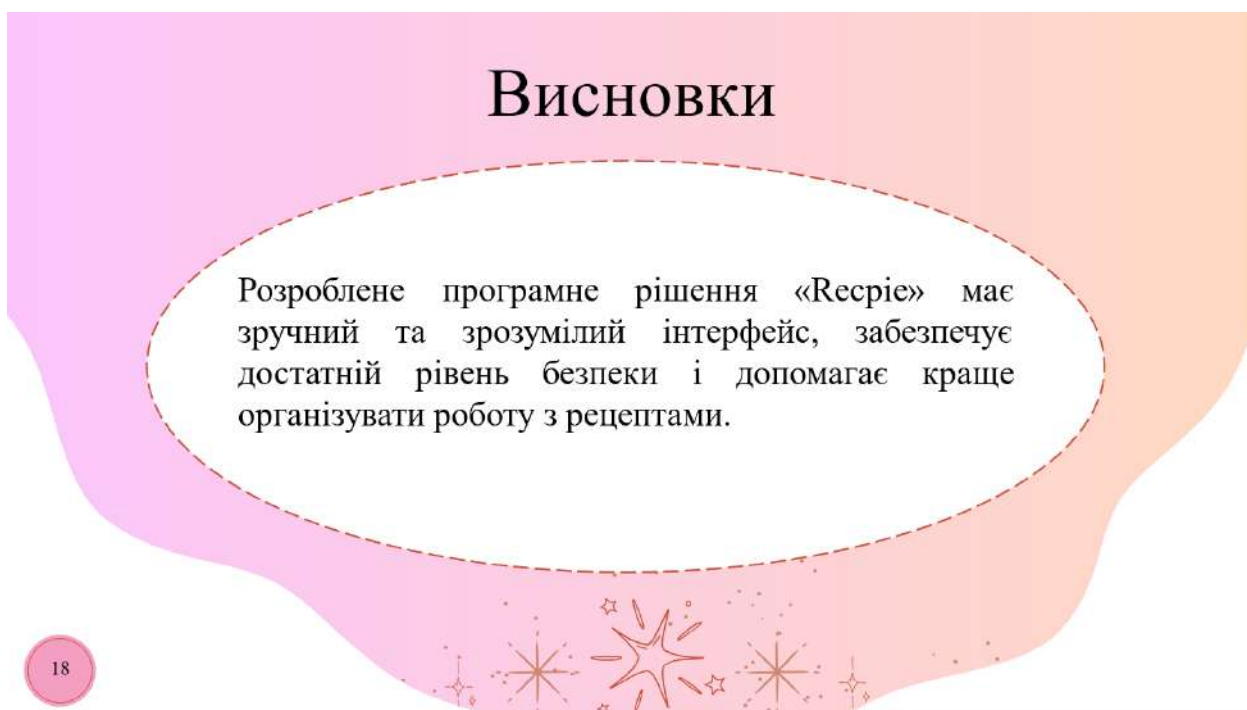


Рисунок Г.18 – Вісімнадцятий слайд

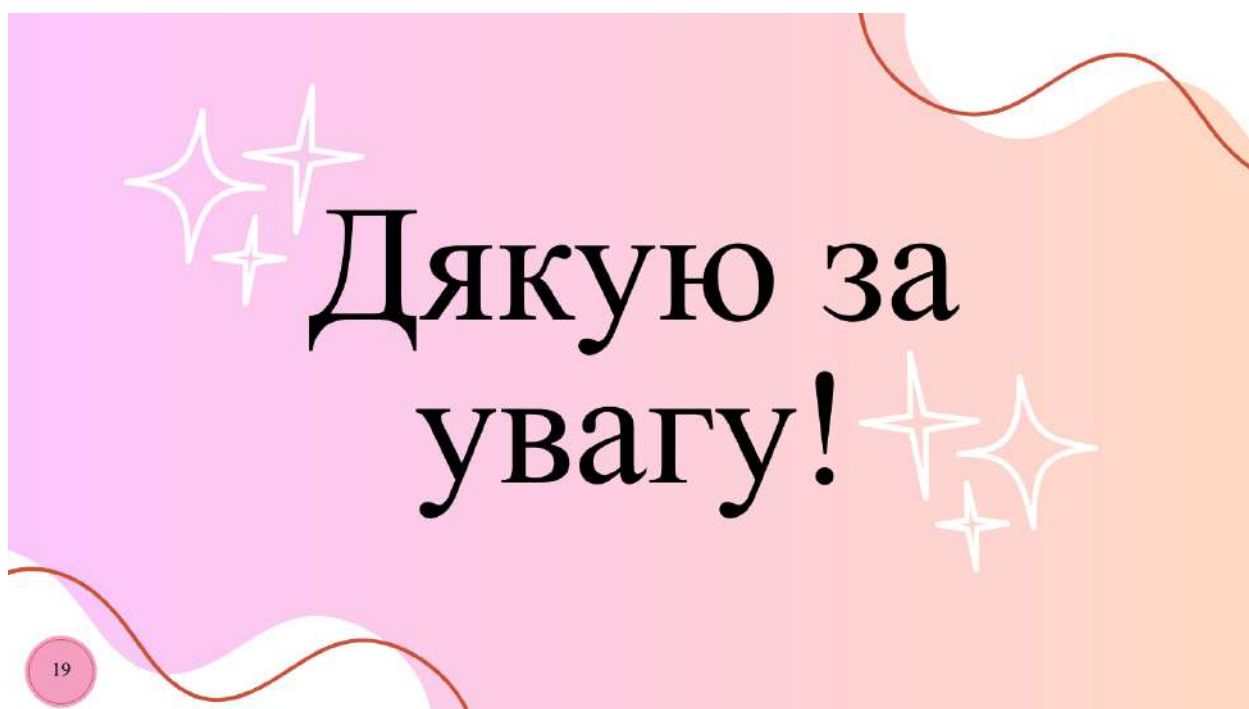


Рисунок Г.19 –Дев'ятнадцятий слайд