Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська дипломна робота

на тему: «Розробка вебзастосунку для підбору рецептів за інгредієнтами з можливістю придбання необхідних продуктів»

Виконала: студентка 4 курсу
група ЗПІ-20б спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Шаповалова О.О.
(прізвище та ініціали)
Керівник: к.т.н., доц. каф. ПЗ Романюк О.В.
(прізвище та ініціали)
Рецензент: к.т.н., доцент каф. ЗІ Войтович О.П.
(прізвище та ініціали)

Допущено до захисту

Завідувач кафедри ПЗ

д.т.н., проф. Романюк О.Н.
(прізвище та ініціали)

«10» червня 2024 р.

ВНТУ – 2024

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший бакалаврський
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О.Н.
«12» березня 2024 р.

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Шаповаловій Ользі Олександрівні

1. Тема роботи – «Розробка вебзастосунку для підбору рецептів за інгредієнтами з можливістю придбання необхідних продуктів»

Керівник роботи: Романюк Оксана Володимирівна, к.т.н., доц. кафедри ПЗ, затверджені наказом вищого навчального закладу від 11 березня 2024 р. № 80.

2. Строк подання студентом роботи 3 червня 2024 р.

3. Вихідні дані до роботи: тип програми – вебзастосунок; модель розробки – ітеративна; засоби організації даних програми – СКБД MongoDB; вхідні дані: персональні дані користувача; вихідні дані: перелік рецептів, список інгредієнтів, фото та відео рецептів; мова програмування – JavaScript.

4. Зміст розрахунково-пояснювальної записки: вступ; аналіз стану розвитку програм для підбору рецептів; розробка моделі, методів і алгоритмів роботи вебзастосунку; розробка програмного забезпечення; тестування розробленого програмного застосунку; висновки; список використаних джерел; додатки.

5. Перелік графічного матеріалу: титульний слайд; актуальність теми; мета, об'єкт та предмет дослідження; задачі дослідження, новизна, практична цінність одержаних результатів; порівняльний аналіз аналогів, використані технології,

метод замовлення продуктів, метод пошуку рецептів, тестування, висновки, апробація результатів роботи і публікації.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Романюк О.В., к.т.н., доцент кафедри ПЗ	13.03.2024	03.06.2024

7. Дата видачі завдання 13 березня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз розвитку програм для підбору рецептів	14.03.2024 – 24.03.2024	виконано
2	Розробка моделі вебзастосунку	25.03.2024 – 31.03.2024	виконано
3	Розробка методу замовлення продуктів	01.04.2024 – 07.04.2024	виконано
4	Розробка методу пошуку рецептів	08.04.2024 – 15.04.2024	виконано
5	Розробка алгоритмів вебзастосунку	16.04.2024 – 23.04.2024	виконано
6	Аналіз і вибір мови програмування та середовища розробки	24.04.2024 – 30.04.2024	виконано
7	Розробка програмного забезпечення	01.05.2024 – 19.05.2024	виконано
8	Тестування програмного забезпечення	20.05.2024 – 24.05.2024	виконано
9	Оформлення матеріалів до захисту БДР	25.05.2024 – 30.05.2024	виконано

Студент _____ **Шаповалова О.О.**
(підпис) (прізвище та ініціали)

Керівник бакалаврської дипломної роботи _____ **Романюк О.В.**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська дипломна робота складається з 62 сторінок формату А4, на яких є 39 рисунків, 3 таблиці, список використаних джерел містить 22 найменування.

У бакалаврській дипломній роботі проведено аналіз стану програм для підбору рецептів. Було проведено аналіз аналогів, визначено їх переваги і недоліки та доведено доцільність розробки власного програмного застосунку.

Запропоновано метод пошуку рецептів, особливість якого полягає у можливості здійснювати пошук рецептів за наявними інгредієнтами, що полегшує користувачам підбір страви для приготування.

Запропоновано метод замовлення продуктів, який дозволяє підбирати продукти, необхідні для приготування обраного рецепту, з можливістю їх доставки.

Виконане обґрунтування вибору мови та середовища програмування, а також технологій, які були використані під час розробки мобільного застосунку. Розроблено програмний продукт, який являє собою вебзастосунок.

Робота реалізована за допомогою мови програмування JavaScript. В якості середовища для розробки програмного забезпечення було обрано Visual Studio Code. Під час розробки було використано сторонні фреймворки, зокрема Express.js, React.js, Node.js. Також було використано систему управління базами даних MongoDB.

Отримані в бакалаврській дипломній роботі результати можна використати для підбору рецептів та придбання необхідних інгредієнтів.

Ключові слова: вебзастосунок, рецепти, аналіз.

ABSTRACT

The bachelor thesis consists of 62 pages of A4 format, on which there are 39 figures, 3 tables, the list of used sources contains 22 titles.

An analysis of the status of programs for selecting recipes was carried out in the bachelor thesis. An analysis of analogues was carried out, their advantages and disadvantages were determined, and the feasibility of developing one's own software application was proven.

A recipe search method is proposed, the feature of which is the ability to search for recipes based on available ingredients, which makes it easier for users to select a dish to prepare.

A method of ordering products is proposed, which allows you to select the products necessary for the preparation of the selected recipe, with the possibility of their delivery.

The justification of the choice of language and programming environment, as well as the technologies that were used during the development of the mobile application, was carried out. A software product, which is a web application, has been developed.

The work is implemented using the JavaScript programming language. Visual Studio Code was chosen as the software development environment. During development, third-party frameworks were used, including Express.js, React.js, Node.js. The MongoDB database management system was also used.

The results obtained in the bachelor thesis can be used to select recipes and purchase the necessary ingredients.

Keywords: web application, recipes, analysis.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Організація процесу приготування страв хоч і може спершу видатися простим завданням, особливо в домашніх умовах, насправді, це може нести свої незручності та ризики. Інколи трапляються ситуації, коли стає проблематично підібрати страву для приготування, виходячи з наявних інгредієнтів. Також, є інша можлива проблема, яка полягає у відсутності зручного способу придбати необхідні продукти.

Причинами таких проблем можуть бути:

1. Деякі рецепти вимагають специфічних інгредієнтів, які можуть бути важко знайти в місцевих магазинах. Це особливо стосується екзотичних або регіональних продуктів

2. Навіть якщо уже заплановано приготувати страву з вже наявних продуктів, виявляється, що ключові інгредієнти можуть бути відсутніми, що змінює плани.

3. Пошук альтернативних інгредієнтів або магазинів, де їх можна придбати, може забирати багато часу і зусиль. Це може бути особливо проблематично для зайнятих людей або тих, хто живе в місцях з обмеженим доступом до різноманітних магазинів.

4. Знаходження замінників для відсутніх інгредієнтів часто вимагає додаткових витрат на нові продукти або компромісні варіанти, що може підвищити витрати на приготування страв.

5. Якщо людина уже запланувала покупку необхідних продуктів заздалегідь, іноді стаються несподівані ситуації, коли змінюється робочий графік або виявляється, що продукти у магазині неналежної якості.

Отже, актуальною є розробка вебзастосунку для підбору рецептів за інгредієнтами з можливістю придбання необхідних продуктів.

Зв’язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно з планом виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою роботи є удосконалення процесу пошуку рецептів та замовлення необхідних продуктів за рахунок розробки спеціального вебзастосунку, в якому реалізовані нові методи пошуку рецептів та замовлення продуктів.

Відповідно до поставленої мети потрібно вирішити такі **завдання**:

- здійснити аналіз стану питання та методів розв'язання задачі;
- розробити модель, загальний алгоритм роботи та діаграми станів застосунку;
- розробити метод замовлення продуктів;
- розробити метод пошуку рецептів;
- здійснити програмну реалізацію вебсистеми;
- провести тестування розробленого функціоналу;
- описати системні вимоги вебзастосунку для підбору рецептів за інгредієнтами з можливістю придбання необхідних продуктів.

Об'єктом дослідження є процес підбору рецептів за інгредієнтами з можливістю придбання необхідних продуктів.

Предметом дослідження є методи і засоби реалізації вебзастосунку для підбору рецептів за інгредієнтами з можливістю придбання необхідних продуктів.

Методи дослідження. У процесі досліджень використовувалися: порівняння функціональних характеристик програм-аналогів для виявлення їх недоліків та формулювання завдань на їх вдосконалення; теорія алгоритмів для розроби та візуалізації алгоритмів роботи програмного забезпечення; комп'ютерне моделювання для аналізу та перевірки отриманих теоретичних положень.

Новизна отриманих результатів:

1. Удосконалено метод пошуку рецептів, який, на відміну від відомих, дозволяє здійснювати пошук рецептів за наявними інгредієнтами, що полегшує користувачам підбір страви для приготування.

2. Удосконалено метод замовлення продуктів, який, на відміну від відомих,

дозволяє підбирати продукти, необхідні для приготування обраного рецепту, з можливістю їх доставки, що полегшує та пришвидшує процес отримання необхідних для приготування продуктів.

Практична цінність отриманих результатів. Практична цінність отриманих результатів полягає в тому, що на основі отриманих у бакалаврській дипломній роботі теоретичних положень запропоновано алгоритми та розроблено програмні засоби для удосконалення процесів пошуку рецептів та замовлення необхідних інгредієнтів.

Особистий внесок здобувача. Усі результати, викладені у бакалаврській дипломній роботі, отримані автором особисто. У науковій праці, опублікованій у співавторстві, автору належать такі результати: таблиця порівняння функціональних характеристик аналогів, блок-схема алгоритму роботи застосунку[1]; аналіз основних компонентів MERN Stack для розробки вебзастосунків та їх переваги [2].

Апробація результатів роботи. Результати роботи доповідалися на:

- LIII Науково-технічній конференції підрозділів Вінницького національного технічного університету (Вінниця, 2024)» [1];
- Міжнародній науково-практичній інтернет-коференції «Молодь в науці: дослідження, проблеми, перспективи» (МН-2024) [2].

Публікації. За тематикою дослідження опубліковано 2 наукові праці у збірниках матеріалів конференцій.

1 АНАЛІЗ СТАНУ РОЗВИТКУ ПРОГРАМ ДЛЯ ПІДБОРУ РЕЦЕПТІВ

1.1 Аналіз предметної області

Застосунки для підбору рецептів є популярним і затребуваним напрямком у сфері кулінарних застосунків та сервісів. Такі застосунки спрямовані на надання користувачам широкого доступу до великих баз даних рецептів та можливості персоналізувати процес пошуку та вибору страв під свої смаки, дієтичні переваги та наявні продукти.

Сучасний стан розвитку таких застосунків характеризується постійним вдосконаленням функціональності та додаванням нових можливостей. Провідні рішення, такі як Cookpad, Yummly та Mealime, пропонують користувачам цілий спектр функцій: пошук рецептів за різноманітними критеріями, збереження улюблених страв, автоматичне формування списків покупок, а подекуди й інтеграцію з сервісами доставки продуктів.

Актуальність розробки нових застосунків для підбору рецептів обґрунтовується низкою важливих факторів. По-перше, сучасні користувачі потребують зручних інструментів для пошуку страв на основі наявних продуктів, особливо коли бракує окремих інгредієнтів для улюблених рецептів. По-друге, планування та ефективне придбання необхідних продуктів часто є проблемою, яку можна вирішити завдяки інтеграції застосунків з сервісами доставки. Крім того, постійно зростає попит на мобільні та вебзастосунки, що спрощують повсякденні завдання, зокрема приготування їжі.

Характерними особливостями таких кулінарних застосунків є великі бази даних рецептів, різноманітні можливості пошуку та фільтрації, функції збереження улюблених страв, детальна інформація про інгредієнти та покрокові інструкції приготування, а також елементи персоналізації рекомендацій під індивідуальні вподобання користувачів. Найбільшого поширення такі застосунки отримали на мобільних платформах iOS та Android, а також у вигляді вебсервісів, оскільки вони забезпечують найбільш зручний та доступний спосіб взаємодії з

кулінарними рецептами та можливостями замовлення продуктів "на ходу".

Розвиток застосунків для підбору рецептів та планування покупок продуктів є важливим трендом у сфері кулінарних онлайн-сервісів. Поєднання великих баз даних рецептів, персоналізованих рекомендацій та можливості замовлення необхідних інгредієнтів робить такі застосунки дедалі затребуванішими серед сучасних користувачів, які прагнуть спростити процес планування та приготування домашніх страв.

1.2 Порівняльний аналіз аналогів

При визначенні вимог до розроблюваного вебзастосунку для підбору рецептів, було проведено аналіз існуючих аналогів. Це необхідно для кращого розуміння потреб користувачів та виявлення функціоналу, який ще не реалізовано в цих застосунках.

Cookpad [3] – це мобільний застосунок та веб-сайт, який надає користувачам можливість знаходити, ділитися та зберігати рецепти страв. Він пропонує велику базу даних рецептів, створених користувачами з усього світу, що дозволяє знайти рецепти на будь-який смак та в будь-якому стилі кулінарії. Інтерфейс застосунка Cookpad наведено на рисунку 1.1.

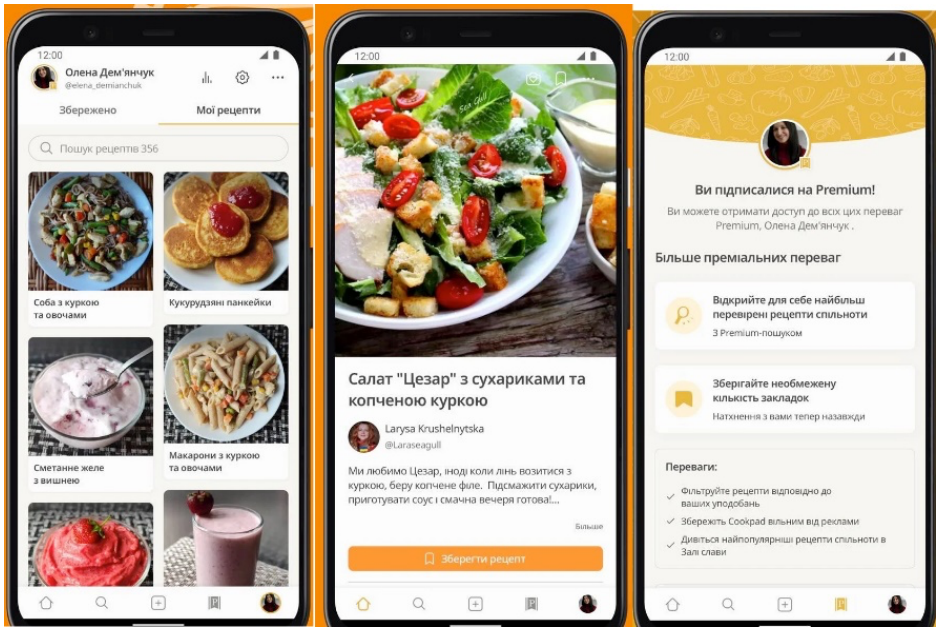

Рисунок 1.1 – Інтерфейс застосунку Cookpad

Основні характеристики Cookpad включають:

1. Користувачі можуть шукати рецепти за різними критеріями, такими як інгредієнти, типи страв, кухні, складність приготування та інші.

2. Користувачі можуть зберігати рецепти, які їм подобаються, у власному обліковому записі для подальшого використання.

3. Користувачі можуть ділитися своїми власними рецептами з іншими користувачами спільноти Cookpad.

Yummly [4] – це онлайн-платформа та мобільний застосунок, який надає користувачам можливість знаходити, зберігати та ділитися рецептами страв. Він відомий своєю великою базою даних рецептів, інтеграцією зі смарт-приладами та іншими корисними функціями.

Головну сторінку веб-сайту Yummly наведено на рисунку 1.2.

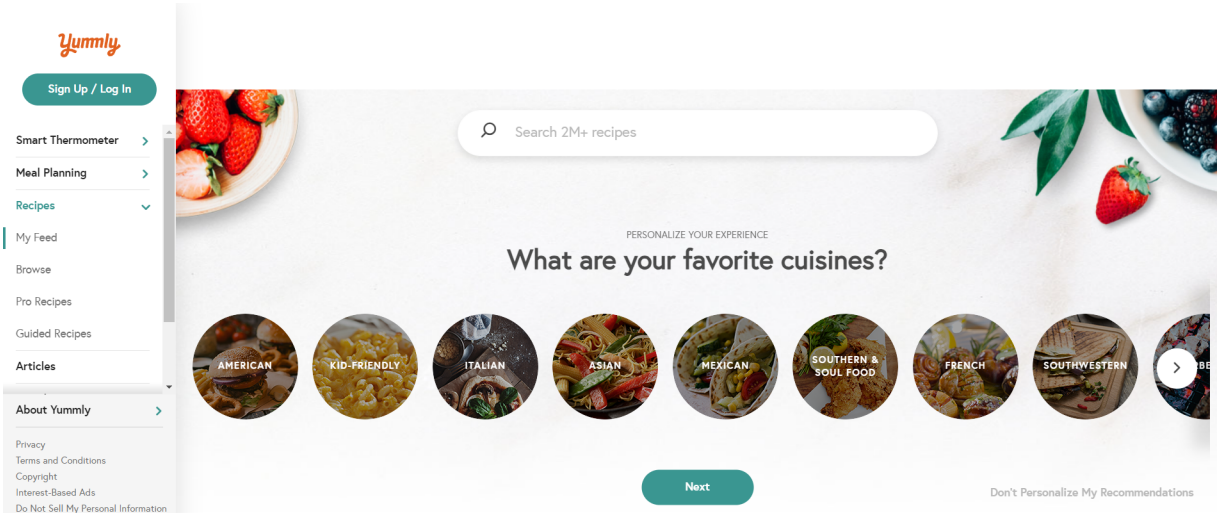

Рисунок 1.2 – Головна сторінка Yummly

Основні характеристики Yummly включають:

1. Yummly використовує алгоритми рекомендацій для адаптації підбору рецептів під індивідуальні вподобання та потреби користувачів.

2. Користувачі можуть шукати рецепти за різними категоріями, такими як тип страв (сніданок, обід, вечеря), кухні світу, дієтичні обмеження та інші.

3. Користувачі можуть зберігати рецепти у власному обліковому записі на платформі для подальшого використання.

4. Yummly автоматично генерує список покупок на основі обраних рецептів, що дозволяє користувачам зручно планувати покупки продуктів.

Mealime [5] – це мобільний застосунок, спрямований на полегшення процесу планування та приготування їжі для користувачів. Він пропонує різноманітні рецепти з можливістю персоналізації та генерації списків покупок для зручності користувачів.

Застосунок Mealime продемонстровано на рисунку 1.3.

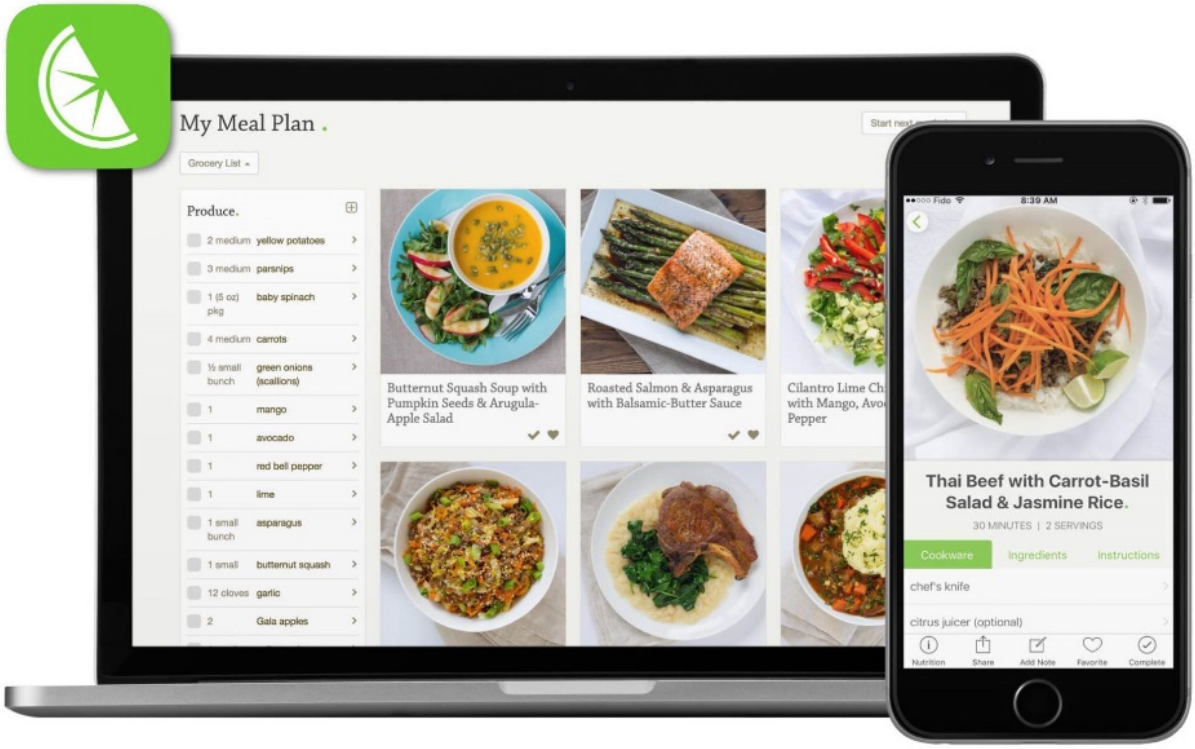

Рисунок 1.3 – Інтерфейс застосунку Mealime

Основні характеристики Mealime включають:

1. Користувачі можуть вибирати рецепти на кожен день тижня з великого списку доступних опцій.

2. Вебзастосунок враховує індивідуальні дієтичні обмеження та вподобання користувачів, пропонуючи рецепти, які відповідають їхнім потребам.

3. Вебзастосунок автоматично генерує списки покупок на основі обраних рецептів, допомагаючи користувачам ефективно планувати покупки продуктів.

4. Кожен рецепт супроводжується детальними інструкціями приготування та фотографіями, що дозволяє користувачам легко готувати страви навіть без великого досвіду в кулінарії.

Для наочної демонстрації відмінностей розглянутих застосунків було зведено їх переваги і недоліки у таблицю порівняння (таблиця 1.1).

Таблиця 1.1 – Порівняльний аналіз аналогів

	Cookpad	Yummly	Mealime	Mealthy
Пошук рецептів	1	1	1	1
Збереження улюблених рецептів	1	1	1	1
Детальна інформація про рецепти	1	1	1	1
Список покупок	0	1	1	1
Замовлення продуктів з магазину	0	0	0	1
Сумарний бал	**3**	**4**	**4**	**5**

Проаналізувавши результати з таблиці 1, можна зробити висновок, що Mealthy має вищий сумарний бал за Cookpad на 40%, за Yummly та Mealime на 20%.

1.3 Аналіз методів розв’язання задачі

Для розробки вебзастосунку для підбору рецептів за інгредієнтами з можливістю придбання необхідних продуктів можуть бути використані різноманітні підходи та інструменти. Одним із популярних варіантів є застосування MERN stack [6] – сучасного технологічного стеку, що включає в себе MongoDB [7], Express.js [8], React.js [9] та Node.js [10].

MERN stack є одним з найкращих варіантів для розробки такого вебзастосунку з кількох ключових причин. По-перше, використання єдиної мови програмування, JavaScript, як для frontend, так і для backend частини, значно спрощує розробку та підтримку проекту. По-друге, MERN stack характеризується високим рівнем гнучкості та масштабованості завдяки модульній архітектурі на

основі мікросервісів. Крім того, екосистема бібліотек та інструментів, доступних для MERN stack, є надзвичайно широкою та активно розвивається, що прискорює процес розробки.

Вибір NoSQL бази даних MongoDB також є вдалим рішенням для такого типу застосунку, оскільки гнучкість у структурі даних, масштабованість та високі показники продуктивності роблять її ідеальним вибором для зберігання та обробки даних про рецепти, інгредієнти, користувачів тощо.

Приклад використання MongoDB наведено на рисунку 1.4.

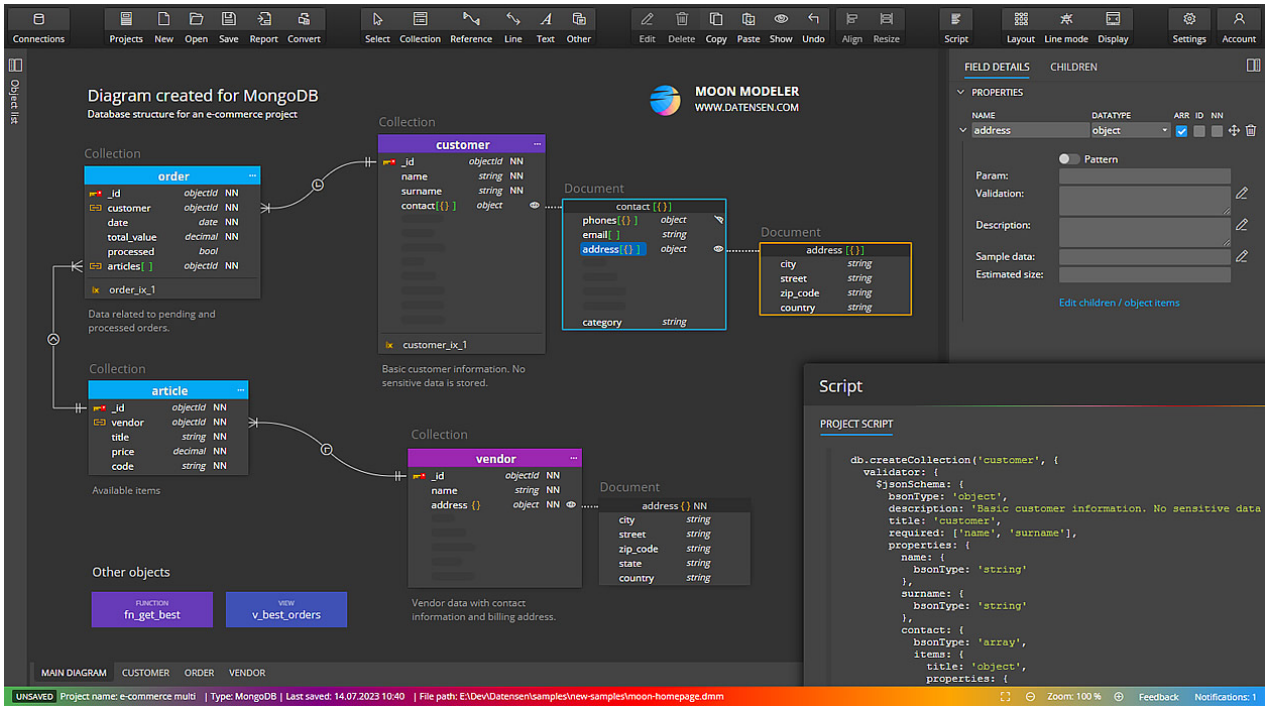

Рисунок 1.4 – Приклад використання MongoDB

Крім того, MERN stack дозволяє створювати SPA (Single Page Application) – застосунки, що забезпечують плавну та інтерактивну взаємодію користувача, що відповідає сучасним вимогам до користувацького досвіду.

Порівняння традиційного багатосторінкового застосунку та односторінкового застосунку наведено на рисунку 1.5.

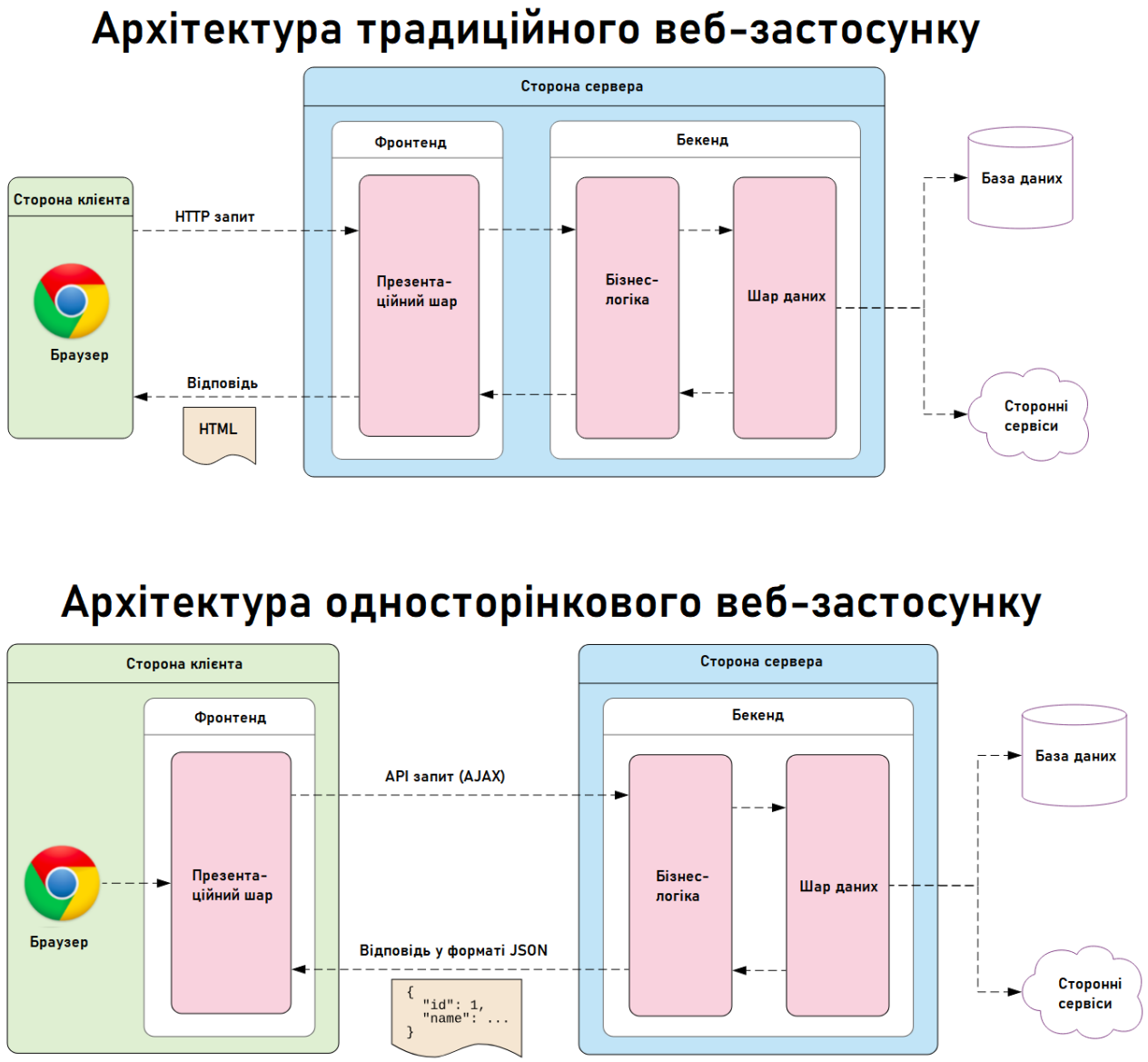

Рисунок 1.5 – Архітектура багатосторінкового за односторінкового застосунків

Express.js – це мінімалістичний та гнучкий веб-фреймворк для Node.js. Він надає набір функціональностей для створення веб-застосунків та API. Express.js спрощує процес маршрутизації, обробки запитів та відповідей, а також управління сесіями та cookies. Розробники цінують Express.js за його простоту, швидкість розробки та велику екосистему плагінів (middleware), які дозволяють легко розширювати функціональність програми. Він є основою для багатьох популярних фреймворків Node.js вищого рівня.

Архітектура Express.js наведена на рисунку 1.6.

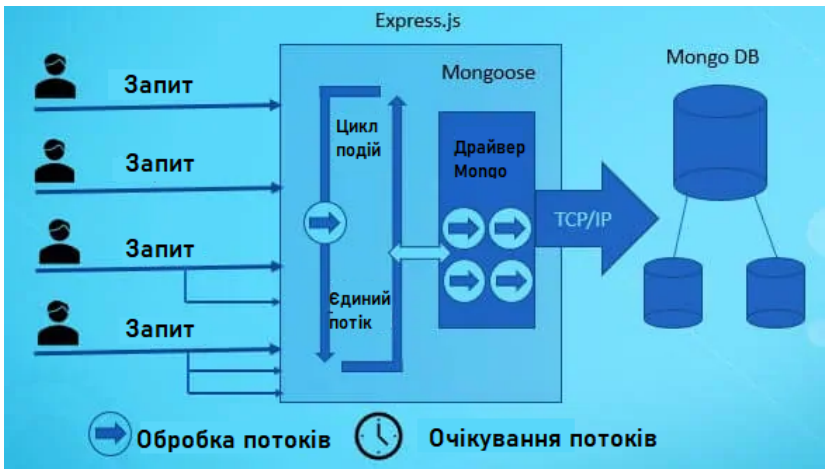

Рисунок 1.6 – Архітектура Express.js

React.js – це бібліотека JavaScript з відкритим кодом для створення користувацьких інтерфейсів. Розроблена та підтримувана Facebook, React використовує компонентний підхід до побудови UI, де кожен компонент може управляти власним станом. Ключовою особливістю React є віртуальний DOM (Document Object Model), який оптимізує оновлення інтерфейсу, порівнюючи віртуальне представлення з реальним DOM і оновлюючи лише змінені частини. React також відомий своєю односпрямованою передачею даних, що робить код більш передбачуваним та простішим для налагодження. Він широко використовується для створення складних, інтерактивних веб-застосунків.

Архітектура ReactJS наведена на рисунку 1.7.

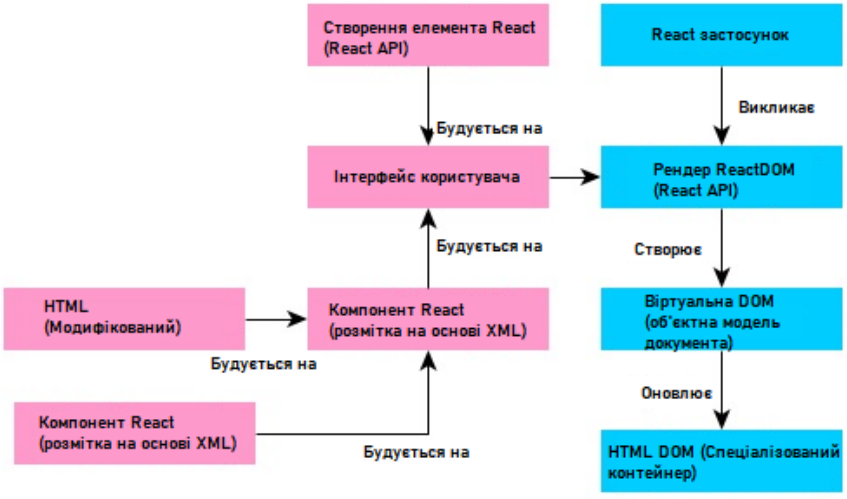

Рисунок 1.7 – Архітектура ReactJS

Node.js – це середовище виконання JavaScript на стороні сервера, побудоване на двигуні V8 від Google Chrome. Воно дозволяє розробникам використовувати JavaScript для написання серверних скриптів, створення веб-серверів та інструментів командного рядка. Node.js використовує неблокуючу, подієво-орієнтовану модель, що робить його легким та ефективним для застосунків, які працюють з даними в реальному часі та інтенсивно використовують введення/виведення. Його пакетний менеджер, npm, є найбільшим у світі реєстром бібліотек з відкритим кодом. Node.js широко використовується для створення масштабованих мережевих застосунків, API та мікросервісів.

Архітектура Node.js продемонстрована на рисунку 1.8.

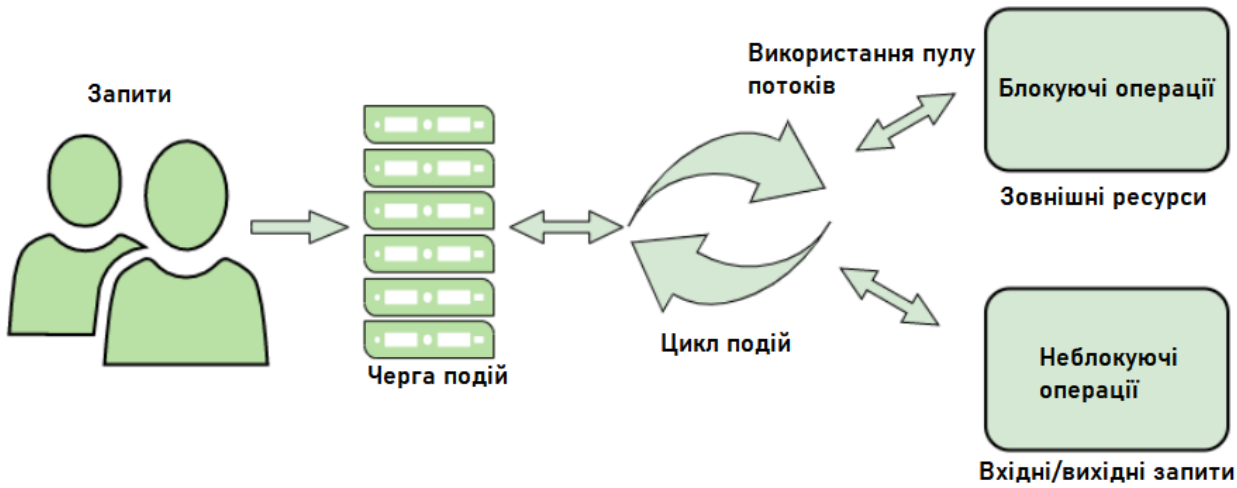

Рисунок 1.8 – Архітектура Node.js

Крім MERN stack, для розробки вебзастосунку можуть бути використані й інші підходи та інструменти. Наприклад, для frontend-частини можна застосувати сучасні JavaScript-фреймворки та бібліотеки, такі як React, Angular або Vue.js, а для backend-частини – мови програмування, такі як Node.js, Python (Django, Flask) або PHP (Laravel, Symfony).

Незалежно від обраного технологічного стеку, важливо також приділити увагу архітектурним підходам, вибору систем контролю версій, інструментів автоматизації, менеджерів пакетів, IDE та інших допоміжних засобів. Крім того,

необхідно забезпечити інтеграцію з зовнішніми сервісами, таким як API сервісів доставки продуктів, API пошуку рецептів та інгредієнтів, платіжні системи, а також приділити увагу забезпеченню безпеки вебзастосунку.

Особливу увагу також слід приділити розробці інтуїтивно зрозумілого та зручного користувацького інтерфейсу, використовуючи сучасні бібліотеки UI-компонентів та проводячи юзабіліті-тестування для вдосконалення UX.

Комплексне застосування описаних підходів та інструментів, зокрема MERN stack, дозволить створити високоякісний, масштабований та безпечний вебзастосунок для підбору рецептів, який задовольнятиме потреби сучасних користувачів.

Отже, на основі проведеного аналізу можна зробити висновок, що застосування MERN stack є найкращим підходом для розробки вебзастосунку для підбору рецептів за інгредієнтами з можливістю придбання необхідних продуктів.

Переваги MERN stack для такого типу застосунку є очевидними. Використання єдиної мови програмування JavaScript як для frontend, так і для backend частини значно спрощує розробку та підтримку проекту. Модульна архітектура на основі мікросервісів забезпечує високу гнучкість та масштабованість системи. Крім того, велика екосистема бібліотек та інструментів, доступних для MERN stack, прискорює процес розробки.

Вибір NoSQL бази даних MongoDB та можливість створення SPA-застосунків також є вагомими перевагами MERN stack для вебзастосунку з функціями пошуку рецептів та замовлення продуктів. MongoDB дозволяє ефективно управляти різноструктурованими даними про рецепти, інгредієнти та користувачів, а SPA-архітектура забезпечує плавну і інтуїтивно зрозумілу взаємодію користувача.

Порівняно з іншими технологічними стеками, MERN stack надає найбільш збалансований та комплексний набір інструментів і підходів для реалізації вебзастосунку з функціоналом, що відповідає сучасним вимогам користувачів. Тому його застосування можна вважати найкращим рішенням для розробки

такого типу веб-сервісу.

1.4 Постановка задач дослідження

Після проведення аналізу предметної області, враховуючи переваги та недоліки застосунків-аналогів та провівши аналіз методів розв’язання задачі, було поставлено такі задачі дослідження:

- здійснити аналіз стану питання та методів розв’язання задачі;
- розробити модель, загальний алгоритм роботи та діаграми станів застосунку;
- розробити метод замовлення продуктів;
- розробити метод пошуку рецептів;
- здійснити програмну реалізацію вебсистеми;
- провести тестування розробленого функціоналу;
- описати системні вимоги вебзастосунку для підбору рецептів за інгредієнтами з можливістю придбання необхідних продуктів.

Технічне завдання на розробку наведено в додатку А.

1.5 Висновки

У першому розділі було проведено аналіз предметної області вебзастосунків для підбору рецептів за інгредієнтами з можливістю придбання необхідних продуктів, виявлено актуальність власної розробки. Проведено порівняння аналогів розроблюваного застосунку, виявлено їхні переваги та недоліки, проведено порівняння з власною розробкою, в результаті якої було доведено доцільність власної розробки. Проведено постановку задач дослідження, сформульовано функціонал власної розробки.

2 РОЗРОБКА МОДЕЛІ, МЕТОДІВ І АЛГОРИТМІВ РОБОТИ ВЕБЗАСТОСУНКУ

2.1 Аналіз даних

Вебзастосунки для пошуку рецептів оперують з різноманітними типами даних для реалізації своєї функціональності. Основними категоріями даних, якими вони працюють, є дані про рецепти, дані про інгредієнти, дані про користувачів, а також дані про замовлення.

Дані про рецепти включають інформацію, таку як назва страви, інгредієнти та їх кількість, інструкції приготування, час та складність приготування, категорія страви, кухня, фотографії готової страви, а також відеоінструкції. Ця інформація є ключовою для пошуку та відображення рецептів користувачам.

Дані про інгредієнти містять назву, опис та характеристики інгредієнтів, їх категоризацію, ціну та одиницю виміру, а також інформацію про наявність в магазинах. Такі дані використовуються для формування списків покупок та замовлення необхідних продуктів.

Дані про користувачів включають персональну інформацію, список улюблених рецептів, історію замовлень, а також відомості про дієтичні обмеження та вподобання. Ця категорія даних дозволяє персоналізувати рекомендації та функціонал застосунку для кожного користувача.

Окремо вебзастосунки обробляють дані про замовлення, такі як інформація про замовлені продукти, дату та час замовлення, статус замовлення, дані про доставку, а також спосіб оплати та суму замовлення. Ці дані є необхідними для реалізації можливості замовлення та доставки продуктів через застосунок.

Ефективне управління та інтеграція цих різноманітних наборів даних є ключовим аспектом у розробці сучасних вебзастосунків для пошуку рецептів та замовлення продуктів харчування. Застосунок "Mealthy", що розробляється, не є виключенням і також використовує та оперує цими категоріями даних для реалізації своїх основних функцій.

2.2 Розробка моделі системи

Для кращого розуміння архітектури розроблюваного застосунка було побудовано модель системи у вигляді діаграми компонентів (рисунок 2.1).

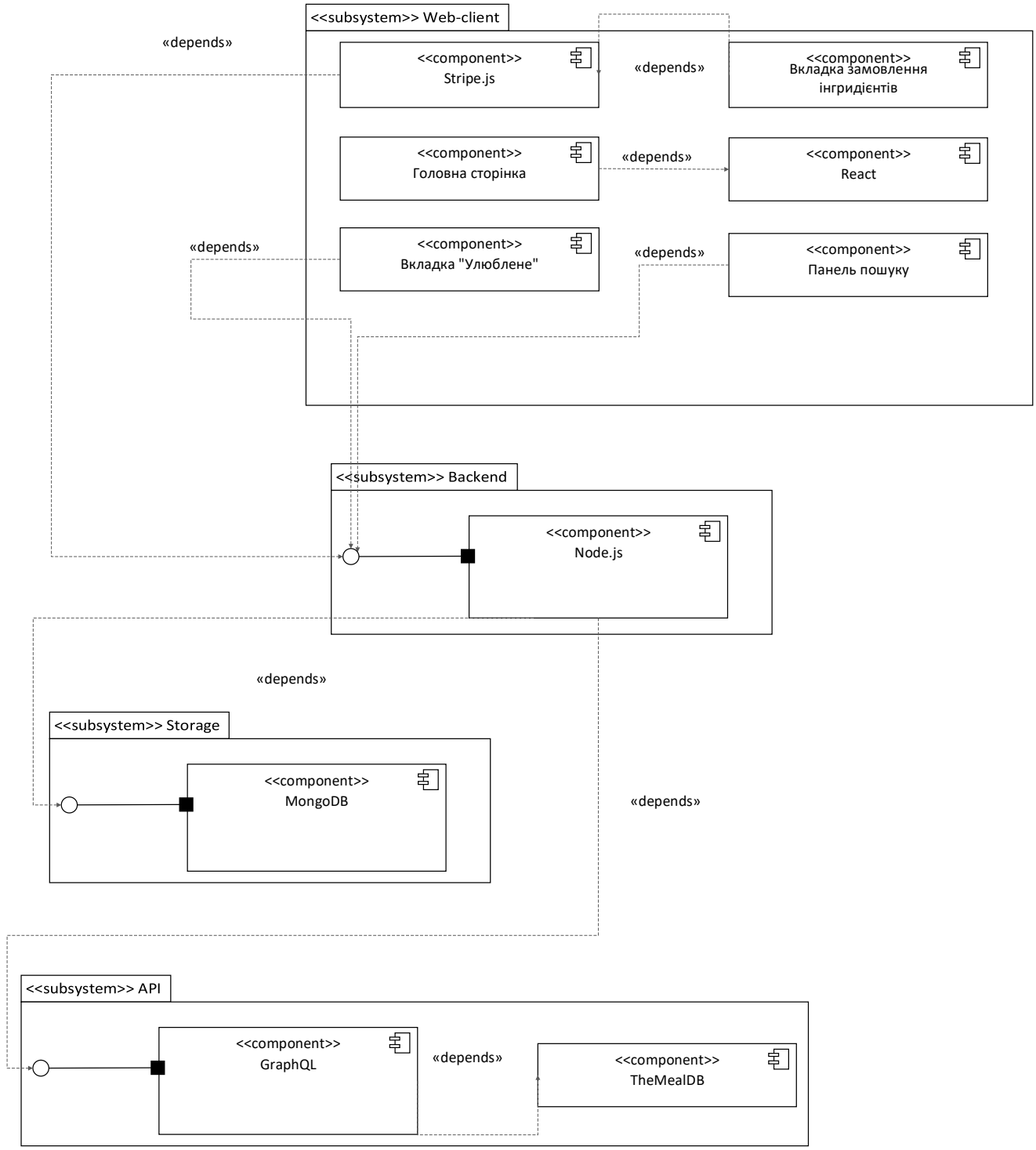

Рисунок 2.1 – Модель системи

Компонент «Web-client» являє собою фронтенд частину застосунку, яка

відповідає за взаємодію з користувачем через веб-браузер. В ній використовується React – популярна JavaScript бібліотеку для створення користувацьких інтерфейсів.

Вкладка замовлення інгредієнтів залежить від бібліотеки Stripe.js для інтеграції з платіжною системою Stripe [11].

Це бібліотека JavaScript від компанії Stripe, яка надає набір програмних інтерфейсів (API) для інтеграції платіжної системи Stripe у веб-застосунки. Вона дозволяє створювати безпечні форми оплати, збирати дані платіжних карток, виконувати процедури 3D Secure, керувати підписками та багато іншого.

Основною перевагою Stripe.js є забезпечення безпечного збору даних платіжної картки. Бібліотека надає захищене поле для введення номера картки, дати закінчення терміну дії, CVC-коду та інших даних. Ці дані шифруються та відправляються безпосередньо на сервери Stripe, оминаючи сервер застосунка. Після збору даних картки Stripe.js повертає одноразовий токен, який можна використовувати для створення платежів або зберігання даних картки на стороні Stripe.

Stripe.js також дозволяє перевіряти дійсність і правильність введених даних платіжної картки перед відправкою на сервер. Крім того, бібліотека підтримує інтеграцію з популярними платіжними системами, такими як Apple Pay та Google Pay, а також надає можливості для локалізації та кастомізації форм оплати відповідно до потреб конкретного застосунка.

У розроблюваній системі Stripe.js використовується для сторінки замовлення та оплати продуктів, які користувач хоче придбати. Він забезпечує формування електронного чека на оплату та переводить користувача на сторінку оплати, де користувач проводить оплату за замовленні інгредієнти.

Головна сторінка відповідає за відображення основного контенту системи. Вона використовує фреймворк React для демонстрації візуальної частини користувачу.

Клієнтська частина системи забезпечується даними з допомогою серверної частини, для якої використовується Node.js.

Node.js – це крос-платформове середовище виконання з відкритим кодом, яке використовує JavaScript для написання серверної частини застосунків. На прикріпленій моделі системи Node.js представлений у вигляді компонента "Node.js".

Ключовою особливістю Node.js є використання неблокуючої архітектури, яка забезпечує ефективну роботу з великою кількістю паралельних з'єднань. Це досягається завдяки подіє-орієнтованій моделі та асинхронному введенню/виведенню, що робить Node.js ідеальним вибором для розробки масштабованих мережевих застосунків з великою пропускною здатністю.

Node.js також пропонує багатий екосистему модулів, що дозволяє розробникам легко інтегрувати різноманітні функції у свої застосунки. Бібліотеки та фреймворки, такі як Express.js для веб-розробки, Socket.IO для реалізації двостороннього зв'язку в реальному часі та Mongoose для роботи з базами даних MongoDB, значно спрощують і прискорюють процес розробки.

Node.js використовується в системі для забезпечення роботи її серверної частини. Він зв’язує клієнтську частину зі сховищем та з API для отримання рецептів.

Сховище представлене базою даних MongoDB – популярною нереляційною базою даних з відкритим кодом.

Використовується для зберігання даних застосунку, таких як користувацька інформація, вміст сайту тощо.

TheGraphQL [12] використовується для зв’язування серверної частини системи з TheMealDB – API, що містить базу даних рецептів.

2.3 Розробка методу пошуку рецептів

Метод пошуку рецептів призначений в першу чергу для полегшення пошуку рецептів тоді, коли є певний набір інгредієнтів, з яких необхідно приготувати страву.

Метод пошуку рецептів складається з таких етапів:

1. Користувач переходить на сторінку пошуку рецептів.

2. Користувач обирає пошук за певними інгредієнтами.
3. Користувач обирає наявні у нього інгредієнти із запропонованого списку.
4. Система формує перелік рецептів, що містять вказані інгредієнти.
5. Користувач обирає один із запропонованих рецептів.
6. Відкривається нове вікно перегляду рецепту.

Блок-схема алгоритму роботи цього методу наведена на рисунку 2.2.

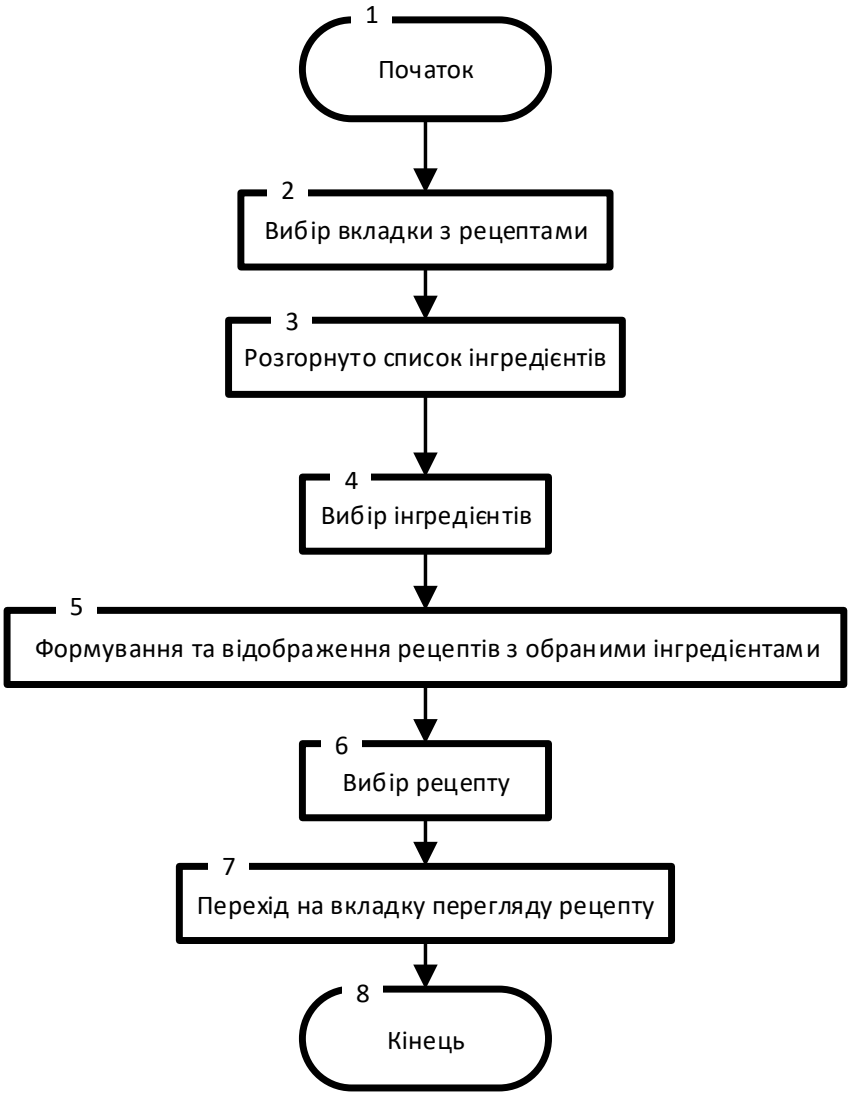

Рисунок 2.2 – Блок-схема алгоритму роботи методу пошуку рецептів

У вікні перегляду рецепту користувач може ознайомитися із рецептом, переглянути відео приготування, а також повний список інгредієнтів, з якого

можна обрати та замовити інгредієнти, яких не вистачає для приготування. Також, користувач може зберегти в «Улюблене» обраний рецепт.

2.4 Розробка методу замовлення продуктів

Можливість замовлення продуктів, яких не вистачає для приготування страви за певним рецептом, є особливою можливістю, яку пропонує розроблюваний вебзастосунок «Mealthy» та якої немає у більшості передових аналогів. Ця можливість може бути дуже корисною для користувачів, адже для них сильно спрощений процес підбору та отримання необхідних інгредієнтів. Окрім того, що існує можливість одразу визначити всі інгредієнти, яких не вистачає, також можна одразу дізнатися вартість, яку буде коштувати та чи інша страва.

Метод замовлення продуктів складається з таких етапів:

1. Користувач обирає рецепт, який він хоче приготувати, із запропонованих варіантів або шляхом пошуку.
2. Вебзастосунок аналізує список інгредієнтів, необхідних для приготування обраного рецепту. Він перевіряє, які з цих інгредієнтів вже є в наявності у користувача.
3. На основі аналізу, застосунок формує список інгредієнтів, яких бракує у користувача для приготування страви.
4. Вебзастосунок пропонує користувачеві замовити відсутні інгредієнти.
5. Користувач може обрати необхідні продукти зі списку, вказати бажану кількість.
6. Користувач вказує зручний час та адресу для доставки замовлених продуктів.
7. Після підтвердження, вебзастосунок передає інформацію про замовлення до сервісів доставки продуктів.
8. Сервіси доставки опрацьовують отримане замовлення, підтверджують його та організовують доставку продуктів на вказану адресу.
9. Замовлені продукти доставляються користувачеві у зазначений час та на

вказану адресу.

Блок-схема цього методу наведена на рисунку 2.3.

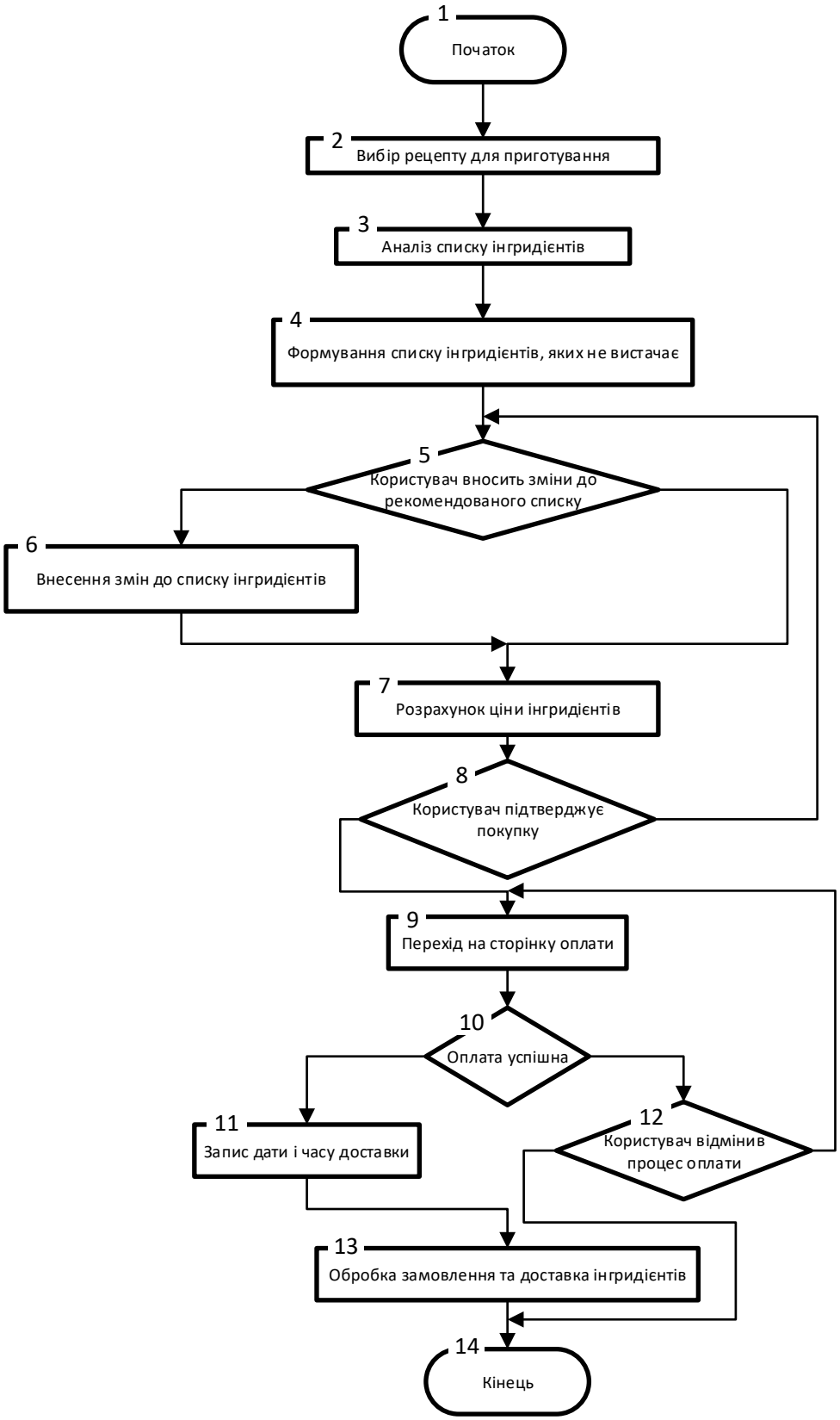

Рисунок 2.3 – Блок-схема методу замовлення продуктів

Таким чином, розроблений метод допомагає користувачам економити час на складанні списку інгредієнтів, що необхідно придбати для приготування рецепту, а також дозволяє замовити необхідні продукти харчування.

2.5 Розробка загального алгоритму роботи застосунку та діаграм станів

Діаграми станів – це графічний спосіб моделювання систем, що складається з різних станів та переходів між ними. Вони широко використовуються для візуалізації поведінки систем у різних умовах.

Діаграми станів корисні для аналізу складних систем, таких як програмне забезпечення, електричні мережі або бізнес-процеси.

Було розроблено діаграми станів, що описують поведінку компонентів системи. Діаграма станів процесу додавання продуктів в кошик наведена на рисунку 2.4.

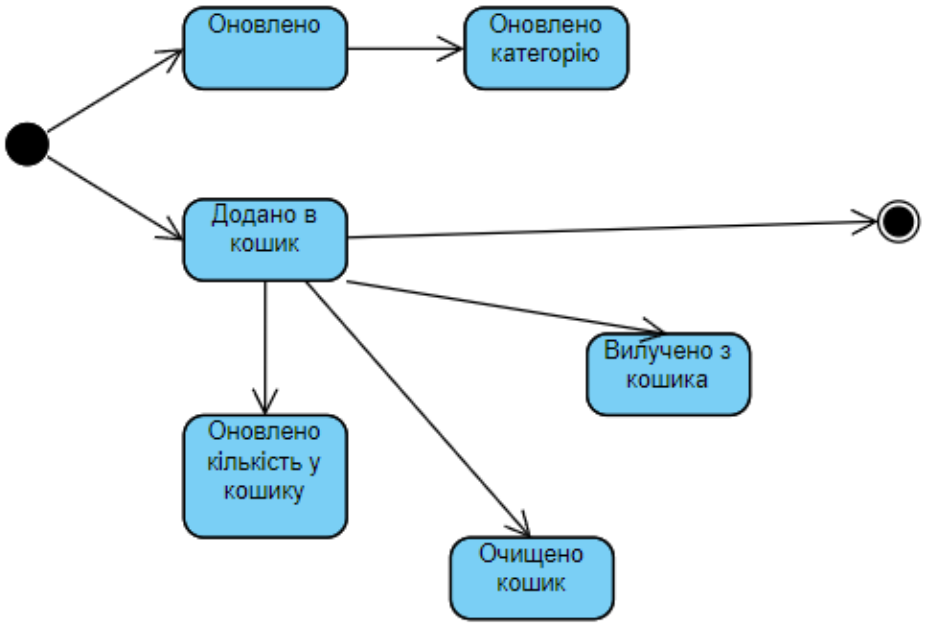

Рисунок 2.4 – Діаграма станів процесу додавання продуктів до кошика

Ця діаграма описує випадки, коли продукти додаються до кошика, вилучаються з нього, коли кошик повністю очищується або оновлюється. Також,

в кошику можна оновити кількість доданого до нього певного продукту.

Діаграма станів процесу пошуку рецептів наведено на рисунку 2.5. В ній описано початковий стан, коли відкрито пошук рецептів, але ще не введено запит, запуск пошуку, перегляд результатів та повернення назад для повторного пошуку.

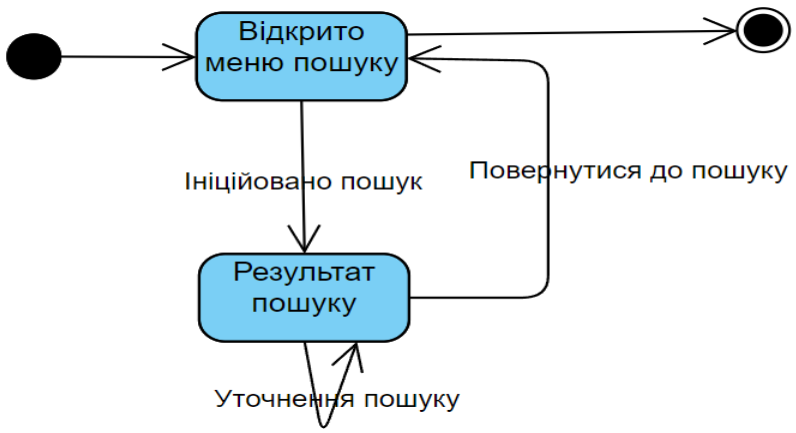

Рисунок 2.5 – Діаграма станів процесу пошуку рецептів

На рисунку 2.6 наведено діаграму станів процесу реєстрації. В ній описано такі кроки, як введення даних, надсилання форми для реєстрації, а також стан успіху, коли дані введені правильно та реєстрація пройшла успішно, та стан помилки, коли введені дані не відповідають вимогам, або виникла інша помилка при реєстрації.

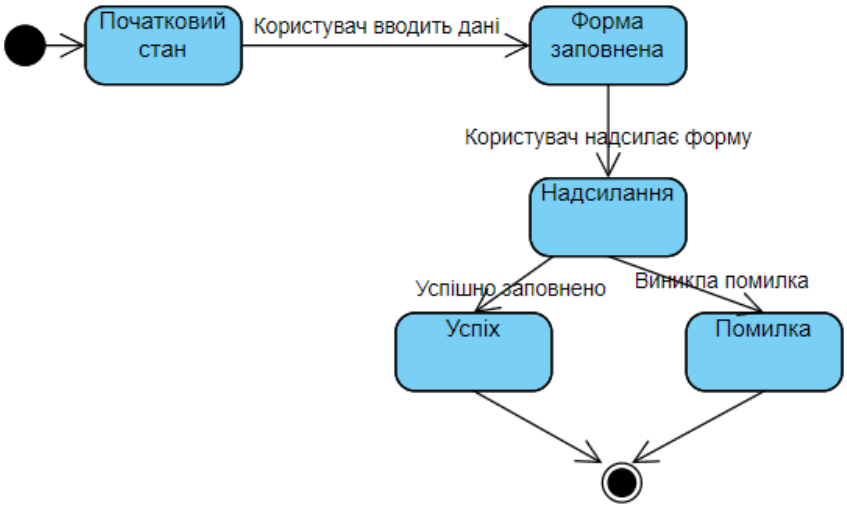

Рисунок 2.6 – Діаграма станів процесу реєстрації

Якщо користувач зареєстрований в системі, тоді він повинен ввести свої дані для входу у свій аккаунт. Якщо він не зареєстрований, тоді йому потрібно пройти процес реєстрації, після чого він зможе користувася усім функціоналом.

Загальний алгоритм роботи застосунку «Mealthy» наведено на рисунку 2.7.

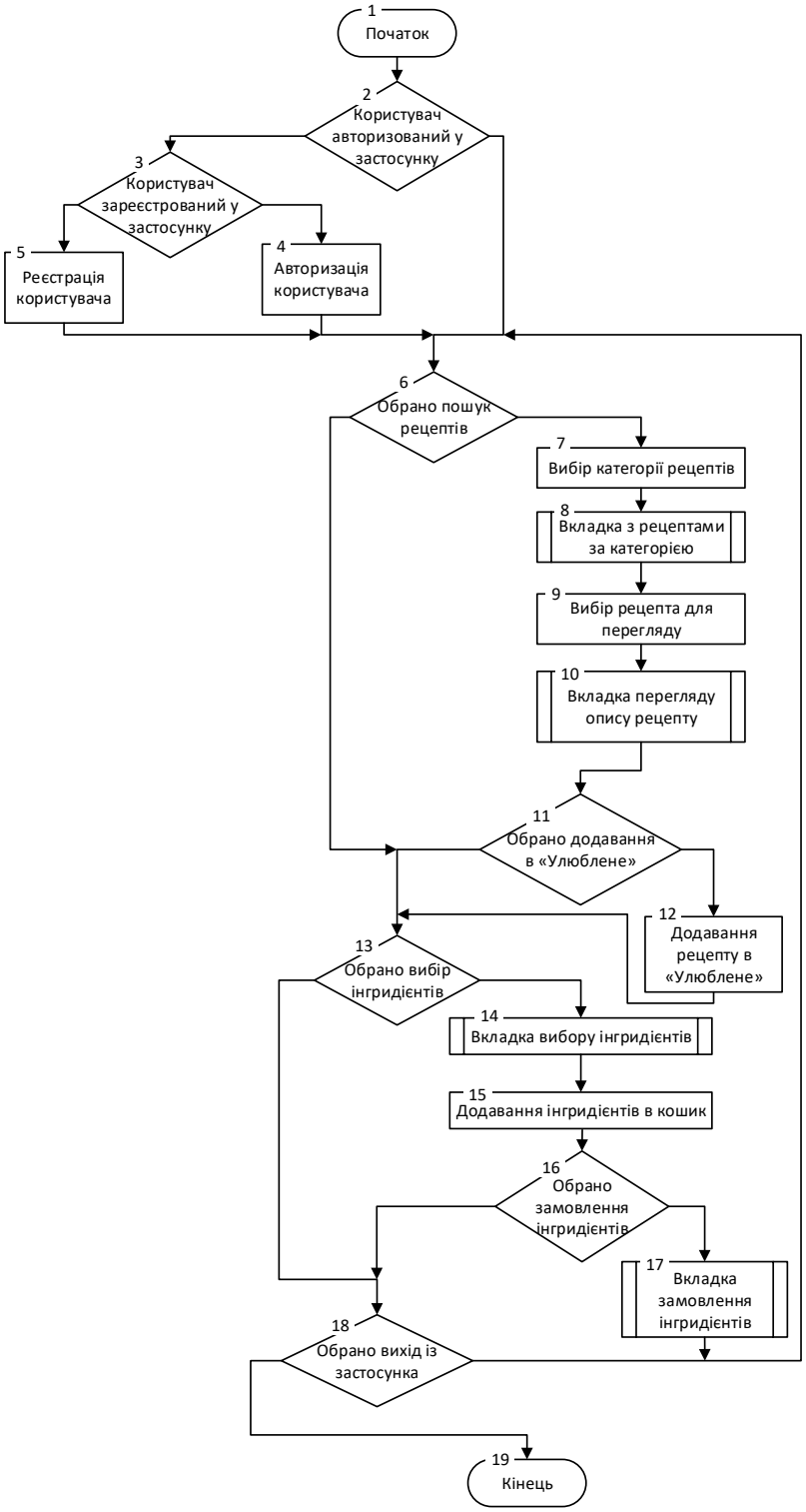

Рисунок 2.7 – Блок-схема загального алгоритму роботи вебзастосунку

При пошуку рецептів, користувач спершу обирає категорію необхідного рецепту, після чого знаходить рецепт, що найбільше йому підходить. Також, він може скористатися меню пошуку рецепту за назвою. Користувач може додати рецепти в «Улюблене». Якщо користувач вирішив обрати перелік інгредієнтів, відкривається відповідна вкладка, з якої користувач також може виконати замовлення необхідних інгредієнтів.

Таким чином, було розроблено загальну блок-схему роботи системи та діаграми станів різних її складових.

2.6 Висновки

У другому розділі, було проведено аналіз даних, які генеруються, використовуються, завантажуються та обробляються при роботі вебзастосунку для підбору рецептів за інгредієнтами з можливістю придбання необхідних продуктів. Було розроблено модель вебзастосунку, метод пошуку рецептів, метод замовлення продуктів, загальний алгоритм роботи та діаграми станів.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного забезпечення

JavaScript [13] – це високорівнева, інтерпретована мова програмування, яка є однією з основних технологій вебконтенту разом з HTML та CSS. JavaScript використовується для створення інтерактивних вебсторінок та вебап вебзастосунків, а також для розробки серверних застосунків за допомогою Node.js.

Однією з основних переваг JavaScript є його простота та легкість у вивченні. JavaScript має синтаксис, схожий на синтаксис мов програмування C та Java, що полегшує його вивчення розробникам, які вже знайомі з цими мовами. Крім того, JavaScript має велику кількість бібліотек та фреймворків, таких як jQuery, React та Angular, які дозволяють розробникам швидко створювати складні вебзастосунки.

JavaScript є інтерпретованою мовою, що означає, що код JavaScript виконується безпосередньо в веббраузері, без необхідності компіляції. Це дозволяє розробникам швидко тестувати та змінювати код, а також забезпечує високий рівень взаємодії з користувачем. JavaScript також підтримує об'єктно-орієнтоване програмування, що дозволяє створювати складні програмні структури та робить код більш зрозумілим та легким у підтримці.

Однією з ключових особливостей JavaScript є його здатність взаємодіяти з вебсторінкою та реагувати на дії користувача. JavaScript може використовуватися для створення анімацій, перемикання вкладок, валідації форм та багатьох інших інтерактивних елементів на вебсторінці. Крім того, JavaScript може взаємодіяти з HTML та CSS, що дозволяє змінювати зовнішній вигляд та розташування елементів на сторінці в реальному часі.

JavaScript також може використовуватися для роботи з даними на клієнтській стороні. JavaScript підтримує роботу з JSON (JavaScript Object Notation), що є легким та зручним форматом для обміну даними між клієнтом та

сервером. JavaScript також підтримує роботу з локальним сховищем даних, що дозволяє зберігати дані на клієнтській стороні та використовувати їх навіть при відсутності з'єднання з сервером.

Використання JavaScript для розробки серверних застосунків за допомогою Node.js дозволяє створювати повноцінні вебзастосунки, які працюють на одній мові програмування як на клієнтській, так і на серверній стороні. Це дозволяє розробникам легше взаємодіяти між клієнтом та сервером та створювати більш ефективні та швидкодіючі вебаплікації.

JavaScript є потужною та гнучкою мовою програмування, яка використовується для створення інтерактивних вебсторінок та вебзастосунків. JavaScript має простий синтаксис, велику кількість бібліотек та фреймворків, а також підтримує об'єктно-орієнтоване програмування. JavaScript є невід'ємною частиною сучасного вебпрограмування та продовжує розвиватися та вдосконалюватися.

TypeScript [14] – це відкрита мова програмування, розроблена компанією Microsoft, яка є надбудовою над мовою JavaScript. TypeScript додає статичну типізацію, класи, інтерфейси та інші особливості, відсутні в JavaScript, що робить його більш підходящим для розробки великих та складних програмних проєктів.

Однією з основних переваг TypeScript є його здатність працювати з існуючими JavaScript-бібліотеками та фреймворками. TypeScript-код компілюється в JavaScript, що дозволяє використовувати його в будь-якому місці, де підтримується JavaScript. Крім того, TypeScript має вбудовані типи для багатьох популярних JavaScript-бібліотек, таких як jQuery, React та Angular, що полегшує їх використання в TypeScript-проєктах.

TypeScript також має статичну типізацію, що дозволяє визначати типи змінних та параметрів функцій. Це допомагає уникнути помилок, пов'язаних з неправильним використанням типів даних, та робить код більш зрозумілим та легким у підтримці. TypeScript також має перевірку типів під час компіляції, що дозволяє виявити помилки в коді до його виконання.

TypeScript підтримує об'єктно-орієнтоване програмування, включаючи класи, інтерфейси, наслідування та поліморфізм. Це дозволяє створювати складні програмні структури та робить код більш модульним та легким у підтримці. TypeScript також підтримує модулі, що дозволяє розбивати код на окремі файли та легко використовувати їх в інших проєктах.

TypeScript має потужну систему інтерфейсів, що дозволяє визначати типи даних, які можуть бути використані в коді. Це дозволяє створювати більш гнучкі та переносимі програми, а також робить код більш зрозумілим та легким у підтримці. TypeScript також має просторовні імена, що дозволяє уникнути конфліктів імен між різними бібліотеками та модулями.

TypeScript має велику кількість інструментів та бібліотек, таких як Angular, Vue.js, React, Ionic, NativeScript, що дозволяють розробникам швидко створювати складні вебзастосунки. TypeScript також має вбудовану підтримку для ES6, ES7 та ES8, що дозволяє використовувати нові особливості JavaScript в TypeScript-проєктах.

Загалом, TypeScript є потужною та гнучкою мовою програмування, яка дозволяє розробникам створювати великі та складні програмні проєкти, використовуючи статичну типізацію, класи, інтерфейси та інші особливості. TypeScript працює з існуючими JavaScript-бібліотеками та фреймворками, має велику кількість інструментів та бібліотек, та підтримує нові особливості JavaScript. TypeScript є відмінним вибором для розробників, які шукають більш надійний та ефективний спосіб розробки вебзастосунків.

PHP (Hypertext Preprocessor) [15] – це серверна мова програмування, яка використовується для створення динамічних вебсторінок та вебзастосунків. PHP є вільним та відкритим програмним забезпеченням, яке розповсюджується під ліцензією PHP License. PHP підтримує багато платформ, включаючи Windows, Linux, macOS та Unix, та може бути використаний з багатьма серверами, включаючи Apache, Nginx та Microsoft IIS.

PHP є інтерпретованою мовою, що означає, що код PHP виконується безпосередньо на сервері, без необхідності компіляції. Це дозволяє розробникам

швидко створювати та тестувати вебзастосунки. PHP також підтримує об'єктно-орієнтоване програмування, що дозволяє створювати складні програмні структури та робить код більш модульним та легким у підтримці.

PHP має велику кількість бібліотек та фреймворків, таких як Laravel, Symfony, CodeIgniter, Yii, що дозволяють розробникам швидко створювати складні вебзастосунки. PHP також підтримує роботу з багатьма базами даних, включаючи MySQL, PostgreSQL, Oracle, SQLite, що дозволяє створювати вебаплікації, які працюють з великими обсягами даних.

PHP має вбудовану підтримку для HTML, CSS та JavaScript, що дозволяє легко створювати динамічні вебсторінки. PHP також підтримує роботу з файлами, що дозволяє завантажувати, зберігати та видаляти файли на сервері. PHP також має вбудовану підтримку для сесій та cookie, що дозволяє зберігати інформацію про користувачів між запитами.

PHP має велику спільноту користувачів та розробників, що забезпечує велику кількість ресурсів для вивчення та підтримки. PHP також має велику кількість готових рішень, таких як CMS (Content Management System), що дозволяють створювати вебсайти без необхідності написання коду з нуля.

PHP має деякі недоліки, такі як низька продуктивність порівняно з іншими мовами програмування, такі як Java або C#, та недостатня підтримка сучасних підходів до розробки, таких як функціональне програмування. Проте, PHP продовжує бути популярною мовою програмування для веброзробки та використовується багатьма великими компаніями, включаючи Facebook, Wikipedia та WordPress.

Загалом, PHP є потужною та гнучкою мовою програмування, яка використовується для створення динамічних вебсторінок та вебзастосунків. PHP має велику кількість бібліотек та фреймворків, підтримує роботу з багатьма базами даних, має велику спільноту користувачів та розробників, та підтримує нові особливості веброзробки. PHP є відмінним вибором для розробників, які шукають надійний та ефективний спосіб створення вебзастосунків.

JavaScript є найкращим вибором серед цих мов для багатьох вебпроєктів,

оскільки він має ряд переваг, які роблять його ідеальним для розробки вебзастосунків.

По-перше, JavaScript є клієнтською мовою програмування, що означає, що код JavaScript виконується безпосередньо в браузері користувача. Це дозволяє створювати інтерактивні вебсторінки та вебзастосунки, які можуть динамічно змінюватися без необхідності перезавантаження сторінки. Крім того, JavaScript дозволяє створювати вебзастосунки, які працюють в автономному режимі, без необхідності підключення до мережі.

По-друге, JavaScript має велику кількість бібліотек та фреймворків, таких як jQuery, React, Angular, Vue.js, що дозволяють розробникам швидко створювати складні вебзастосунки. Ці бібліотеки та фреймворки надають готові рішення для багатьох завдань веброзробки, таких як робота з DOM, AJAX-запити, управління станом та маршрутизація.

По-третє, JavaScript має велику спільноту користувачів та розробників, що забезпечує велику кількість ресурсів для вивчення та підтримки. JavaScript є однією з найпопулярніших мов програмування в світі, і багато компаній та розробників використовують його для створення вебзастосунки.

По-четверте, JavaScript є мовою програмування, яка підтримується всіма сучасними веб-браузерами, що означає, що вебзастосунки, створені на JavaScript, будуть працювати на будь-якому пристрої та в будь-якому браузері.

Щодо середовища розробки, то Visual Studio Code [16] є одним з найкращих середовищ розробки вебзастосунків через його широкий спектр функцій, гнучкість і зручність використання. По-перше, VS Code має вбудовану підтримку багатьох мов програмування, включаючи JavaScript, TypeScript, HTML, CSS, Python, Java, C++ і багато інших. Це означає, що розробники можуть використовувати одне середовище розробки для всіх своїх веб-проектів, незалежно від мови програмування, яку вони використовують.

По-друге, VS Code має дуже потужні інструменти для налагодження коду. Вбудований дебагер дозволяє розробникам встановлювати точки зупинки, переглядати змінні, стек викликів і навіть виконувати код крок за кроком.

3.2 Розробка структури бази даних

Розробка структури бази даних є ключовим етапом у процесі створення програмного забезпечення. Правильно спроектована база даних забезпечує ефективне зберігання, організацію та доступ до даних, що відповідає вимогам проєкту.

Першим кроком у розробці структури бази даних є аналіз вимог до даних. Це включає вивчення функціональних та нефункціональних вимог, а також потреб користувачів у доступі до інформації.

Другим кроком є проектування схеми бази даних. На цьому етапі визначається структура таблиць, включаючи поля та типи даних, а також зв'язки між таблицями. Важливо враховувати нормалізацію даних для забезпечення їхньої цілісності та уникнення аномалій.

Третій крок – реалізація бази даних. Це включає створення таблиць, індексів, обмежень цілісності даних та інших об'єктів бази даних за допомогою мови запитів (наприклад, SQL) або інструментів адміністрування баз даних.

Для роботи бази даних, було створено таблиці таблиці Category, User, Comment, Product, Order, OrderProduct, Recipe, UserSavedRecipes.

Таблиця Category містить записи про категорії продуктів та рецептів та складається з таких полів:

- _id – унікальний ідентифікатор категорії. Тип даних – автоматично зростаючий номер.
- name – назва категорії. Тип даних – VARCHAR(255).

Таблиця User містить дані про користувачів системи та складається з таких полів:

- _id – унікальний ідентифікатор користувача. Тип даних: SERIAL (автоматично зростаючий номер).
- username – ім'я користувача. Тип даних: VARCHAR(255). Поле є обов'язковим.
- Email – тлектронна адреса користувача. Тип даних: VARCHAR(255). Поле є обов'язковим.

- password – пароль користувача. Тип даних: VARCHAR(255). Поле є обов'язковим.

Таблиця Post містить дані про пости або статті на кулінарну тематику, що опубліковані в системі та складається з таких полів:

- _id – унікальний ідентифікатор поста. Тип даних: SERIAL (автоматично зростаючий номер).
- postText – текст поста. Тип даних: TEXT. Поле є обов'язковим.
- postAuthor – ідентифікатор автора поста. Тип даних: INT. Поле є обов'язковим. Зовнішній ключ, що посилається на таблицю User.
- createdAt – дата і час створення поста. Тип даних: TIMESTAMP. Поле є обов'язковим.

Таблиця Comment містить записи коментарів, що користувачі залишили в системі та складається з таких полів:

- _id – унікальний ідентифікатор коментаря. Тип даних: SERIAL (автоматично зростаючий номер).
- commentText – текст коментаря. Тип даних: TEXT. Поле є обов'язковим.
- commentAuthor – ідентифікатор автора коментаря. Тип даних: INT. Поле є обов'язковим. Зовнішній ключ, що посилається на таблицю User.
- createdAt – дата і час створення коментаря. Тип даних: TIMESTAMP. Поле є обов'язковим.
- postId – ідентифікатор поста, до якого належить коментар. Тип даних: INT. Поле є обов'язковим. Зовнішній ключ, що посилається на таблицю Post.

Таблиця Product містить записи про продукти харчування та складається з таких полів:

- _id – унікальний ідентифікатор продукту. Тип даних: SERIAL (автоматично зростаючий номер).
- name – назва продукту. Тип даних: VARCHAR(255).
- description – опис продукту. Тип даних: TEXT.
- image – URL зображення продукту. Тип даних: VARCHAR(255).
- quantity – кількість продукту. Тип даних: INT.

- price – ціна продукту. Тип даних: FLOAT.
- categoryId – ідентифікатор категорії, до якої належить продукт. Тип даних: INT. Зовнішній ключ, що посилається на таблицю Category.

Таблиця Order містить записи про замовлення продуктів, що здійснені користувачами та містить такі поля:

- _id – унікальний ідентифікатор замовлення. Тип даних: SERIAL (автоматично зростаючий номер).
- purchaseDate – дата і час покупки. Тип даних: TIMESTAMP.
- userId – ідентифікатор користувача, який зробив замовлення. Тип даних: INT. Зовнішній ключ, що посилається на таблицю User.

Таблиця OrderProduct є проміжною таблицею, що зв’язує записи із замовленнями з продуктами та складається з таких полів:

- orderId – ідентифікатор замовлення. Тип даних: INT. Зовнішній ключ, що посилається на таблицю Order. Поле є частиною первинного ключа.
- productId – ідентифікатор продукту. Тип даних: INT. Зовнішній ключ, що посилається на таблицю Product. Поле є частиною первинного ключа.

Таблиця Recipe містить в собі записи рецептів та складається з таких полів:

- _id – унікальний ідентифікатор рецепта. Тип даних: SERIAL (автоматично зростаючий номер).
- idMeal – унікальний ідентифікатор страви. Тип даних: VARCHAR(255). Поле є обов'язковим.
- strMeal – назва страви. Тип даних: VARCHAR(255).
- strCategory – категорія страви. Тип даних: VARCHAR(255).
- strArea – регіон або країна походження страви. Тип даних: VARCHAR(255).
- strInstructions – інструкції по приготуванню страви. Тип даних: TEXT.
- strMealThumb – URL зображення страви. Тип даних: VARCHAR(255).
- strTags – теги страви. Тип даних: TEXT.
- strYoutube – URL відео з рецептом на YouTube. Тип даних: VARCHAR(255).

• strIngredients – інгредієнти страви, збережені у форматі тексту. Тип даних: TEXT.

• strMeasures – кількість інгредієнтів, збережена у форматі тексту. Тип даних: TEXT.

Таблиця UserSavedRecipes містить збережені користувачем рецепти та складається з таких полів:

• userId – ідентифікатор користувача, який зберіг рецепт. Тип даних: INT. Зовнішній ключ, що посилається на таблицю User. Поле є частиною первинного ключа.

• recipeId – ідентифікатор рецепта. Тип даних: INT. Зовнішній ключ, що посилається на таблицю Recipe. Поле є частиною первинного ключа.

Схема бази даних наведена на рисунку 3.1.

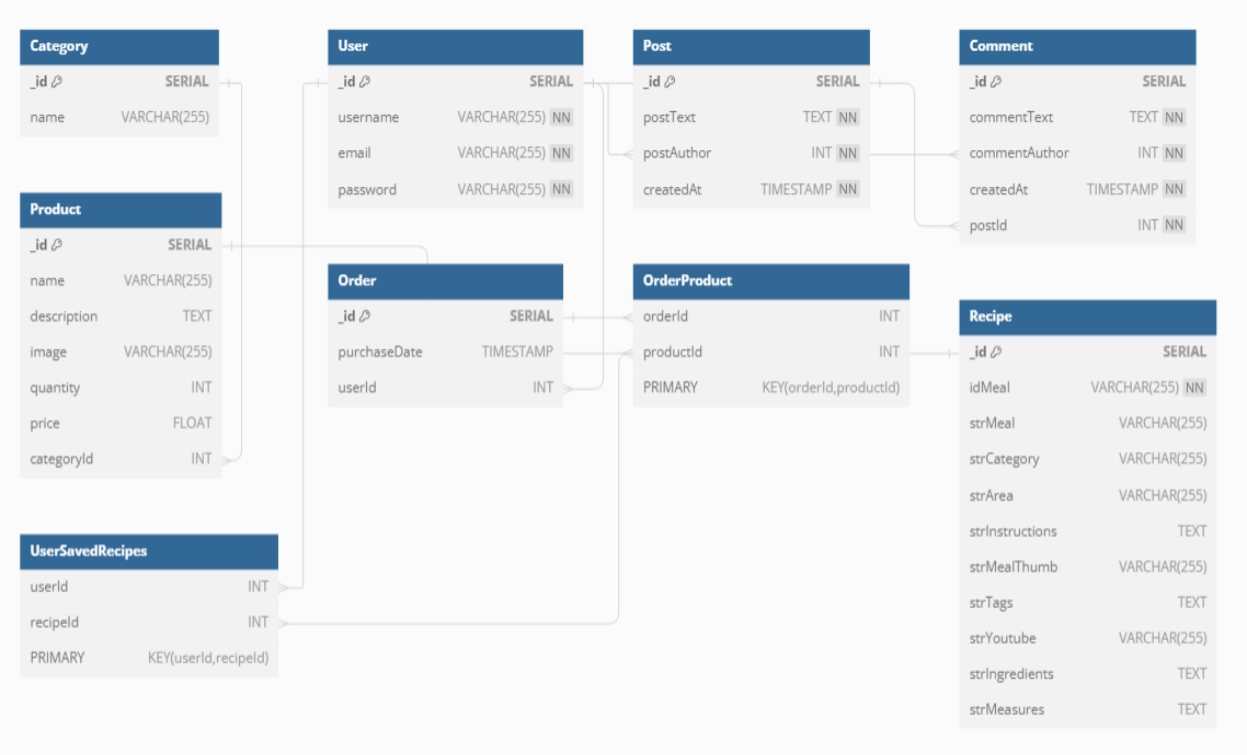

Рисунок 3.1 – Схема бази даних

Отже, було розроблено схему бази даних, в якій зберігатимуться дані про користувачів, рецепти, збережені рецепти, замовлені продукти, коментарі. Проведено детальний опис усіх таблиць та полів бази даних.

3.3 Розробка модуля авторизації та реєстрації користувача

Процес реєстрації користувача є базовим для отримання ним доступу до функціоналу системи. Реєстрація аккаунту користувача дозволяє прив’язати дані користувача до збережених ним в аккаунті даних в базі даних, що згодом дозволить отримати до них доступ при авторизації користувача в системі. При вході в аккаунт з будь-якого пристрою, дані будуть відображені користувачу, адже вони зберігаються на віддаленому сервері, а не на пристрої користувача.

Було розроблено функціонал для авторизації та реєстрації користувача. Він об’єднаний у сервіс AuthService та складається з кількох функцій, що покривають всі випадки використання системи, що стосуються реєстрації та авторизації.

Блок-схема алгоритму роботи функції перевірки токена авторизації наведена на рисунку 3.2.

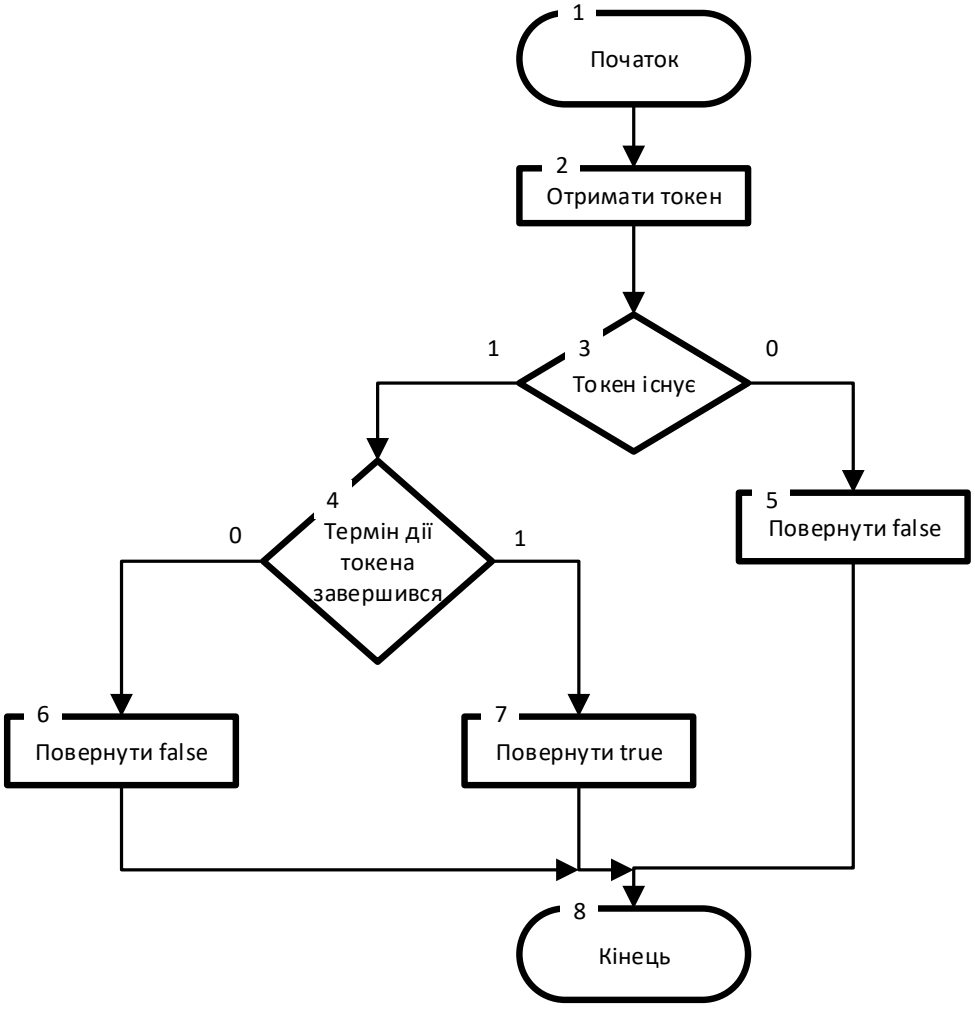

Рисунок 3.2 – Блок-схема алгоритму роботи функції перевірки токена авторизації

Спершу проводиться перевірка, чи вийшов термін дії токена авторизації користувача. Якщо токен не виявлено, тоді повертається значення false, адже в такому випадку неможливо перевірити, чи вийшов термін токена, якого не існує, а також те, що користувач ніколи не був авторизований у системі.

Якщо токен існує та його термін дії не завершився, система повертає false. Якщо він завершився, тоді повертається true і в такому випадку генерується новий токен авторизації.

Наступний крок – отримання профілю користувача та його авторизація. Блок-схема алгоритму роботи функції для отримання профілю користувача та його токена авторизації наведена на рисунку 3.3.

Рисунок 3.3 – Блок-схема алгоритму роботи функції для отримання профілю користувача

Після цього, проводиться авторизація користувача в профіль, збереження створеного токена авторизації в базу даних, перенаправлення користувача на

головну сторінку.

Було розроблено графічну схему сторінки реєстрації. Вона наведена на рисунку 3.4.

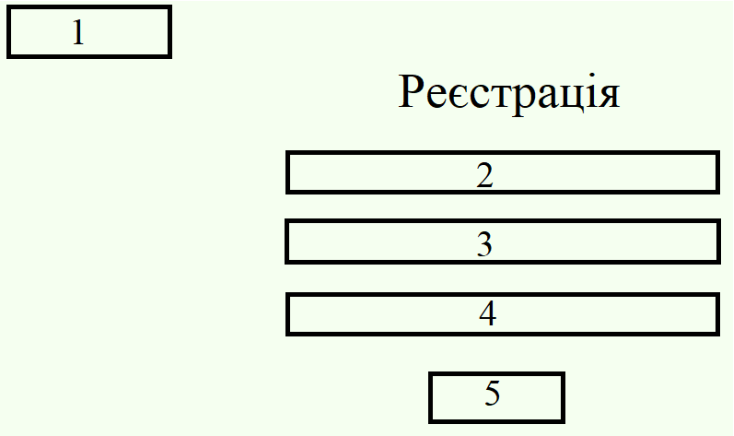

Рисунок 3.4 – Графічна схема сторінки реєстрації

Елементи сторінки реєстрації:

1. Логотип застосунку.
2. Поле імені користувача.
3. Поле електронної пошти.
4. Поле для паролю.
5. Кнопка підтвердження реєстрації.

Також, було розроблено графічну схему сторінки авторизації, яка подана на рисунку 3.5. Вона дозволяє користувачу авторизуватися в застосунок.

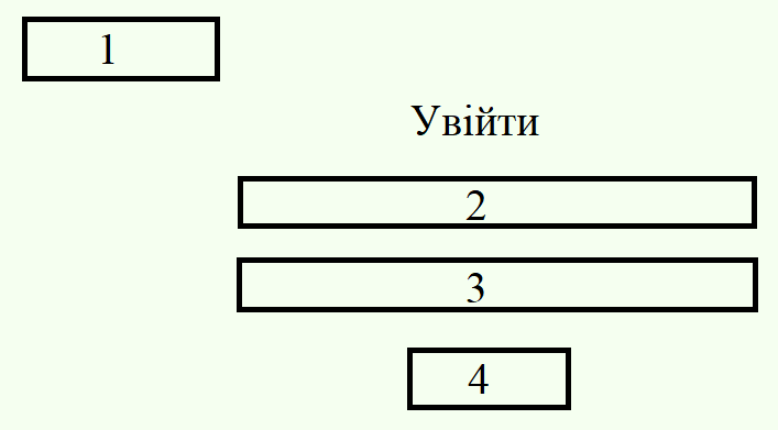

Рисунок 3.5 – Графічна схема сторінки авторизації

Елементи сторінки авторизації:

1. Логотип.
2. Поле електронної пошти.
3. Поле для паролю.
4. Кнопка «Увійти».

Для кращого розуміння можливих сценаріїв роботи та зв’язку між собою сторінок авторизації та реєстрації, було побудовано блок-схему, яка наведена на рисунку 3.6.

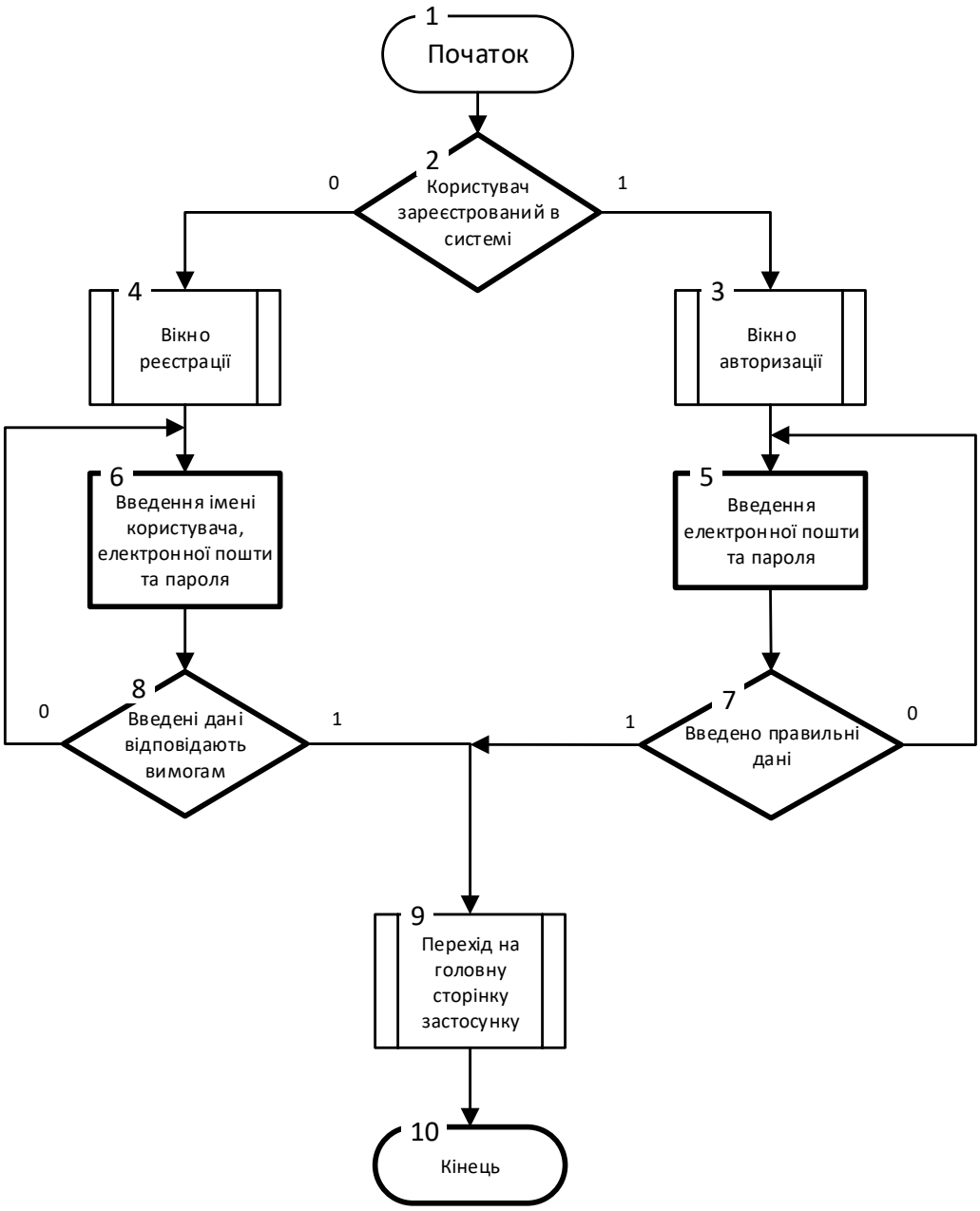

Рисунок 3.6 – Блок-схема алгоритму процесу реєстрації та авторизації

3.4 Розробка головної вкладки та модуля перегляду рецептів вебзастосунку

Головна вкладка вебзастосунку є основною та запускається при переході на веб-сайт застосунку. Вона надає доступ до інших вкладок застосунку. Найбільше місця на головній сторінці займає перелік рецептів, які відбираються системою випадковим чином. При кожному оновленні сторінки цей перелік оновлюється.

Також, у верхній частині робочої області є пошуковик рецептів за повною назвою або ж ключовими словами, а також за назвою інгредієнту, який міститься у рецептах. Окрім цього, додано 2 випадаючих списки, які оновлюють перелік запропованих рецепти на головній сторінці. Можна обирати серед певного переліку національних кухонь, а також за певними категоріями страв.

Структурна схема інтерфейсу головної вкладки вебзастосунку «Mealthy» наведено на рисунку 3.7.

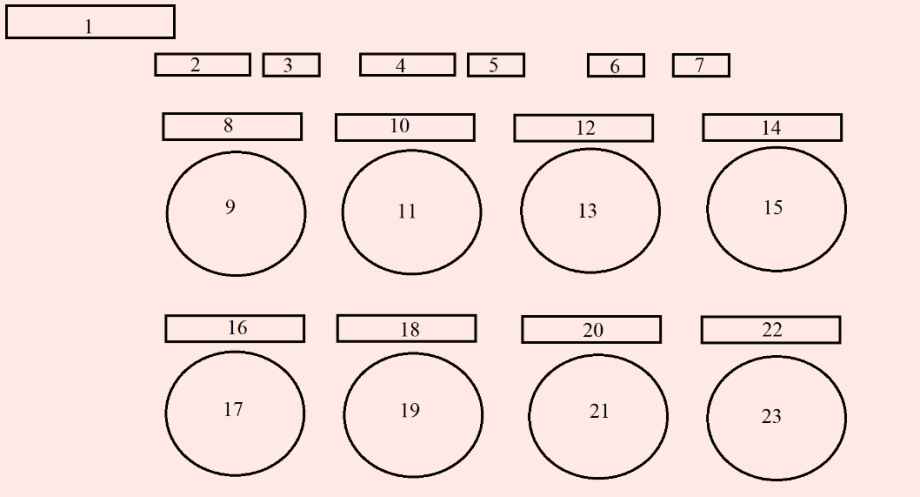

Рисунок 3.7 – Структурна схема інтерфейсу головної вкладки вебзастосунку

Елементи головної вкладки:

1. Логотип.
2. Поле для пошуку страв за ключовими словами.
3. Кнопка пошуку рецептів за ключовими словами.
4. Поле для пошуку страв за інгредієнтами.
5. Кнопка пошуку рецептів за ключовими словами.

6. Випадаючий список категорій страв.
7. Випадаючий список національних кухонь.

8, 10, 12, 14, 16, 18, 20, 22 – назви страв на головній сторінці.

9, 11, 13, 15, 17, 19, 21, 23 – фото страв.

Кількість страв, що продемонстровані на головній сторінці, може відрізнятися. Це залежить від обсягу категорій, що обираються користувачем. За замовчуванням, при початковому запуску головної сторінки відображається 10 випадкових рецептів.

При пошуку за назвою, список оновлюється та відображає ті рецепти, які містять введені в пошуку слова у назві.

При пошуку за інгредієнтами, відображаються ті рецепти, в яких міститься вказаний інгредієнт.

Для вибору рецептів за категорією та за національною кухнею розроблено випадаючі списки.

Випадаючі списки, також відомі як dropdown menus, є поширеним елементом інтерфейсу користувача в вебзастосунках. Вони дозволяють користувачам обирати один або кілька варіантів з набору опцій, що з'являються в розкривному меню. Це ефективний спосіб зберігати простір на сторінці та зменшувати візуальне перевантаження, оскільки опції приховані, поки користувач не взаємодіє з випадаючим списком.

Випадаючі списки можуть бути реалізовані за допомогою різних веб-технологій, таких як HTML, CSS та JavaScript. У HTML зазвичай використовується елемент <select>, щоб створити основу випадаючого списку, в якому елементи <option> представляють окремі опції. CSS використовується для стилізації вигляду випадаючого списку, наприклад, кольору, розміру шрифту та візуальних ефектів при наведенні курсора або виборі опції. JavaScript може бути використано для додавання інтерактивності та динамічної поведінки, наприклад, автоматичного заповнення опцій на основі введених користувачем даних або зміни вмісту сторінки відповідно до обраної опції.

Основною перевагою використання випадаючих списків є те, що вони

дозволяють користувачам швидко та легко обирати потрібну опцію з великого набору варіантів. Це особливо корисно, коли є обмежений простір на сторінці або коли список опцій дуже довгий. Крім того, випадаючі списки можуть бути використані для групування пов'язаних опцій, що полегшує навігацію та робить інтерфейс користувача більш інтуїтивно зрозумілим.

Коли користувач розкриває цікавий йому випадаючий список та обирає необхідну йому категорію, список страв оновлюється аналогічно, як при пошуку за інгредієнтами чи ключовими словами.

Список страв дозволяє переглянути демонстраційне фото страви та його назву. При натисканні на страву, відкривається нова вкладка з її описом, рецептом та переліком інгредієнтів.

Розроблено блок-схему алгоритму відображення рецепту, яка наведена на рисунку 3.8.

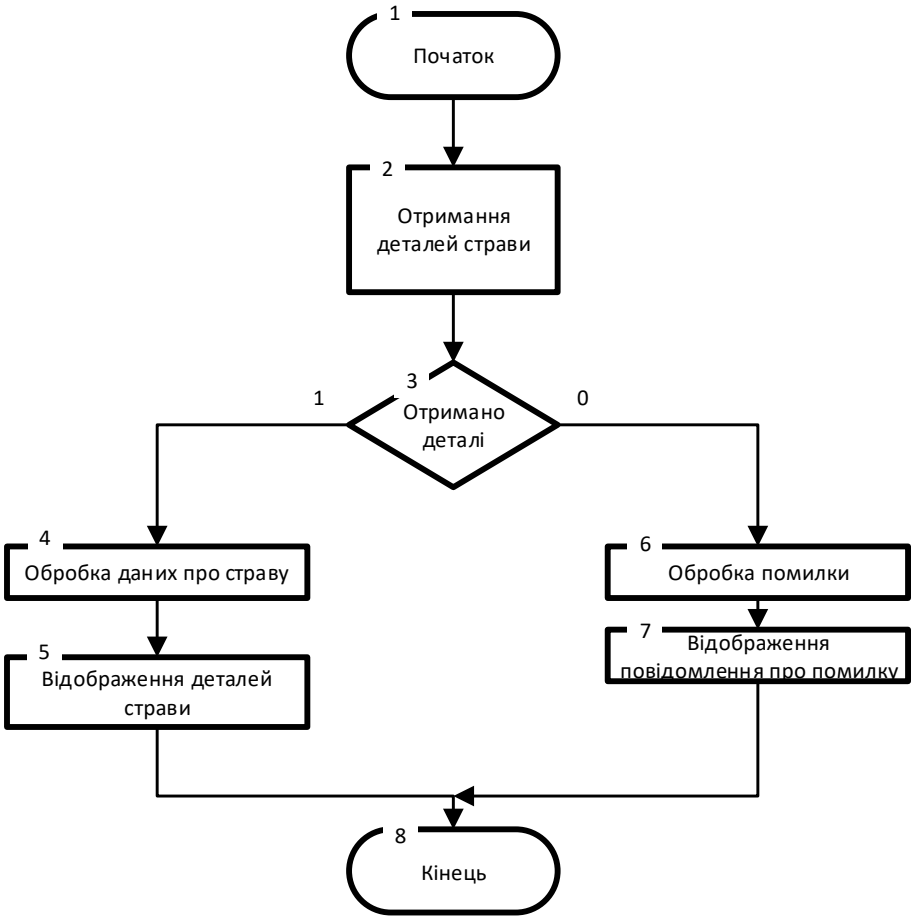

Рисунок 3.8 – Блок-схема алгоритму відображення рецепту

Структурна схема цієї вкладки наведена на рисунку 3.9.

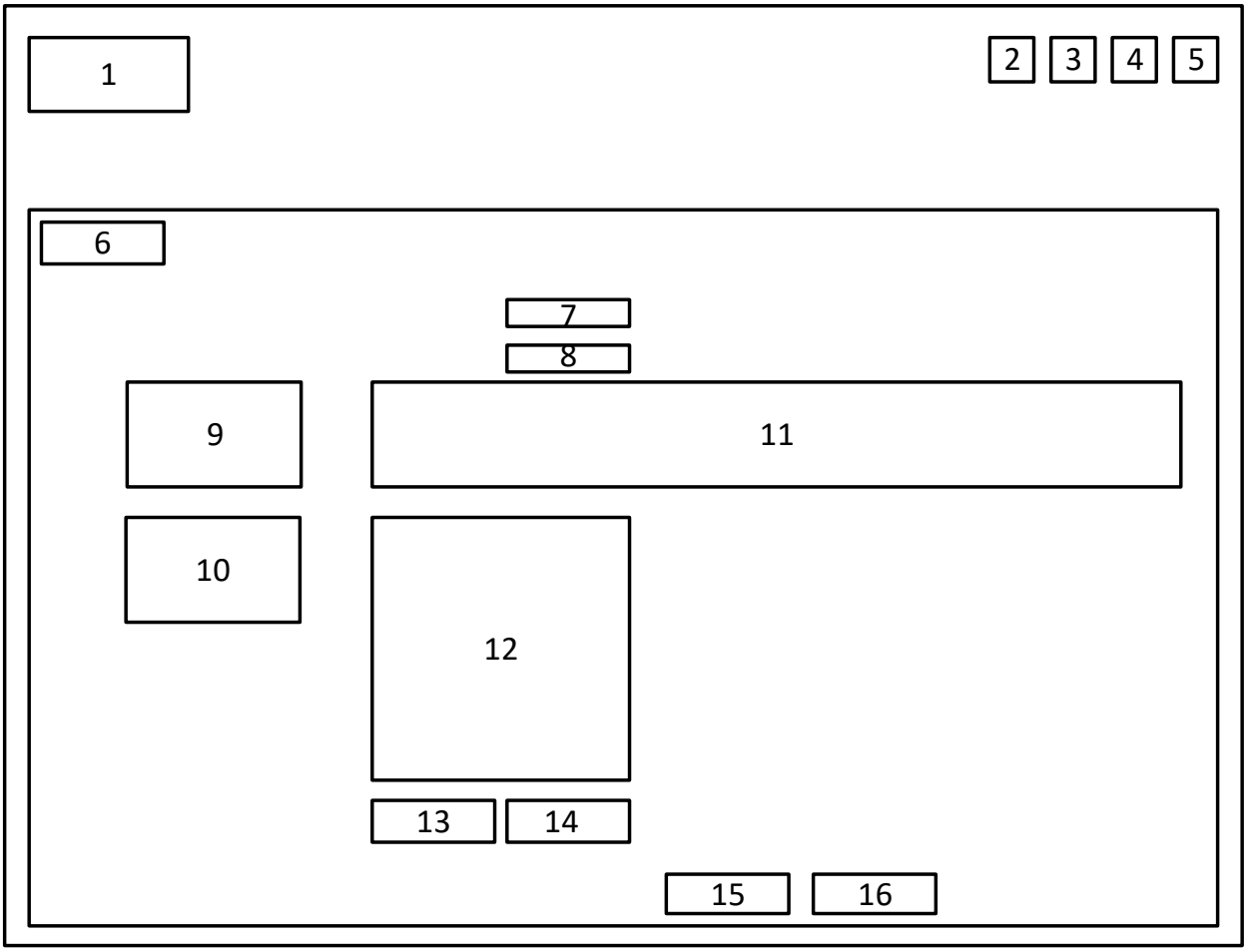

Рисунок 3.9 – Структурна схема вкладки перегляду рецепту

Елементи вкладки перегляду рецепту:

1. Логотип застосунку.
2. Кнопка пошуку рецептів.
3. Кнопка переходу в профіль користувача.
4. Кнопка виходу з аккаунту.
5. Кнопка переходу в кошик продуктів.
6. Кнопка повернення на попередню сторінку.
7. Назва страви.
8. Категорія страви.
9. Фото страви.
10. Відео приготування страви на Youtube.

11. Текст рецепту.

12. Перелік інгредієнтів.

13. Кнопка «Вибрати всі інгредієнти».

14. Кнопка «Зняти вибір з усіх інгредієнтів».

15. Кнопка «Зберегти рецепт».

16. Кнопка «Додати продукти до кошика».

Крім цього, розроблено блок-схему алгоритму роботи функції для додавання рецептів в «Улюблене» (рисунок 3.10).

Рисунок 3.10 – Блок-схема алгоритму роботи функції для додавання рецептів в «Улюблене»

3.5 Висновки

У третьому розділі, було проведено варіантний аналіз та здійснено вибір засобів реалізації вебзастосунку для підбору рецептів за інгредієнтами з можливістю придбання необхідних продуктів. Розроблено блок-схеми алгоритмів й структурні схеми інтерфейсу роботи модуля авторизації та реєстрації користувача, блок-схеми алгоритмів роботи та структурні схеми інтерфейсу головної вкладки, вкладки перегляду рецепту вебзастосунку.

4 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Аналіз методів тестування

Тестування вебзастосунків є невід'ємною частиною процесу їх розробки. Воно дозволяє перевірити функціонування програмного забезпечення та виявити помилки і недоробки [17].

Тестування вебзастосунків включає в себе перевірку як користувацького інтерфейсу, так і серверної частини. Кожен з цих компонентів вимагає свого підходу до тестування.

Тестування користувацького інтерфейсу вебзастосунків полягає в перевірці зручності та функціональності інтерфейсу. Це включає в себе перевірку навігації, дизайну, взаємодії з користувачем та інших аспектів.

Тестування серверної частини вебзастосунків полягає в перевірці роботи серверу, бази даних та інших серверних компонентів. Це включає в себе перевірку обробки даних, безпеки, продуктивності та інших аспектів.

Існує багато методів тестування вебзастосунків. Вони можуть бути поділені на ручні та автоматизовані методи.

Ручне тестування передбачає виконання тестів людиною. Цей метод дозволяє провести детальну перевірку програмного забезпечення та виявити помилки, які можуть бути пропущені автоматичними тестами.

Автоматизоване тестування передбачає виконання тестів спеціальними програмами. Цей метод дозволяє швидко та ефективно протестувати великі обсяги програмного забезпечення [18].

Одним з популярних методів тестування вебзастосунків є функціональне тестування. Воно полягає в перевірці функціонування програмного забезпечення відповідно до вимог специфікації.

Нефункціональне тестування вебзастосунків включає в себе перевірку таких аспектів, як продуктивність, надійність, безпека та зручність використання [19].

Тестування на проникнення є одним з видів нефункціонального

тестування. Воно полягає в перевірці безпеки вебзастосунків шляхом моделювання атак зловмисників.

Тестування на вразливості є ще одним видом нефункціонального тестування. Воно полягає в перевірці вебзастосунків на наявність вразливостей, які можуть бути використані зловмисниками.

Тестування на сумісність дозволяє перевірити, чи працює вебзастосунок коректно в різних середовищах. Це включає в себе перевірку сумісності з різними браузерами, операційними системами та пристроями [20].

Тестування на навантаження дозволяє перевірити, чи може вебзастосунок працювати коректно при великому навантаженні. Це включає в себе перевірку продуктивності, стабільності та надійності.

Тестування на стійкість дозволяє перевірити, чи може вебзастосунок працювати коректно при тривалому навантаженні. Це включає в себе перевірку стійкості до збоїв та відновлення після них.

Тестування на локалізацію дозволяє перевірити, чи працює вебзастосунок коректно в різних мовних середовищах. Це включає в себе перевірку перекладів, форматів дати та часу, валют та інших аспектів.

Тестування на коректність роботи з датами дозволяє перевірити, чи працює вебзастосунок коректно з датами. Це включає в себе перевірку обробки дат, часових зон та інших аспектів [21].

Тестування на коректність роботи з електронною поштою дозволяє перевірити, чи працює вебзастосунок коректно з електронною поштою. Це включає в себе перевірку надсилання та отримання повідомлень, обробки вкладень та інших аспектів.

Тестування на коректність роботи з базами даних дозволяє перевірити, чи працює вебзастосунок коректно з базами даних. Це включає в себе перевірку обробки даних, запитів до бази даних та інших аспектів.

Тестування на коректність роботи з API дозволяє перевірити, чи працює вебзастосунок коректно з API. Це включає в себе перевірку обробки запитів, відповідей та інших аспектів.

Тестування на коректність роботи з мережею дозволяє перевірити, чи працює вебзастосунок коректно з мережею. Це включає в себе перевірку обробки запитів, відповідей, протоколів та інших аспектів.

Тестування на коректність роботи з файлами дозволяє перевірити, чи працює вебзастосунок коректно з файлами. Це включає в себе перевірку завантаження, збереження, видалення файлів та інших аспектів.

Тестування на коректність роботи з мультимедіа дозволяє перевірити, чи працює вебзастосунок коректно з мультимедіа. Це включає в себе перевірку відтворення, запису, редагування мультимедіа та інших аспектів.

Тестування на коректність роботи з форматами файлів дозволяє перевірити, чи працює вебзастосунок коректно з різними форматами файлів. Це включає в себе перевірку читання, запису, конвертування файлів та інших аспектів.

Тестування на коректність роботи з браузерами дозволяє перевірити, чи працює вебзастосунок коректно з різними браузерами. Це включає в себе перевірку сумісності з Chrome, Firefox, Safari, Edge, Internet Explorer та іншими браузерами.

Тестування на коректність роботи з серверами дозволяє перевірити, чи працює вебзастосунок коректно з серверами.

Таким чином, тестування вебзастосунків є складним процесом, який включає в себе перевірку різних аспектів функціонування програмного забезпечення. Для проведення тестування можуть бути використані різні методи, як ручні, так і автоматизовані.

4.2 Тестування вебзастосунку

Для демонстрації функціоналу розробленого вебзастосунку для підбору рецептів за інгредієнтами з можливістю придбання необхідних продуктів та коректність його роботи, проведемо його тестування.

Спершу, відкриємо вкладку реєстрації та створимо новий аккаунт (рисунок 4.1).

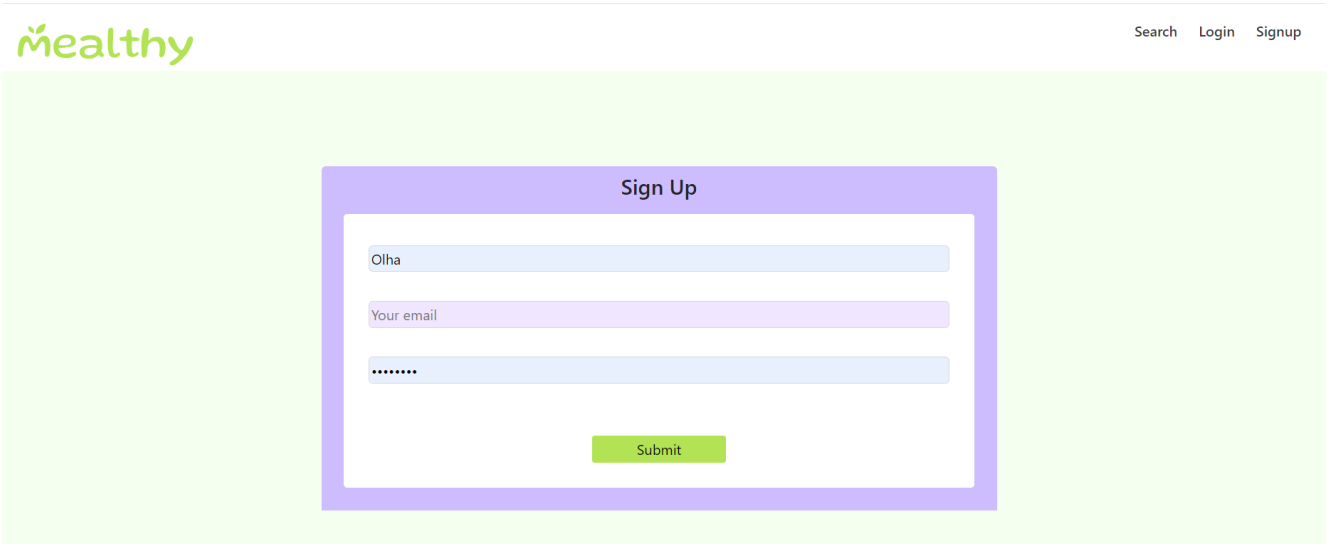

Рисунок 4.1 – Вкладка реєстрації

Після успішної реєстрації, відбувається автоматична авторизація та перехід на головну сторінку застосунку (рисунок 4.2). Головна сторінка містить короткий опис функціоналу вебзастосунку та містить кнопку для переходу на вкладку пошуку рецептів посеред екрану.

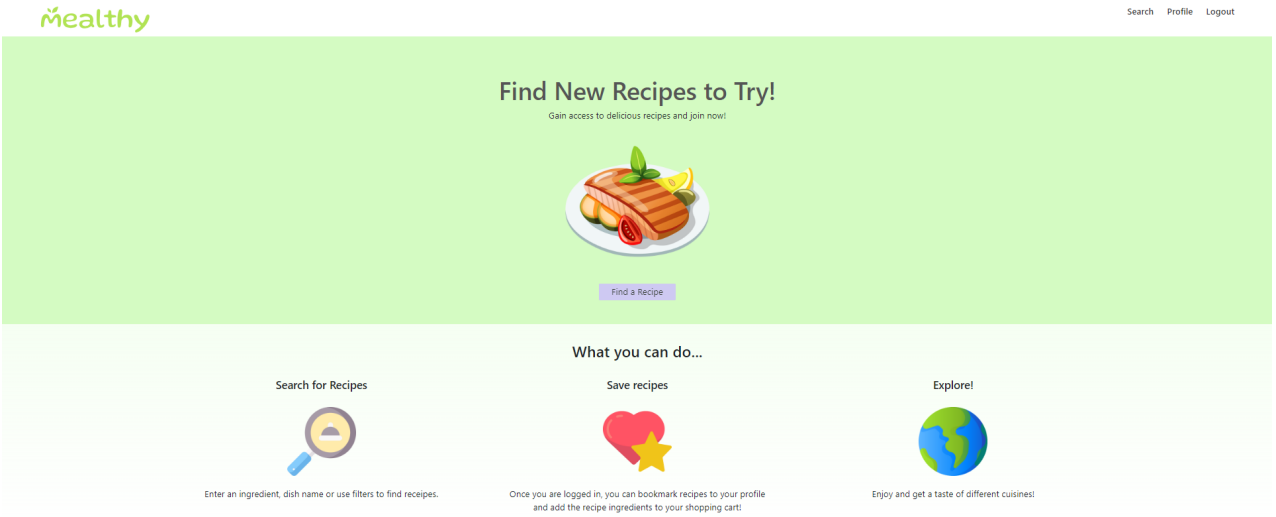

Рисунок 4.2 – Головна сторінка застосунку

Відкриємо сторінку пошуку рецептів, натиснувши кнопку «Find a Recipe» (рисунок 4.3). Спершу буде відображено випадковий перелік з 10 страв. Після кожного оновлення сторінки, цей список оновлюється. Він перебуватиме у такому стані доти, доки не буде обрано інший критерій для показу рецептів.

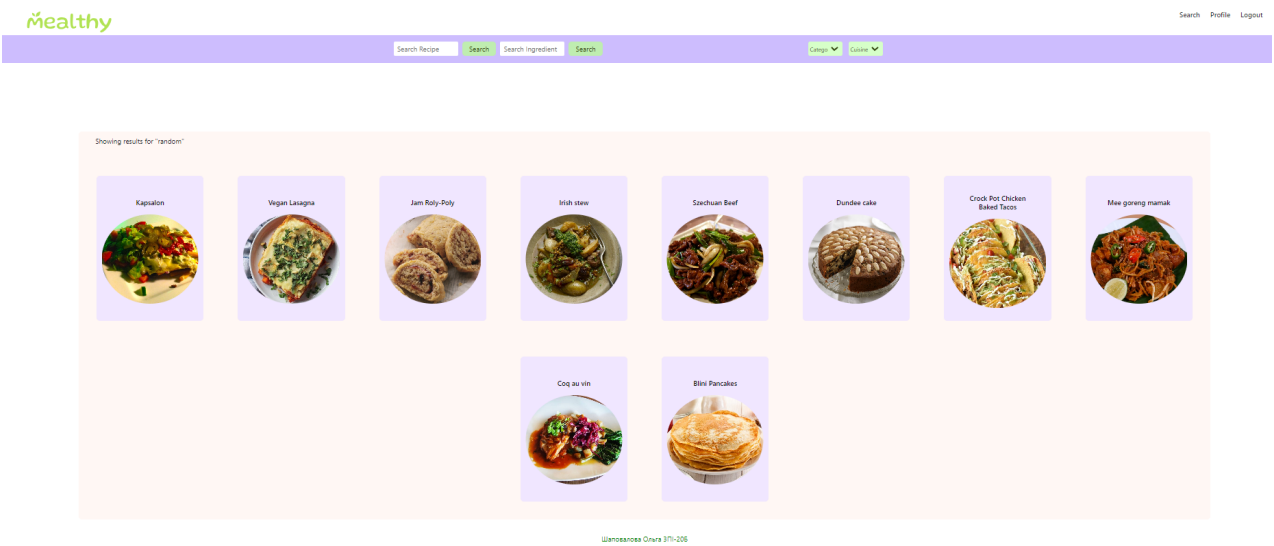

Рисунок 4.3 – Сторінка пошуку рецептів

Перевіримо пошук за інгредієнтами. Для цього, введемо у виділене для цього поле інгредієнт «огірок» (cucumber). В результаті, буде відображено перелік рецептів, в яких міститься огірок (рисунок 4.4).

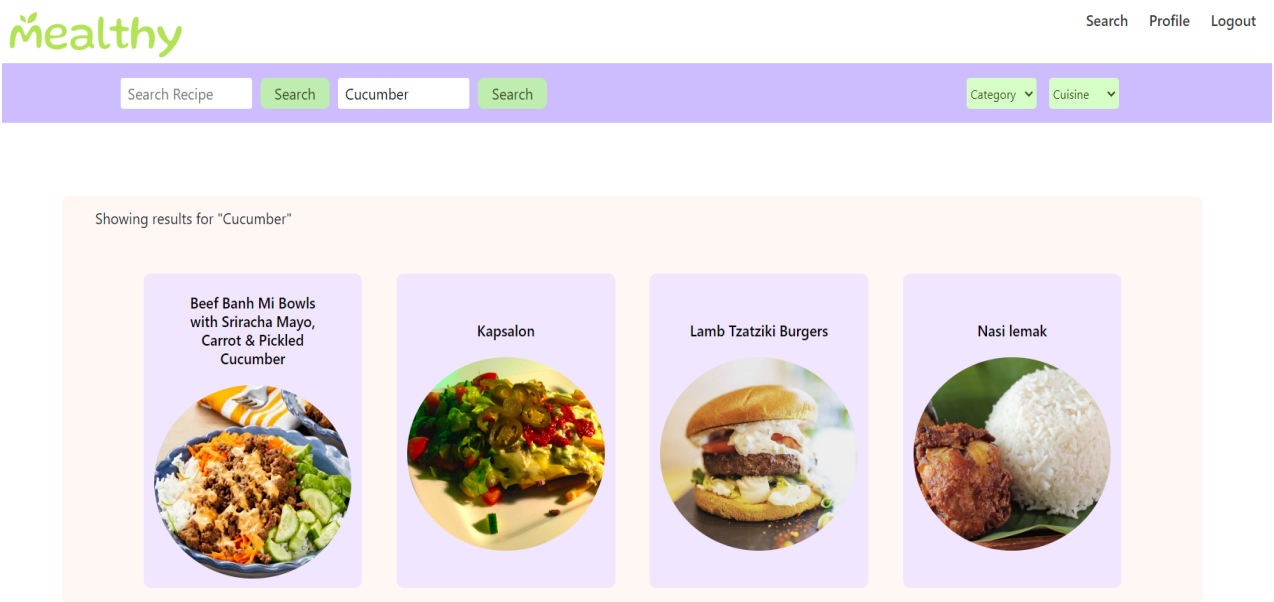

Рисунок 4.4 – Пошук за інгредієнтами

Здійснимо пошук за назвою страви, наприклад «піца» (рисунок 4.5). Можемо переконатися, що пошуковик працює, як передбачалося, та виводить шуканий рецепт.

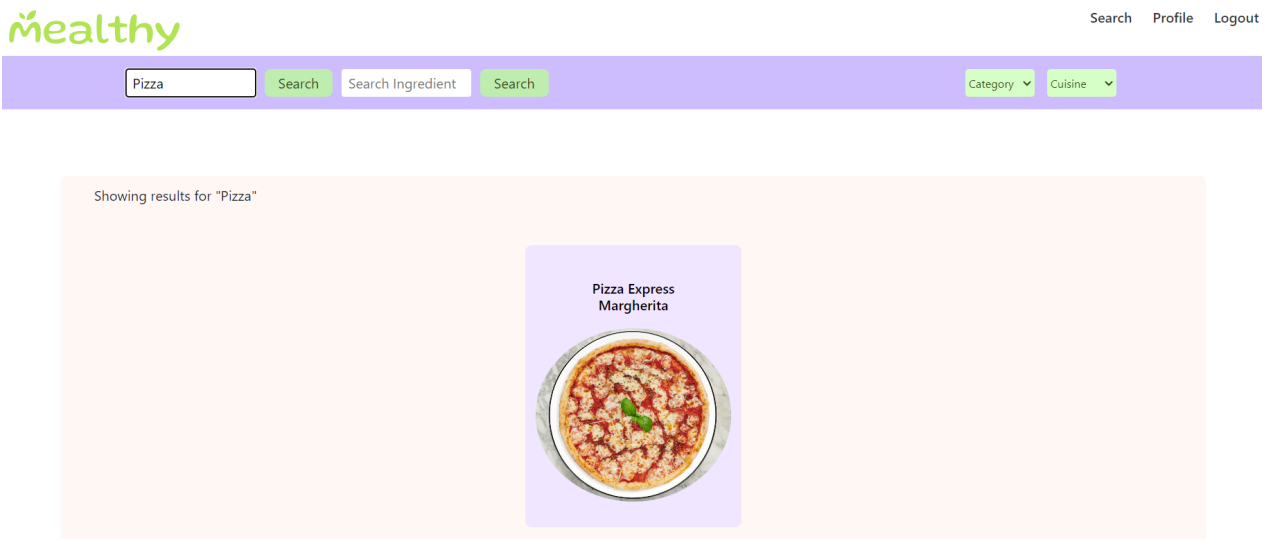

Рисунок 4.5 – Пошук за рецептом

Перевіримо пошук за національними кухнями, для цього відкриємо перелік відповідний список (рисунок 4.6).

Рисунок 4.6 – Перелік національних кухонь

Цей список дозволяє обрати серед значного переліку національних кухонь, що містяться в базі даних вебзастосунку. Вибір певної країни відображає перелік рецептів, які походять з цієї країни, в алфавітному порядку.

Оберемо американську кухню. Результат зображено на рисунку 4.7.

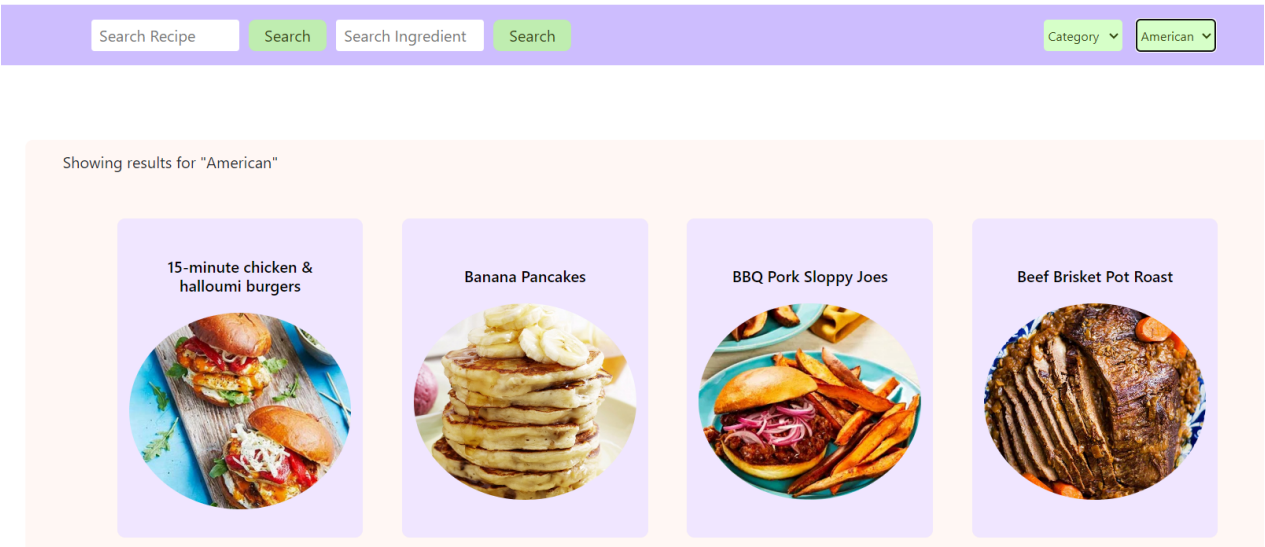

Рисунок 4.7 – Перелік страв американської національної кухні

Перевіримо пошук за категоріями страв, для цього відкриємо перелік категорій страв (рисунок 4.8).

Рисунок 4.8 – Перелік категорій страв

Цей список є аналогічним до списку пошуку за національними кухнями, однак, призначений для пошуку за інгридієнтами.

Оберемо з цього переліку «Vegetarian» (вегетаріанські страви). В результаті, будуть відображені вегетеріанські страви (рисунок 4.9).

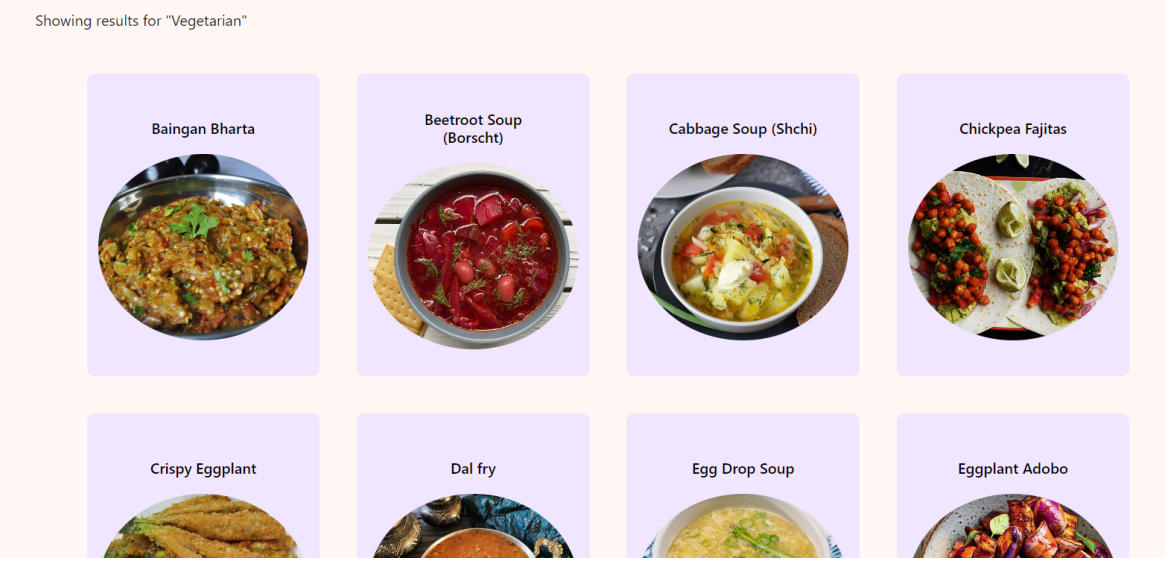

Рисунок 4.9 – Перелік вегетеріанських страв

Відкриємо один з цих рецептів, наприклад борщ. Відбудеться перехід на нову вкладку з переглядом рецепту приготування та іншими деталями рецепту. Результат зображено на рисунку 4.10.

Рисунок 4.10 – Сторінка з рецептом

Тут ми можемо переглянути рецепт страви, відео приготування на Youtube,

та перелік інгредієнтів.

Збережемо цей рецепт. Тепер він відображатиметься в профілі (рисунок 4.11).

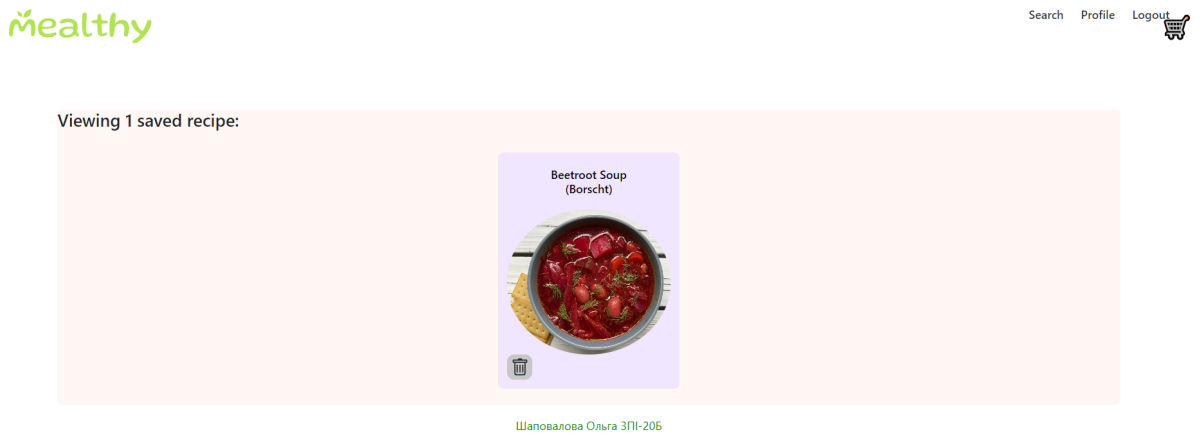

Рисунок 4.11 – Збережений рецепт в профілі користувача

Оберемо всі інгредієнти, що необхідні для приготування борщу, та додамо їх до кошика (рисунок 4.12). У кошику ж ми можемо змінити кількість певних інгридієнтів, що потрібно замовити, або видалити з нього такі, що не є потрібними.

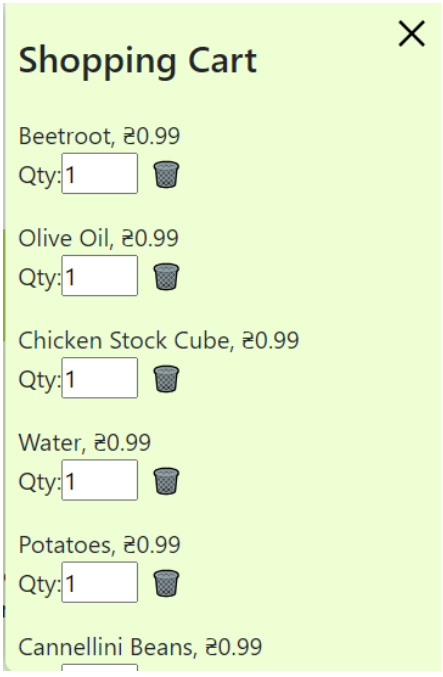

Рисунок 4.12 – Додані до кошика інгредієнти

Перейдемо на сторінку оплати цих інгредієнтів. Відбувається переадресія на сторінку Stripe, яка надає функціонал для прийому оплати (рисунок 4.13). На цій сторінці доступні функції оплати через Google Pay або шляхом введення банківської карти.

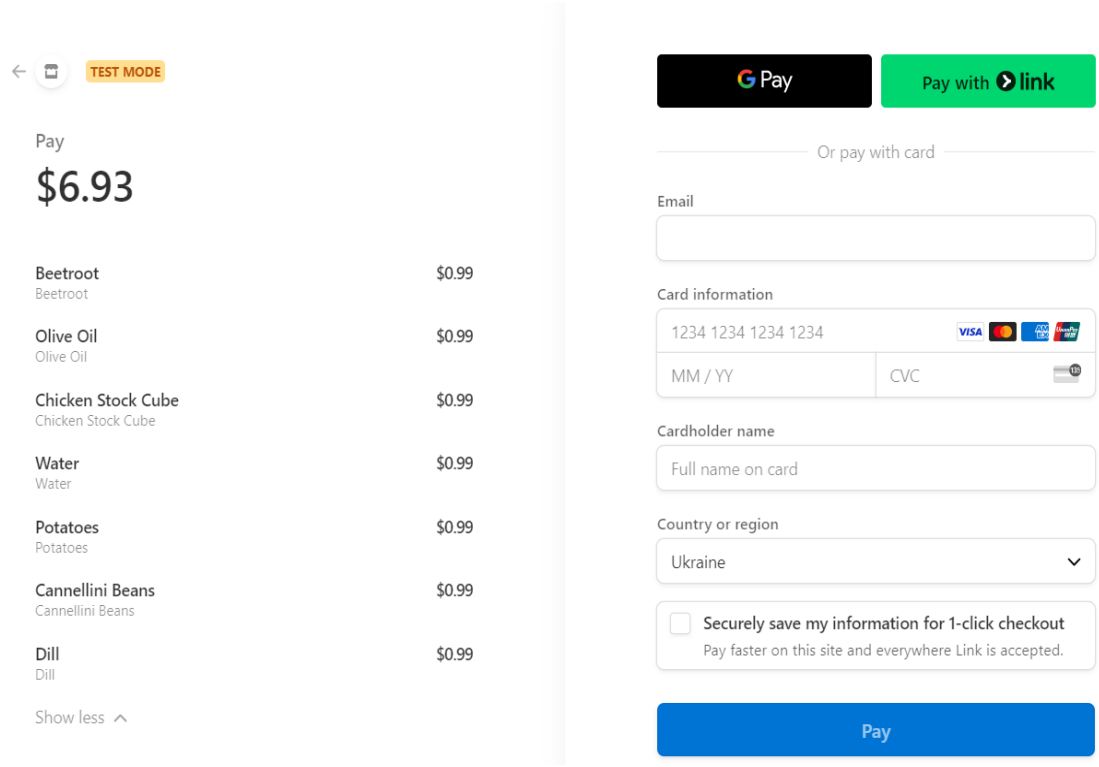

Рисунок 4.13 – Сторінка оплати за інгредієнти

Після введення даних банківської карти та підтвердження покупки, виводиться повідомлення про успішну оплату, як на рисунку 4.14.

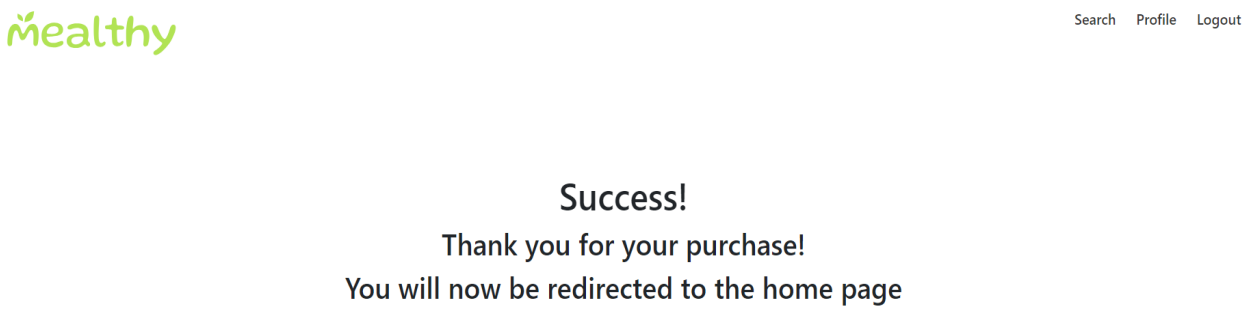

Рисунок 4.14 – Повідомлення про успішну оплату

4.3 Системні вимоги

У таблиці 4.1 наведені мінімальні системні вимоги для роботи вебзастосунку.

Таблиця 4.1 – Мінімальні конфігураційні вимоги до апаратного забезпечення

Процесор	Intel Core i3-9100F
Оперативна пам'ять	2 GB
Пропускна здатність мережі	10 Mbps
Вільного місця на диску	2ГБ

У таблиці 4.2 наведені рекомендовані системні вимоги для роботи вебзастосунку.

Таблиця 4.2 – Рекомендовані конфігураційні вимоги до апаратного забезпечення

Процесор	Intel Core i5-1135S 2.9 GHz
Оперативна пам'ять	8 GB
Пропускна здатність мережі	100 Mbps
Вільного місця на диску	4ГБ

Нижче наведено детальніший опис системних вимог та підтримуваних мов вебзастосунку.

1. Операційна система: Windows 10 або новіше**,** macOS 10.14 (Mojave) або новіше**,** Linux (рекомендовані дистрибутиви: Ubuntu 18.04 або новіше)**,** Android 8.0 (Oreo) або новіше**,** iOS 12 або новіше.

2. Веб-браузери: остання версія Google Chrome**,** Mozilla Firefox**,** Safari, Microsoft Edge**,** Opera.

3. Мобільні пристрої (смартфони та планшети):

- Процесор: двоядерний або краще (Snapdragon 450/Exynos 7904 або краще)

- Оперативна пам'ять: Мінімум 2 ГБ (рекомендується 4 ГБ або більше)
- Дозвіл екрану: Мінімум 720x1280 (рекомендується 1080x1920 або більше)

4. Підключення до Інтернету. Швидкість інтернет-з'єднання сінімум 1 Мбіт/с (мегабіт на секунду) (рекомендується 5 Мбіт/с (мегабіт на секунду) або вище для швидкого завантаження сторінок та медіа контенту).

5. Додаткові вимоги:

- Актуальна версія JavaScript увімкнена в браузері.
- Підтримка cookies у браузері (для збереження налаштувань та сесій користувача).
- Підтримка Web Storage (для локального зберігання даних на клієнті).

Таким чином, було визначено мінімальні та рекомендовані системні вимоги для роботи з розробленим вебзастосунком для пошуку рецептів з можливістю придбання необхідних продуктів, описано детальні вимоги та додаткові вимоги.

4.4 Висновки

У четвертому розділі, було проведено аналіз методів тестування веб-систем. Було проведено тестування вебзастосунку для підбору рецептів за інгредієнтами з можливістю придбання необхідних продуктів. Було продемонстровано його функціонал та його здатність виконувати поставлені на початку розробки задачі. Визначено мінімальні та рекомендовані системні вимоги для роботи з вебзастосунком.

ВИСНОВКИ

В результаті виконання бакалаврської дипломної роботи, було розроблено вебзастосунок для підбору рецептів за інгредієнтами з можливістю придбання необхідних продуктів. Розроблений вебзастосунок вирішує дві основні проблеми: складний процес пошуку рецептів та недоліки у процесі придбання необхідних продуктів для приготування страв.

Для розробки було використано середовище розробки Visual Studio Code, мову програмування JavaScript.

У першому розділі, було проведено аналіз стану питання вебзастосунків для підбору рецептів за інгредієнтами, порівняльний аналіз аналогів, аналіз методів розв’язання задачі. На їх основі було доведено доцільність та актуальність власної розробки, поставлено задачі дослідження.

У другому розділі, було проведено аналіз даних, які генеруються, використовуються, завантажуються та обробляються при роботі вебзастосунку для підбору рецептів за інгредієнтами з можливістю придбання необхідних продуктів. Було розроблено модель вебзастосунку, метод пошуку рецептів, метод замовлення продуктів, загальний алгоритм роботи та діаграми станів.

У третьому розділі, було проведено варіантний аналіз та здійснено вибір засобів реалізації вебзастосунку для підбору рецептів за інгредієнтами з можливістю придбання необхідних продуктів. Розроблено блок-схеми алгоритмів й структурні схеми інтерфейсу роботи модуля авторизації та реєстрації користувача, блок-схеми алгоритмів роботи та структурні схеми інтерфейсу головної вкладки, вкладки перегляду рецепту вебзастосунку.

У четвертому розділі, було проведено аналіз методів тестування веб-систем. Було проведено тестування вебзастосунку для підбору рецептів за інгредієнтами з можливістю придбання необхідних продуктів. Було продемонстровано його функціонал та його здатність виконувати поставлені на початку розробки задачі. Визначено мінімальні та рекомендовані системні вимоги для роботи з вебзастосунком.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ольга Шаповалова. Розробка вебзастосунку для підбору рецептів за інгредієнтами з можливістю придбання необхідних продуктів/ О. О. Шаповалова, О. В. Романюк // Матеріали конференції «LIII Науково-технічна конференція підрозділів Вінницького національного технічного університету (2024)», Вінниця, 2024. [Електронний ресурс] – Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2024/paper/view/20632/17194

2. Романюк О.В. Використання MERN stack при розробці веб-застосунків / О.В. Романюк, О.О. Шаповалова // Матеріали конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2024)», Вінниця, 2024. URL: https://conferences.vntu.edu.ua/index.php/mn/mn2024/paper/viewFile/21625/17890

3. Cookpad: веб-сайт [Електронний ресурс] – Режим доступу до ресурсу: https://cookpad.com/ua/homepage (дата звернення: 15.03.2024).

4. Yummly: веб-сайт [Електронний ресурс] – Режим доступу до ресурсу: https://www.yummly.com/ (дата звернення: 15.03.2024).

5. Mealime: веб-сайт [Електронний ресурс] – Режим доступу до ресурсу: https://www.mealime.com/ (дата звернення: 15.03.2024).

6. MERN Stack explained: веб-сайт [Електронний ресурс] – Режим доступу до ресурсу: https://www.mongodb.com/resources/languages/mern-stack (дата звернення: 16.03.2024).

7. MongoDB: веб-сайт [Електронний ресурс] – Режим доступу до ресурсу: https://www.mongodb.com/ (дата звернення: 16.03.2024).

8. Express.js: веб-сайт [Електронний ресурс] – Режим доступу до ресурсу: https://expressjs.com/ (дата звернення: 16.03.2024).

9. React: веб-сайт [Електронний ресурс] – Режим доступу до ресурсу: https://legacy.reactjs.org/ (дата звернення: 18.03.2024).

10. Node.js: веб-сайт [Електронний ресурс] – Режим доступу до ресурсу: https://nodejs.org/en (дата звернення: 19.03.2024).

11. Stripe: веб-сайт [Електронний ресурс] – Режим доступу до ресурсу: https://stripe.com/ (дата звернення: 02.04.2024).

12. TheGraphQL: веб-сайт [Електронний ресурс] – Режим доступу до ресурсу: https://thegraph.com/docs/en/querying/graphql-api/ (дата звернення: 03.04.2024).

13. JavaScript: веб-сайт [Електронний ресурс] – Режим доступу до ресурсу: https://w3schoolsua.github.io/js/index.html#gsc.tab=0 (дата звернення: 24.04.2024).

14. TypeScript: веб-сайт [Електронний ресурс] – Режим доступу до ресурсу: https://www.typescriptlang.org/ (дата звернення: 25.04.2024).

15. PHP: веб-сайт [Електронний ресурс] – Режим доступу до ресурсу: https://www.php.net/ (дата звернення: 26.04.2024).

16. Visual Studio Code: веб-сайт [Електронний ресурс] – Режим доступу до ресурсу: https://code.visualstudio.com/ (дата звернення: 27.04.2024).

17. Web application testing: веб-сайт [Електронний ресурс] – Режим доступу до ресурсу: https://qatestlab.com/solutions/by-focus-area/web-application-testing/ (дата звернення: 20.05.2024).

18. Automated software testing: веб-сайт [Електронний ресурс] – Режим доступу до ресурсу: https://www.atlassian.com/continuous-delivery/software-testing/automated-testing (дата звернення: 21.05.2024).

19. Non functional testing: веб-сайт [Електронний ресурс] – Режим доступу до ресурсу: https://www.perforce.com/blog/alm/what-non-functional-testing (дата звернення: 22.05.2024).

20. Roger Pressman. Software Engineering: A Practitioner's Approach. Нью-Йорк: McGraw Hill, 2023. 704с.

21. Cen Kaner. Lessons Learned in Software Testing. Гобокен: Wiley, 2022. 320с.

ДОДАТКИ

64

ДОДАТОК А

Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О.Н.
« 12 » березня 2024 р.

Технічне завдання

на бакалаврську дипломну роботу «Розробка вебзастосунку для підбору рецептів за інгредієнтами з можливістю придбання необхідних продуктів» за спеціальністю 121 – Інженерія програмного забезпечення

Керівник бакалаврської дипломної роботи:
к.т.н., доц. О.В. Романюк
« 12 » березня 2024 року.

Виконала:
студентка гр. ЗПІ-20б О.О. Шаповалова
« 12 » березня 2024 року.

м. Вінниця – 2024 рік

1. Найменування та галузь застосування

Бакалаврська дипломна робота: «Розробка вебзастосунку для підбору рецептів за інгредієнтами з можливістю придбання необхідних продуктів».

Галузь застосування – вебтехнології.

2. Підстава для розробки.

Завдання на роботу, яке затверджене на засіданні кафедри програмного забезпечення – протокол №17 від « 20 » березня 2024 р.

3. Мета та призначення розробки.

Метою роботи є удосконалення процесу пошуку рецептів та замовлення необхідних продуктів за рахунок розробки спеціального вебзастосунку, в якому реалізовані нові методи пошуку рецептів та замовлення продуктів.

4. Вихідні дані для проведення НДР

1. MERN Stack explained [Електронний ресурс] – Режим доступу до ресурсу: https://www.mongodb.com/resources/languages/ mern-stack

2. React [Електронний ресурс] – Режим доступу до ресурсу: https://legacy.reactjs.org/

3. JavaScript [Електронний ресурс] – Режим доступу до ресурсу: https://w3schoolsua.github.io/js/index.html#gsc.tab=0

4. Web application testing [Електронний ресурс] – Режим доступу до ресурсу: https://qatestlab.com/solutions/by-focus-area/web-application-testing/

5. Технічні вимоги

Тип програми – вебзастосунок; модель розробки – ітеративна; засоби організації даних програми – СКБД MongoDB; вхідні дані: персональні дані користувача; вихідні дані: перелік рецептів, список інгредієнтів, фото та відео рецептів; середовище розробки – Visual Studio Code; мова програмування – JavaScript.

6. Конструктивні вимоги.

Вебзастосунок має відповідати ергономічним та технічним вимогам. Текстова та графічна документація повинна відповідати стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

–пояснювальна записка до бакалаврської дипломної роботи;

–технічне завдання;

–лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи
1	Аналіз розвитку програм для підбору рецептів	14.03.2024 – 24.03.2024
2	Розробка моделі вебзастосунку	25.03.2024 – 31.03.2024
3	Розробка методу замовлення продуктів	01.04.2024 – 07.04.2024
4	Розробка методу пошуку рецептів	08.04.2024 – 15.04.2024
5	Розробка алгоритмів вебзастосунку	16.04.2024 – 23.04.2024
6	Аналіз і вибір мови програмування та середовища розробки	24.04.2024 – 30.04.2024
7	Розробка програмного забезпечення	01.05.2024 – 19.05.2024
8	Тестування програмного забезпечення	20.05.2024 – 24.05.2024
9	Оформлення матеріалів до захисту БДР	25.05.2024 – 30.05.2024

10 Порядок контролю та прийняття.

Всі етапи бакалаврської роботи контролюються науковим керівником згідно плану по виконанню роботи. Прийняття бакалаврської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком захисту. Дозволяється корегування бакалаврської дипломної роботи.

67

ДОДАТОК Б

Протокол перевірки кваліфікаційної роботи на наявність текстових запозичень

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Розробка вебзастосунку для підбору рецептів за інгредієнтами з можливістю придбання необхідних продуктів.
Тип роботи: БДР
Підрозділ : кафедра програмного забезпечення, ФІТКІ
Науковий керівник: к.т.н., доц. каф. ПЗ Романюк О.В.

Оригінальність	92,9%
Схожість	7,1 %

Аналіз звіту подібності

Особа, відповідальна за перевірку _____ Черноволик Г. О.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи _____ Шаповалова О.О.
Керівник роботи _____ Романюк О.В.

ДОДАТОК В

Лістинг програми

```
import React, { useState } from "react";
import { Container, Row, Col } from "react-bootstrap";
import { Link } from "react-router-dom";
// import RecipeSearch from "../extras/RecipeSearch";
import "./home.css";
import { homeArt } from "../assets";
import * as Icon from "react-bootstrap-icons";

const Main = () => {
  console.log('home: /')
  return (
    <Col className="home-container">
      <Row>
        <Col className="home-banner banner-container">
          <h2>Find New Recipes to Try!</h2>
          <p>Gain access to delicious recipes and join now!</p>
          <img src={homeArt} alt="palceholder" />
          <button id="get-started-btn">
            <Link to="/search">Find a Recipe</Link>
          </button>
        </Col>
      </Row>
      <Col className="home-description">
        <h3>What you can do...</h3>
        <Row className="description-item-container">
          <Col className="description-item">
            <h4>Search for Recipes</h4>
```

```
                <img src="/search.png" alt="" style={{maxWidth: '125px', borderRadius: 0, margin: "1.25rem 0"}}/>
                {/* <img src="https://via.placeholder.com/125" alt="palceholder" /> */}
                <p>
                  Enter an ingredient, dish name or use filters to find receipes.
                </p>
              </Col>
              <Col className="description-item">
                <h4>Save recipes</h4>
                <img src="/heart.png" alt="" style={{maxWidth: '125px', borderRadius: 0, margin: "1.25rem 0"}}/>
                {/* <img src="https://via.placeholder.com/125 " alt="palceholder" /> */}
                <p>
                  Once you are logged in, you can bookmark recipes to your profile
                  and add the recipe ingredients to your shopping cart!
                </p>
              </Col>
              <Col className="description-item">
                <h4>Explore!</h4>
                <img src="/planet.png" alt="" style={{maxWidth: '125px', borderRadius: 0, margin: "1.25rem 0"}}/>
                <p>Enjoy and get a taste of different cuisines!</p>
              </Col>
            </Row>
          </Col>
        </Col>
      );
    };

    export default Main;
```

```
import React, { useState } from 'react';
import { Link } from 'react-router-dom';
import { useMutation } from '@apollo/client';
import { LOGIN_USER } from '../utils/mutations';
import { saveMealIds } from "../utils/localStorage";

import './Login.css';
import Auth from '../utils/auth';

const Login = (props) => {
  const [formState, setFormState] = useState({ email: '', password: '' });
  const [login, { error, data }] = useMutation(LOGIN_USER);

  // update state based on form input changes
  const handleChange = (event) => {
    const { name, value } = event.target;

    setFormState({
      ...formState,
      [name]: value,
    });
  };

  // submit form
  const handleFormSubmit = async (event) => {
    event.preventDefault();
    console.log(formState);
    try {
      const { data } = await login({
```

```
      variables: { ...formState },
    });

    Auth.login(data.login.token);

    const idMeals = data.login.user.savedRecipes.map((recipe) => {
      return recipe.idMeal;
    });

    saveMealIds(idMeals);

  } catch (e) {
    console.error(e);
  }

  // clear form values
  setFormState({
    email: '',
    password: '',
  });
};

return (
  <main
    className="login-container"
    style={{ backgroundColor: "#F5FFF0", height: "80vh" }}
  >
    <div className="login-form">
      <div className="card" style={{ border: "none" }}>
        <h4 className="card-header" style={{ backgroundColor: "#cdbdfe" }}>
```

```
  Login
</h4>
<div
  className="card-body"
  style={{
    display: "flex",
    flexDirection: "column",
    justifyContent: "center",
    margin: "0 auto",
    padding: "9px 25px 25px 25px",
    backgroundColor: "#cdbdfe",
  }}
>
  {data ? (
    <p>
      Success! You may now head{" "}
      <Link to="/">back to the homepage.</Link>
    </p>
  ) : (
    <form
      onSubmit={handleFormSubmit}
      style={{
        textAlign: "center",
        backgroundColor: "white",
        padding: "4%",
        borderRadius: "4px",
      }}
    >
      <input
        className="form-input login-input"
```

```
    placeholder="Your email"
    name="email"
    type="email"
    value={formState.email}
    onChange={handleChange}
   />
   <input
    className="form-input login-input"
    placeholder="******"
    name="password"
    type="password"
    value={formState.password}
    onChange={handleChange}
   />
   <button
    className="login-btn"
    style={{
     cursor: "pointer",
     backgroundColor: "rgb(177, 227, 84)",
     alignItems: "center",
     marginTop: "5%",
     width: "150px",
     height: "30px",
     borderRadius: "3px",
     border: "none",
    }}
   >
    Submit
   </button>
  </form>
```

```
          )}

          {error && (
            <div className="my-3 p-3 bg-danger text-white">
              {error.message}
            </div>
          )}
        </div>
      </div>
    </div>
  </main>
 );
};

export default Login;

import React, { useState, useEffect } from "react";

import "bootstrap/dist/css/bootstrap.min.css";
import {Col, Row} from "react-bootstrap";


import { searchRecipes } from "../utils/API";
import SearchResult from "../components/SearchResult";

import "./searchForm.css";

const inputReset = (id) => {

  document.getElementById(id).value = "";
```

```
};

const selectReset = (id) => {

  document.getElementById(id).selectedIndex = 0;

};



const SearchForm = () => {

  // const [state, dispatch] = useRecipeContext();

  // const { categories, areas, category, area, ingredient, mealName } = state;


  const [categoryList, setCategoryList] = useState([]);
  const [areaList, setAreaList] = useState([]);

  const [selectedCategory, setSelectedCategory] = useState("");
  const [selectedArea, setSelectedArea] = useState("");
  const [searchIngredient, setSearchIngredient] = useState("");
  const [searchMealName, setSearchMealName] = useState("");
  const [selectedIngredient, setSelectedIngredient] = useState("");
  const [selectedMealName, setSelectedMealName] = useState("");

  const handleSelectChange = (event) => {
    if (event.target.name === "area") {
```

```
    setSelectedArea(event.target.value);
    setSelectedCategory("");
    inputReset("mealName");
    inputReset("ingredient");
    selectReset("category");

  } else if (event.target.name === "category") {
    setSelectedCategory(event.target.value);
    setSelectedArea("");
    inputReset("mealName");
    inputReset("ingredient");
    selectReset("area");

  } else if (event.target.name === "ingredient") {
    setSearchIngredient(event.target.value);
    setSelectedCategory("");
    setSelectedArea("");



  } else if (event.target.name === "mealName") {
    setSearchMealName(event.target.value);
    setSelectedCategory("");
    setSelectedArea("");

  }
};

const handleFormSubmit = (event) => {
  event.preventDefault();
```

```
  console.log("Event: ", event.target.name);
  // if (!selectedCategory && !selectedArea) {
  //     return false;
  // }
  if (event.target.name === "ingredient") {
    setSelectedIngredient(searchIngredient);
    setSelectedMealName("");
    inputReset("mealName");
    selectReset("category");
    selectReset("area");
    // console.log(searchIngredient);
  } else if (event.target.name === "mealName") {
    setSelectedMealName(searchMealName);
    setSelectedIngredient("");
    inputReset("ingredient");
    selectReset("category");
    selectReset("area");
    // console.log("selectedMealName",selectedMealName);
  }

  // console.log("selectedIngredient: ",selectedIngredient);
  setSelectedCategory("");
  setSelectedArea("");

};

useEffect(() => {
  const getCategory = async (query) => {
    try {
      const response = await searchRecipes(query);
```

```
      if (!response.ok) {
        throw new Error("something went wrong!");
      }

      const { meals } = await response.json();
      // console.log(meals);

      if (query === "list.php?c=list") {
        setCategoryList(meals);
      } else {
        setAreaList(meals);
      }
    } catch (err) {
      console.error(JSON.parse(JSON.stringify(err)));
    }
  };

  getCategory("list.php?c=list");
  getCategory("list.php?a=list");
}, []);

return (
  <div className="container-fluid">
    <div className="search-container-wrap">
      <div className="search-container">
        <div className="search-small-screen">
          <input
            className="searchbar"
            type="text"
```

```
        name="mealName"
        placeholder="Search Recipe"
        onChange={handleSelectChange}
        id="mealName"
      />
      <button
        className="searchbar-btn"
        type="submit"
        name="mealName"
        onClick={handleFormSubmit}
      >
        Search
      </button>
    </div>
    <div>
      <input
        type="text"
        name="ingredient"
        className="searchbar"
        placeholder="Search Ingredient"
        onChange={handleSelectChange}
        id="ingredient"
      />
      <button
        type="submit"
        className="searchbar-btn"
        name="ingredient"
        onClick={handleFormSubmit}
      >
        Search
```

```
            </button>
          </div>
          <div className="filters">
            <select
              name="category"
              className="filter"
              onChange={handleSelectChange}
              id="category"
            >
              <option value="">Category</option>


              {categoryList.map((category) => (
                <option key={category.strCategory} value={category.strCategory}>
                  {category.strCategory}
                </option>
              ))}
            </select>{" "}
            <select name="area" className="filter"
onChange={handleSelectChange} id="area">
              <option value=""> Cuisine</option>
              {areaList.map((area) => (
                <option key={area.strArea} value={area.strArea}>
                  {area.strArea}
                </option>
              ))}
            </select>
          </div>
        </div>
      </div>
```

```
            <div>
              <SearchResult
                category={selectedCategory}
                area={selectedArea}
                ingredient={selectedIngredient}
                mealName={selectedMealName}
              />
            </div>
          </div>
        );
      };

      export default SearchForm;

      const { AuthenticationError } = require('apollo-server-express');
      const { User, Post, Comment, Recipe } = require('../models');
      const { signToken } = require('../utils/auth');
      require('dotenv').config();
      // const stripe = require('stripe')('sk_test_4eC39HqLyjWDarjtT1zdp7dc');
      const stripe = require('stripe')("sk_test_51P5oIwC3ykhbTLACVASdXx5xC1mSy6vab1yJdXhTvEAR7bUyDUPeN0qJ4ibokJxai2P5yiu2w71IoiSvbq0EmBmw00FvE8Vyxt");

      const resolvers = {
        Query: {
          me: async (parent, args, context) => {
            if (context.user) {
              return User.findOne({ _id: context.user._id }).populate('posts').populate('savedRecipes');
            }
```

```
  throw new AuthenticationError('You need to be logged in!');
},
posts: async (parent, { username }) => {
  const params = username ? { username } : {};
  return Post.find(params).sort({ createdAt: -1 });
},
post: async (parent, { postId }) => {
  return Post.findOne({ _id: postId }).populate('comments');
},
comments: async (parent, { postId }) => {
  const params = postId ? { postId } : {};
  return Comment.find(params).sort({ createdAt: -1 });
},
users: async () => {
  return User.find().populate('posts').populate('savedRecipes');
},
product: async (parent, { _id }) => {
  return await Product.findById(_id).populate('category');
},
order: async (parent, { _id }, context) => {
  if (context.user) {
    const user = await User.findById(context.user._id).populate({
      path: 'orders.products',
      populate: 'category'
    });

    return user.orders.id(_id);
  }

  throw new AuthenticationError('Not logged in');
```

```
},
checkout: async (parent, { products }, context) => {
  const url = new URL(context.headers.referer).origin;
  // const order = new Order({ products: args.products });
  const line_items = [];

  console.log("process.env.STRIPE_KEY:", process.env.STRIPE_KEY);

  console.log("url: ", url);
  console.log('args.products: ', products);
  // const { products } = args.products;

  // const { products } = await order.populate('products');

  try {
    for (let i = 0; i < products.length; i++) {
      const product = await stripe.products.create({
        name: products[i],
        description: products[i],
      });


      const price = await stripe.prices.create({
        product: product.id,
        unit_amount: 0.99 * 100,
        currency: 'usd',
      });

      line_items.push({
        price: price.id,
```

```
            quantity: 1
          });
        }

        console.log("line_items: ", line_items);

        const session = await stripe.checkout.sessions.create({
          payment_method_types: ['card'],
          line_items,
          mode: 'payment',
          success_url:
`${url}/success?session_id={CHECKOUT_SESSION_ID}`,
          cancel_url: `${url}/`
        });

        return { session: session.id };
      } catch (err) {
        console.log("err: ", err);

      }

    }
  },
  Mutation: {
    addUser: async (parent, args) => {
      const user = await User.create(args);
      const token = signToken(user);
      return { token, user };
    },
    login: async (parent, { email, password }) => {
```

```
  const user = await User.findOne({ email });

  if (!user) {
    throw new AuthenticationError('No user found with this email address');
  }

  const correctPw = await user.isCorrectPassword(password);

  if (!correctPw) {
    throw new AuthenticationError('Incorrect credentials');
  }

  const token = signToken(user);

  return { token, user };
},
addPost: async (parent, { postText }, context) => {
  if (context.user) {
    const post = await Post.create({
      postText,
      postAuthor: context.user.username,
    });

    await User.findOneAndUpdate(
      { _id: context.user._id },
      { $addToSet: { posts: post._id } }
    );

    return post;
  }
```

```
  throw new AuthenticationError('You need to be logged in!');
 },
 removePost: async (parent, { postId }, context) => {
  if (context.user) {
   const post = await post.findOneAndDelete({
    _id: postId,
    postAuthor: context.user.username,
   });

   await User.findOneAndUpdate(
    { _id: context.user._id },
    { $pull: { posts: post._id } }
   );

   return post;
  }
  throw new AuthenticationError('You need to be logged in!');
 },
 addComment: async (parent, { postId, commentText }, context) => {
  if (context.user) {
   return Post.findOneAndUpdate(
    { _id: postId },
    {
     $addToSet: {
      comments: { commentText, commentAuthor: context.user.username },
     },
    },
    {
     new: true,
     runValidators: true,
```

```
      }
    );
  }
  throw new AuthenticationError('You need to be logged in!');
},
removeComment: async (parent, { postId, commentId }, context) => {
  if (context.user) {
    return Post.findOneAndUpdate(
      { _id: postId },
      {
        $pull: {
          comments: {
            _id: commentId,
            commentAuthor: context.user.username,
          },
        },
      },
      { new: true }
    );
  }
  throw new AuthenticationError('You need to be logged in!');
},
saveRecipe: async (parent, { mealData }, context) => {
  console.log("mealData:", mealData);
  if (context.user) {
    return await User.findOneAndUpdate(
      { _id: context.user._id },
      { $addToSet: { savedRecipes: { ...mealData } } },
      { new: true, }
    );
```

```
      }
      throw new AuthenticationError('You need to be logged in!');
    },
    removeRecipe: async (parent, { idMeal }, context) => {
      if (context.user) {
        return await User.findOneAndUpdate(
          { _id: context.user._id },
          { $pull: { savedRecipes: { idMeal } } },
          { new: true, }
        );
      }
    }
  }
}

module.exports = resolvers;
```

ДОДАТОК Г

Ілюстративна частина

ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА ВЕБЗАСТОСУНКУ ДЛЯ ПІДБОРУ РЕЦЕПТІВ ЗА ІНГРЕДІЄНТАМИ З МОЖЛИВІСТЮ ПРИДБАННЯ НЕОБХІДНИХ ПРОДУКТІВ

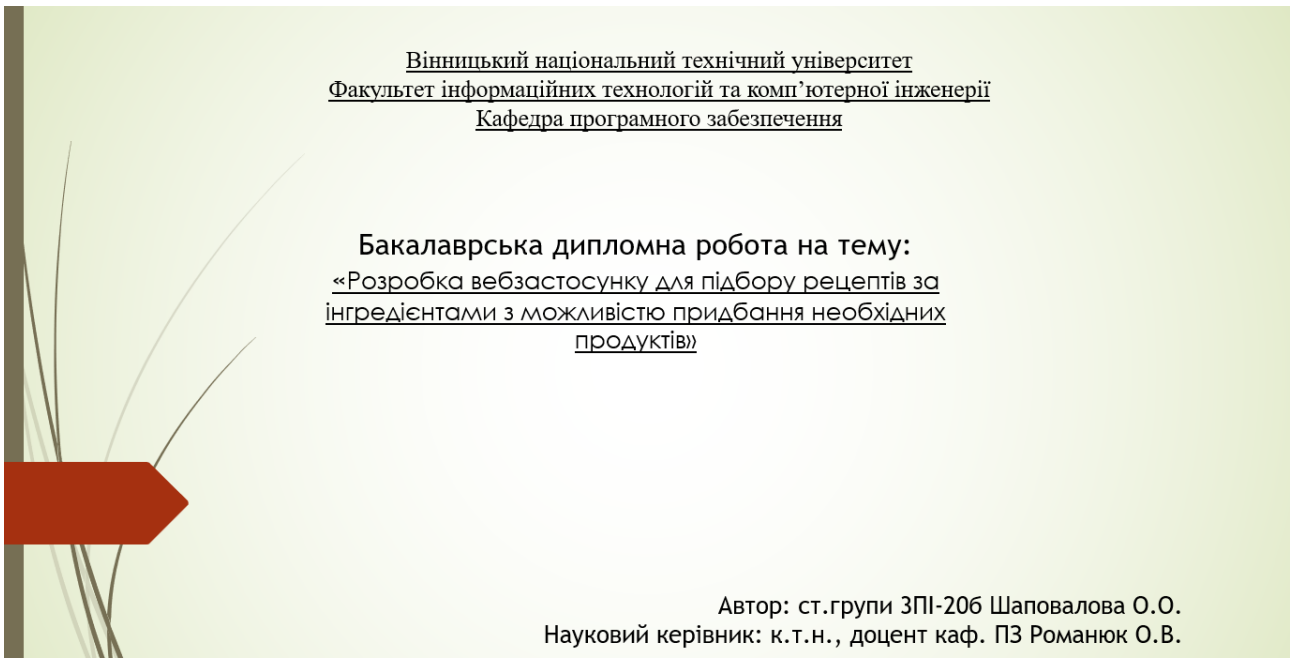

Рисунок Г.1 – Титульний слайд

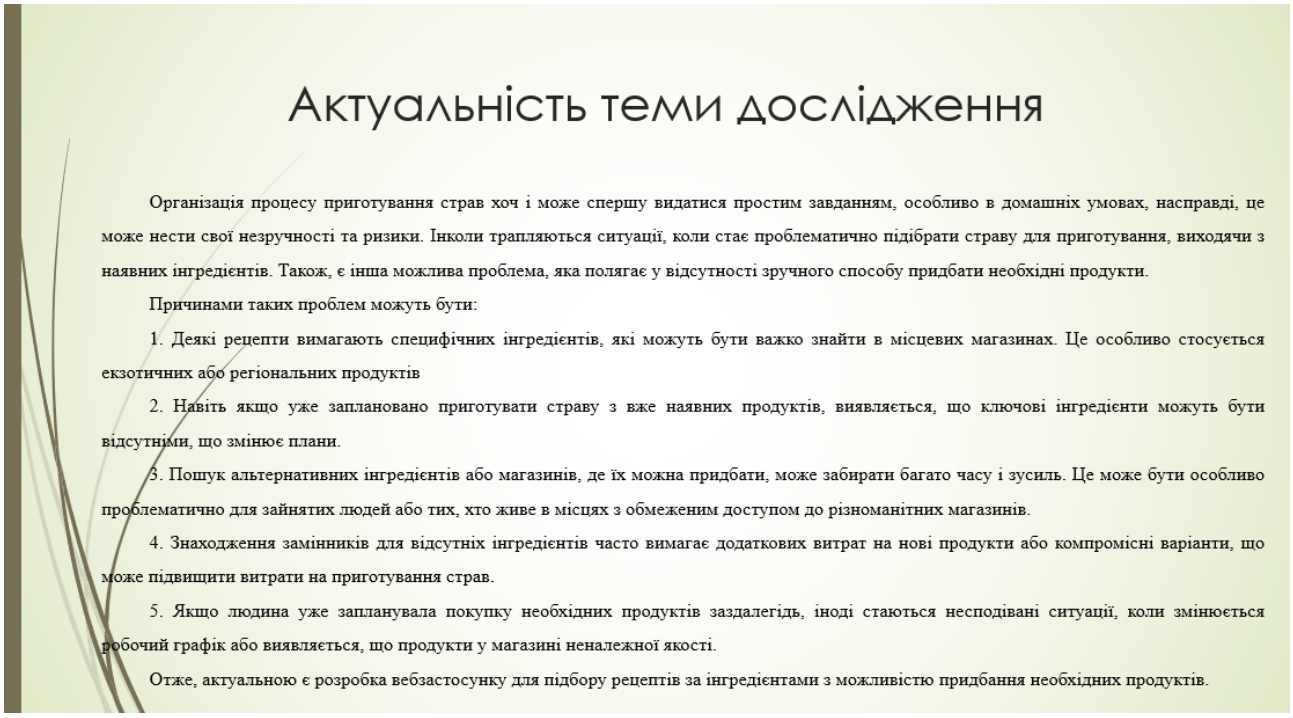

Рисунок Г.2 – Актуальність теми дослідження

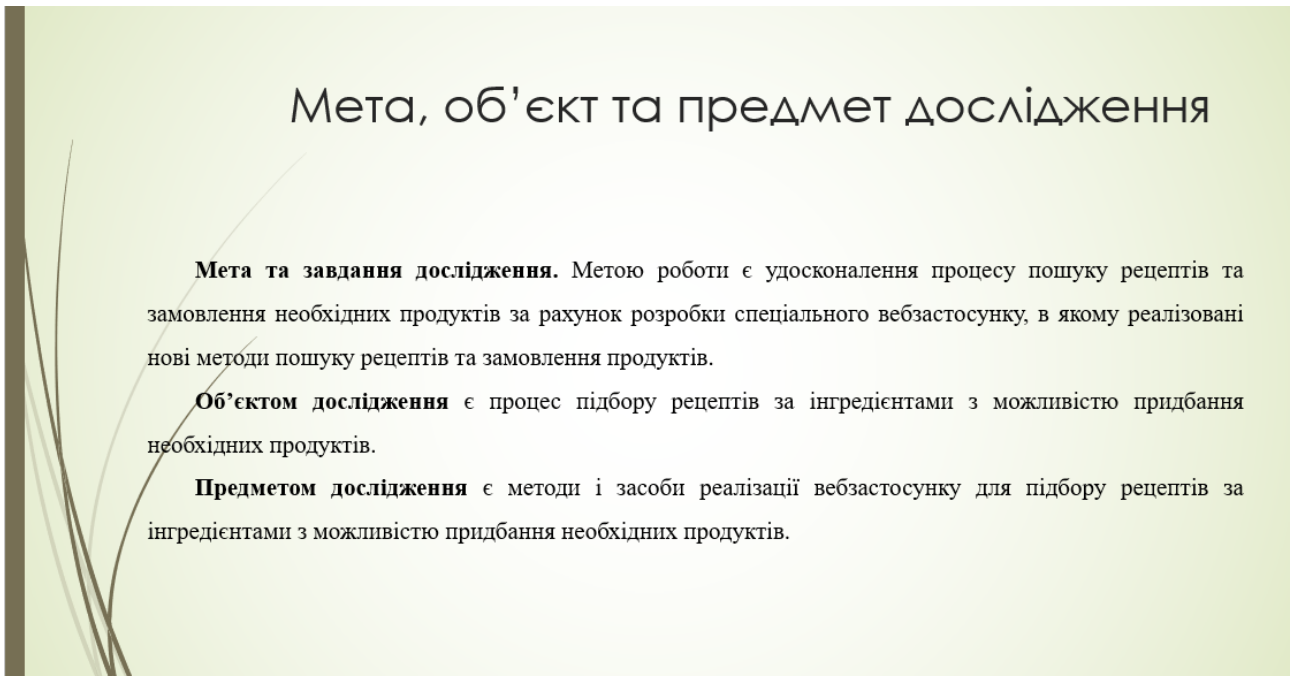

Рисунок Г.3 – Мета, об'єкт та предмет дослідження

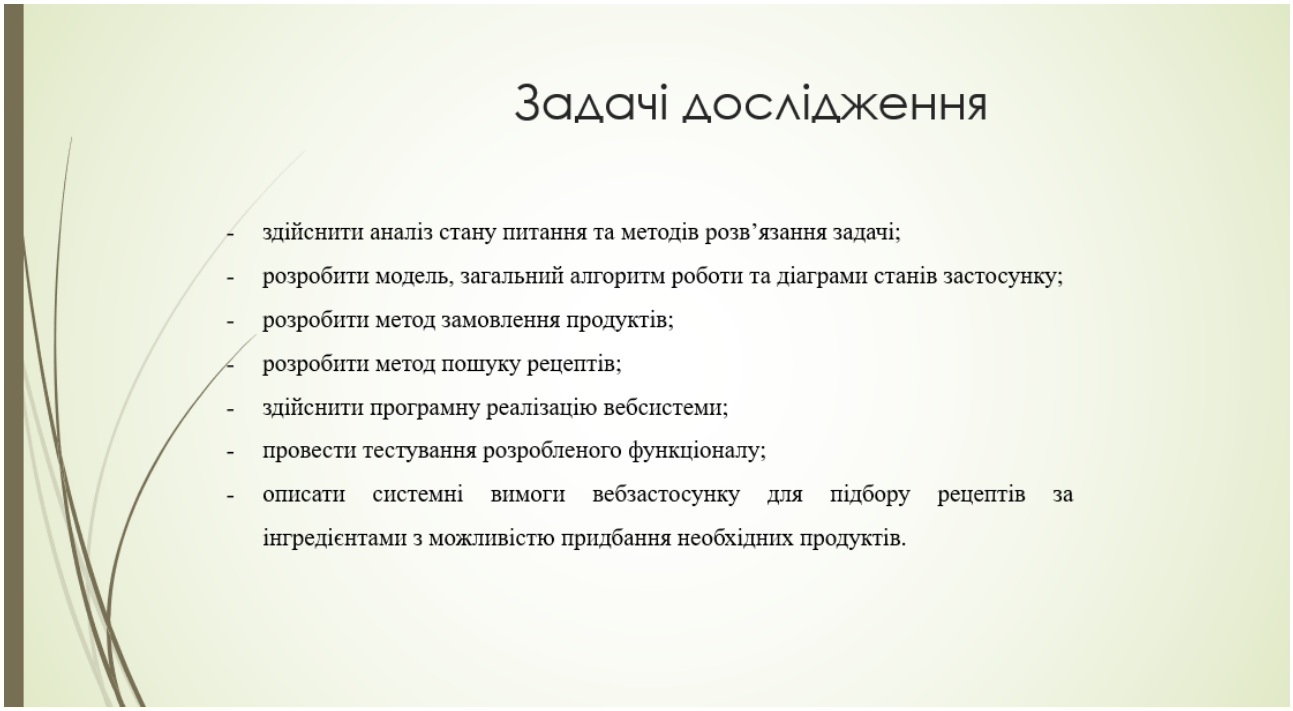

Рисунок Г.4 – Задачі дослідження

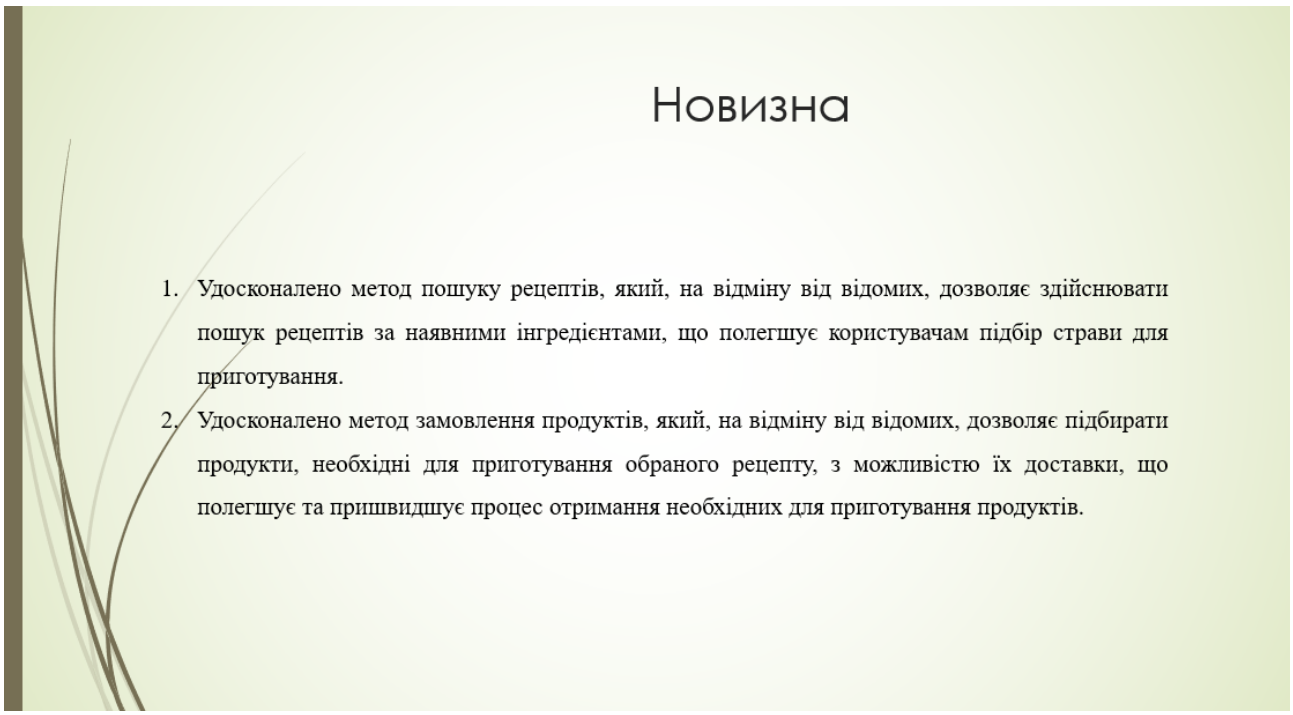

Рисунок Г.5 – Новизна

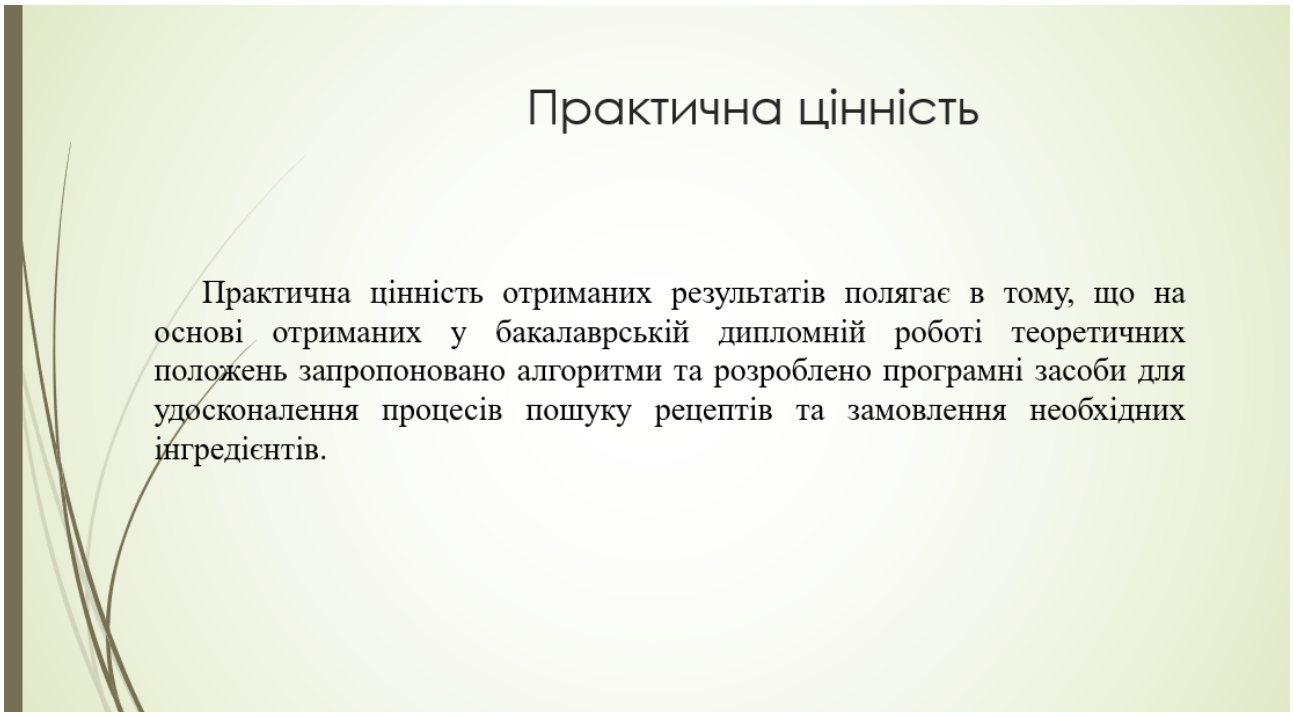

Рисунок Г.6 – Практична цінність

Рисунок Г.7 – Аналоги

Порівняльний аналіз аналогів

	Cookpad	Yummly	Mealime	Mealthy
Пошук рецептів	1	1	1	1
Збереження улюблених рецептів	1	1	1	1
Детальна інформація про рецепти	1	1	1	1
Список покупок	0	1	1	1
Замовлення продуктів з магазину	0	0	0	1
Сумарний бал	3	4	4	5

Рисунок Г.8 – Порівняльний аналіз аналогів

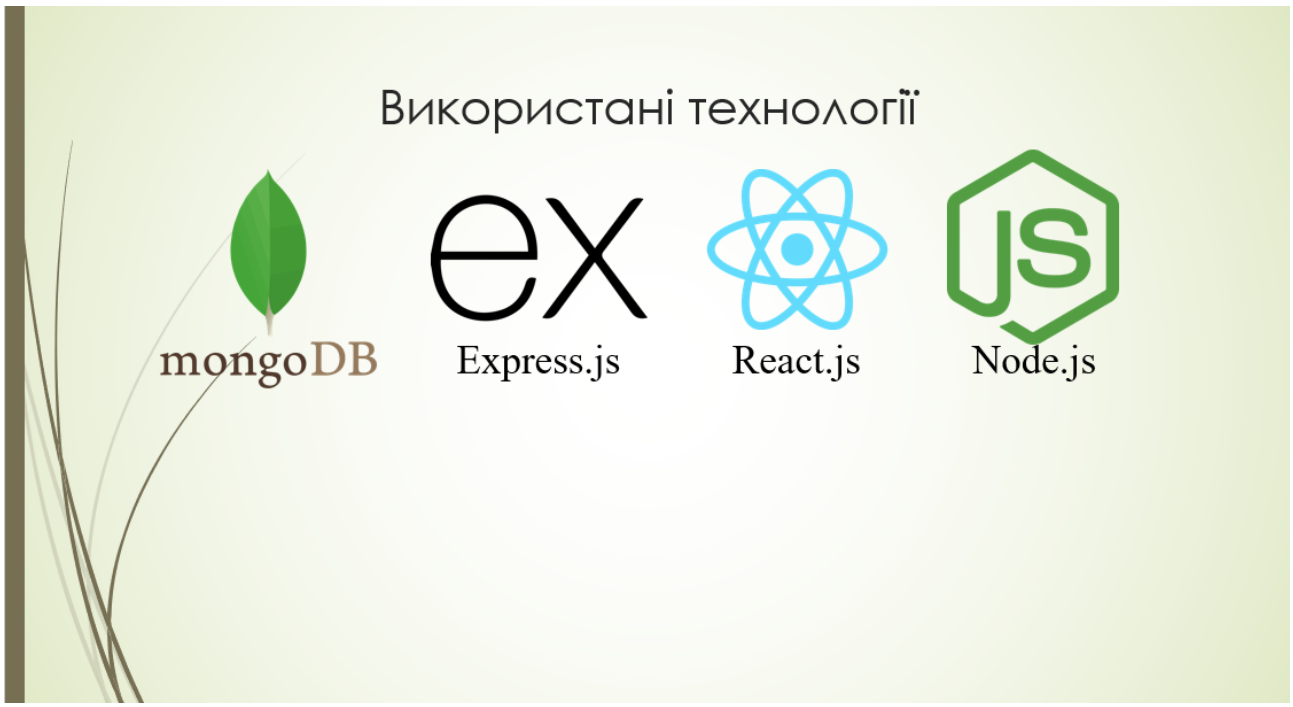

Рисунок Г.9 – Використані технології

Рисунок Г.10 – Метод пошуку рецептів

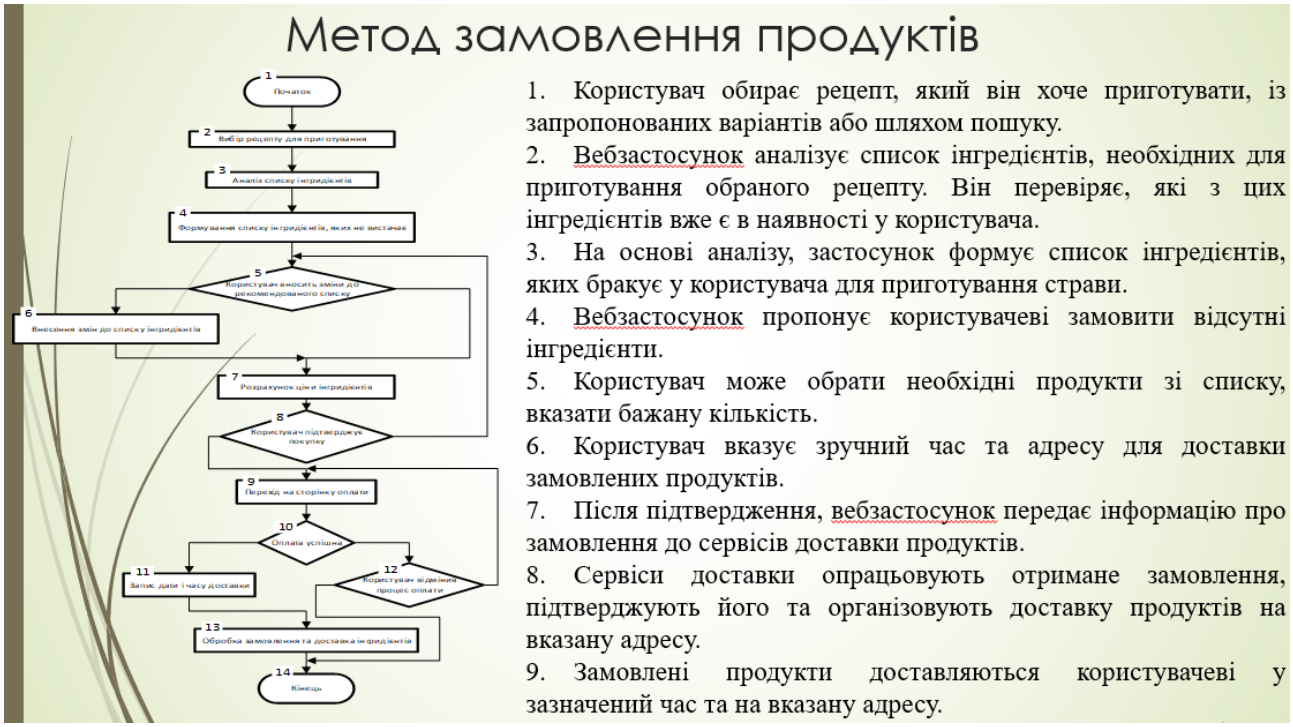

Рисунок Г.11 – Метод замовлення продуктів

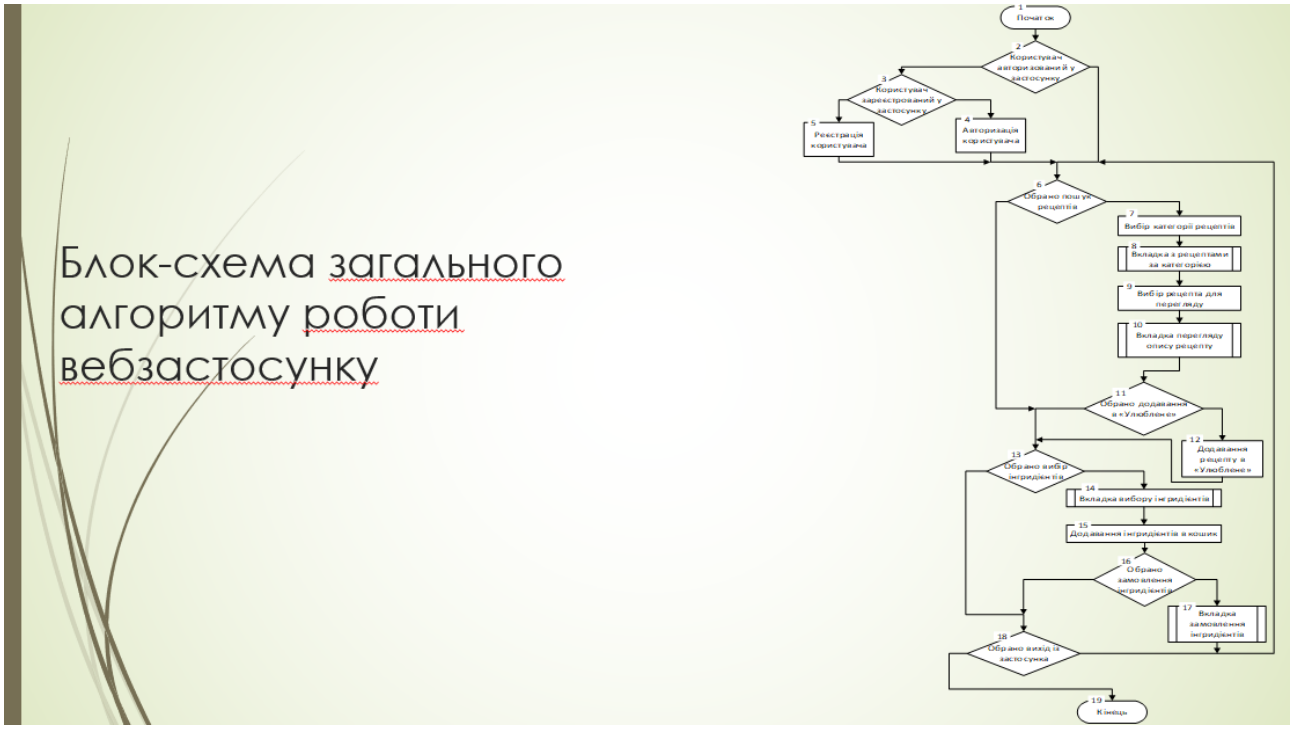

Рисунок Г.12 – Блок-схема загального алгоритму роботи застосунку

Рисунок Г.13 – Розробка вебзастосунку

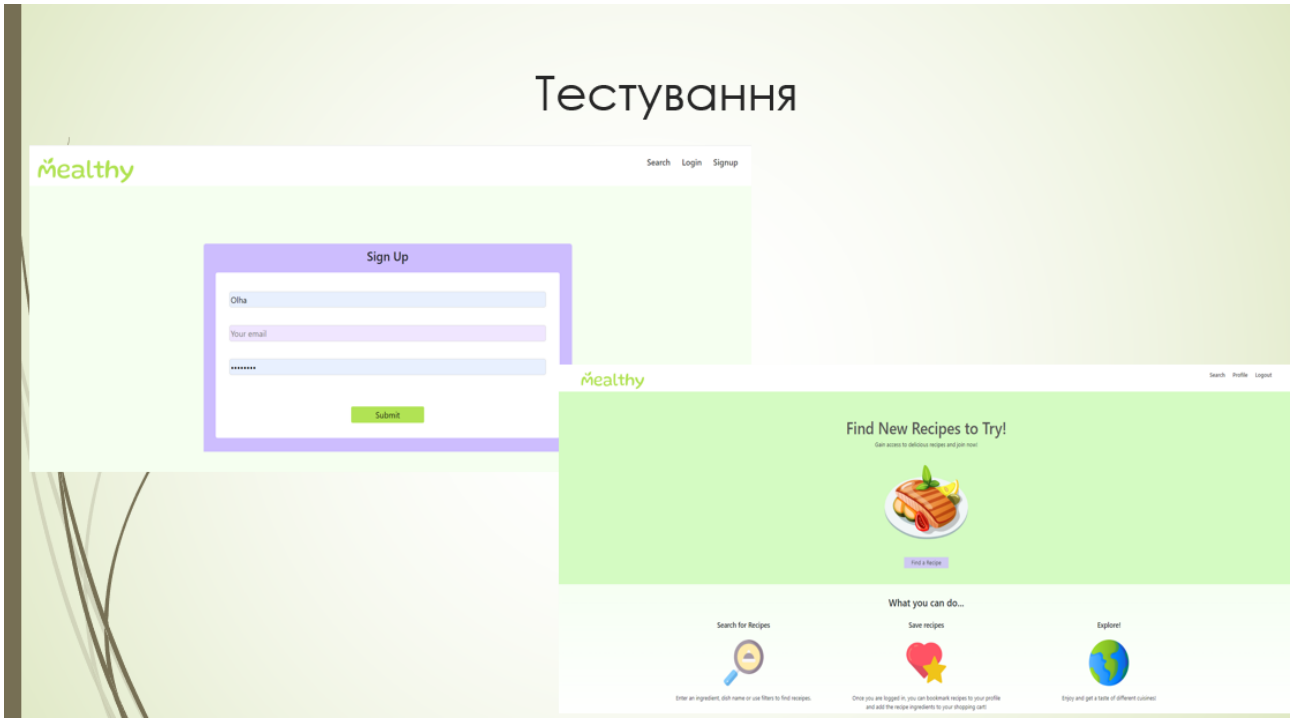

Рисунок Г.14 – Тестування вкладки реєстрації та головної вкладки вебзастосунку

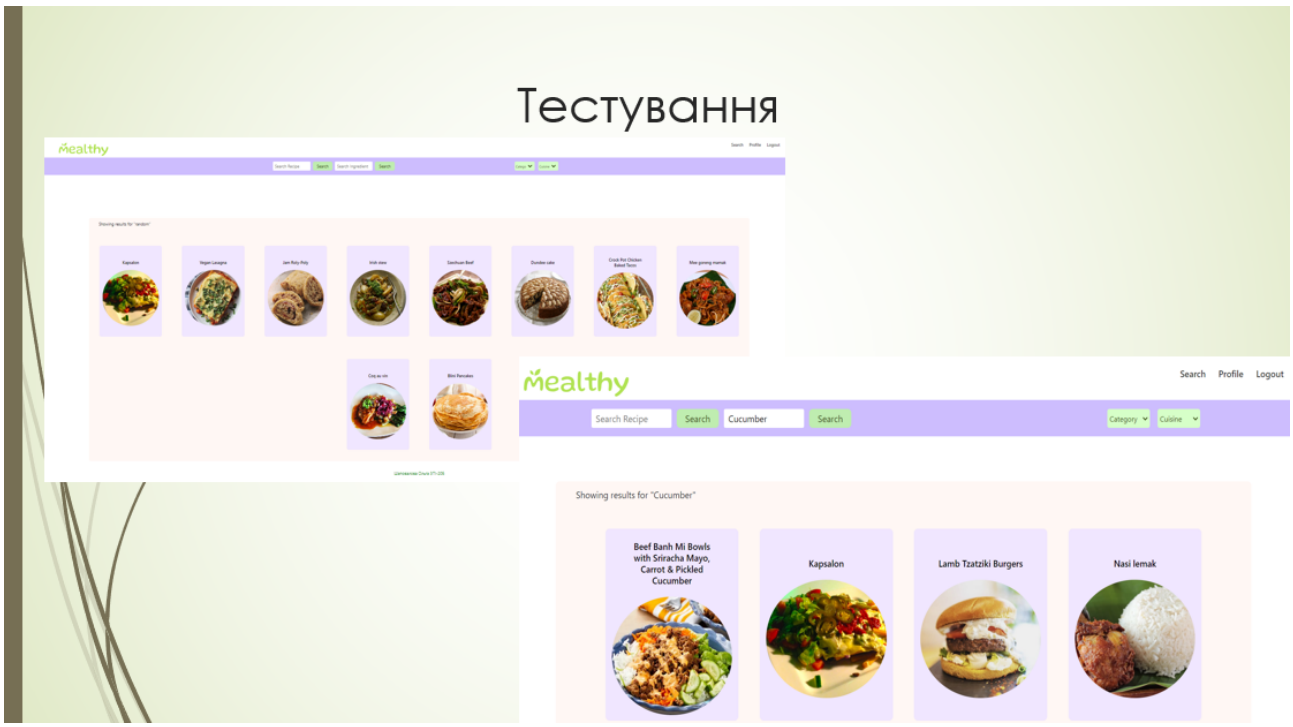

Рисунок Г.15 – Тестування вкладки пошуку рецептів

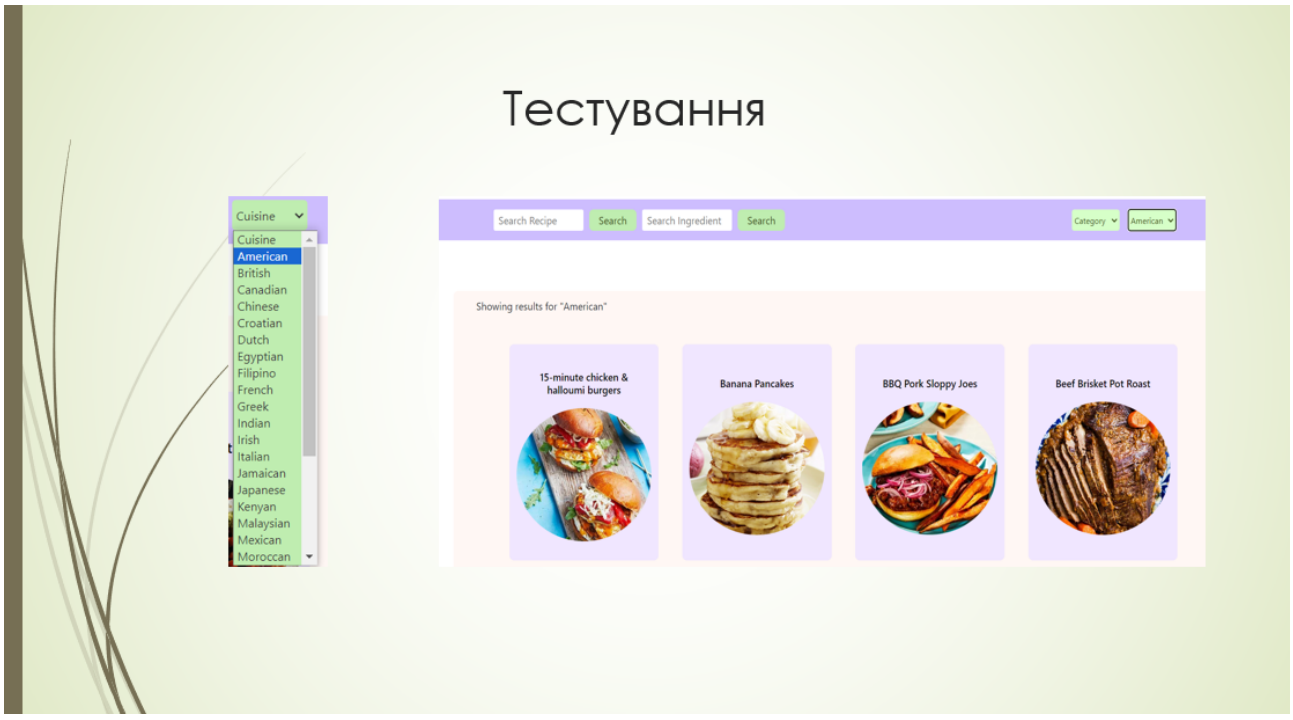

Рисунок Г.16 – Тестування вкладки пошуку рецептів за національними кухнями

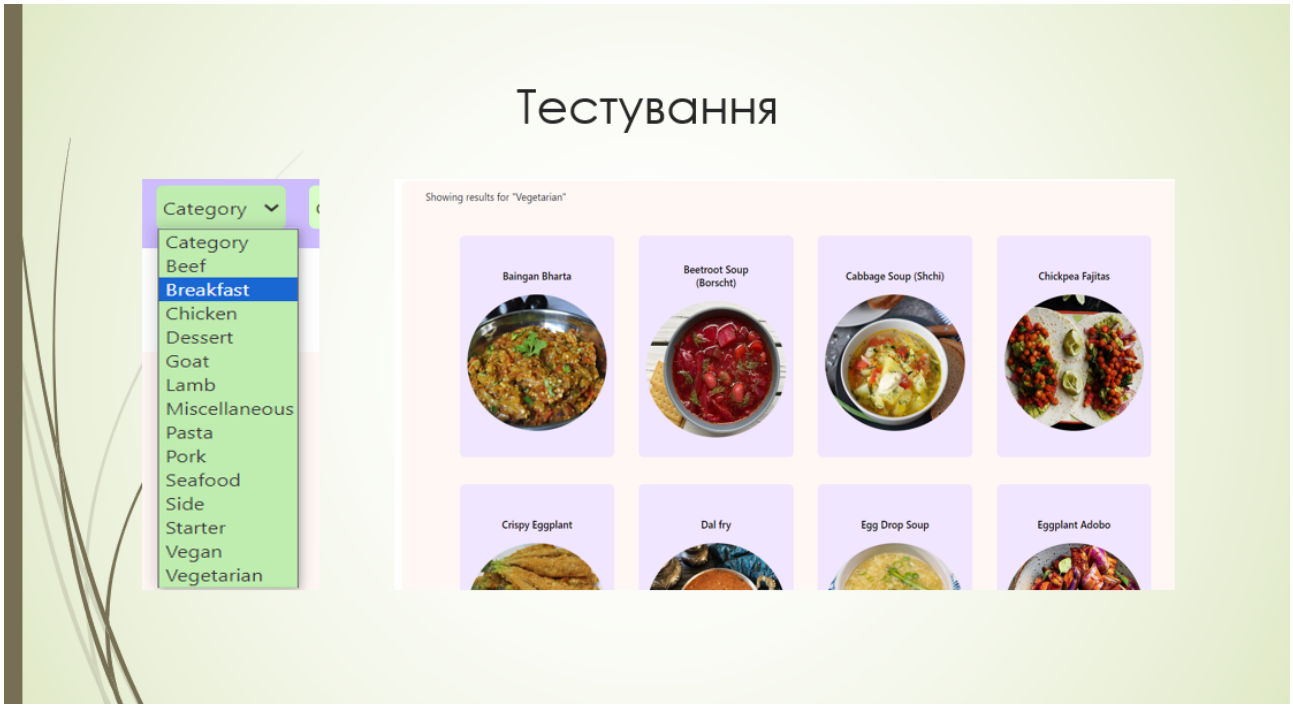

Рисунок Г.17 – Тестування вкладки пошуку рецептів за видами страв

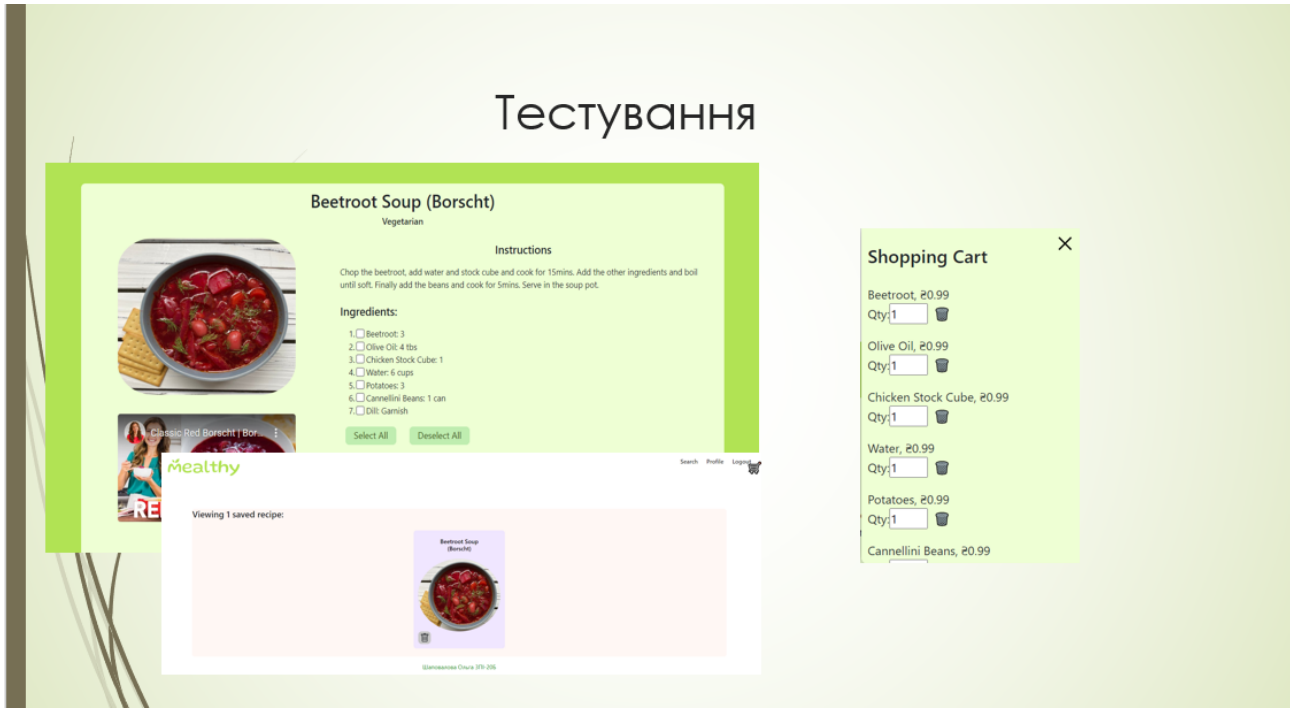

Рисунок Г.18 – Тестування вкладки перегляду рецепту та кошика для замовлення інгредієнтів

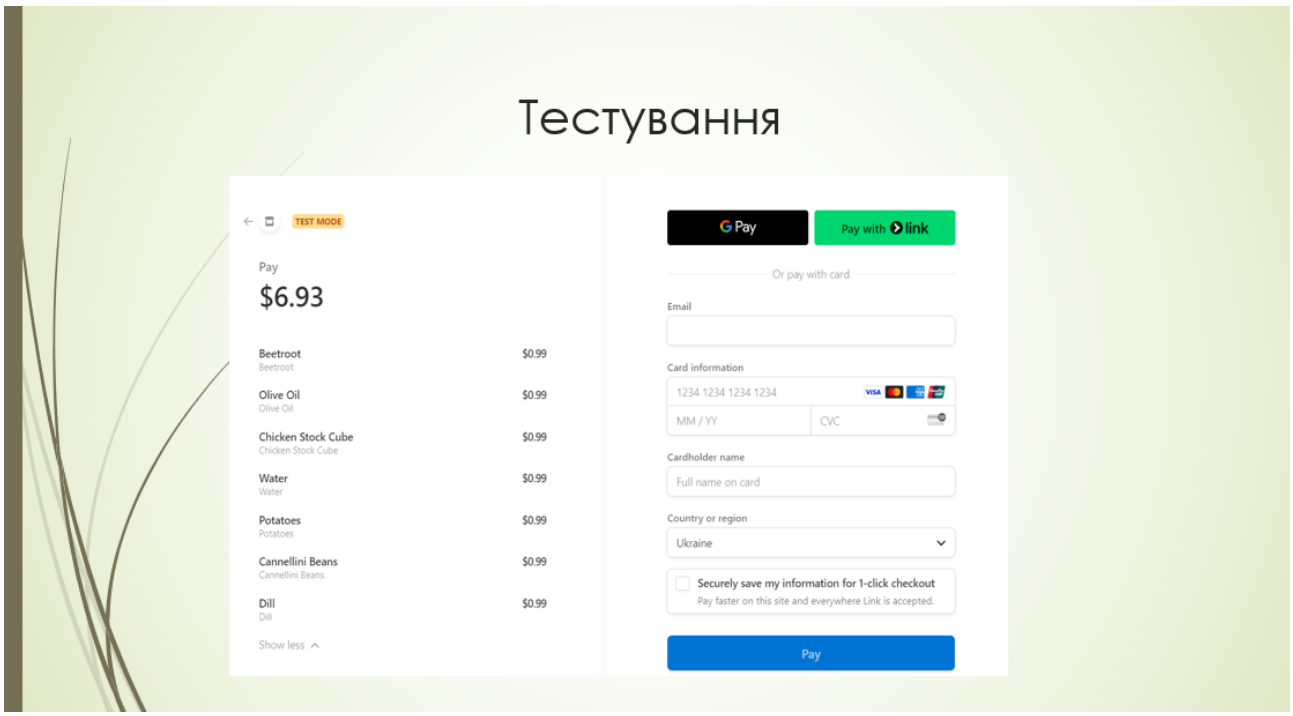

Рисунок Г.19 – Тестування вкладки покупки інгредієнтів

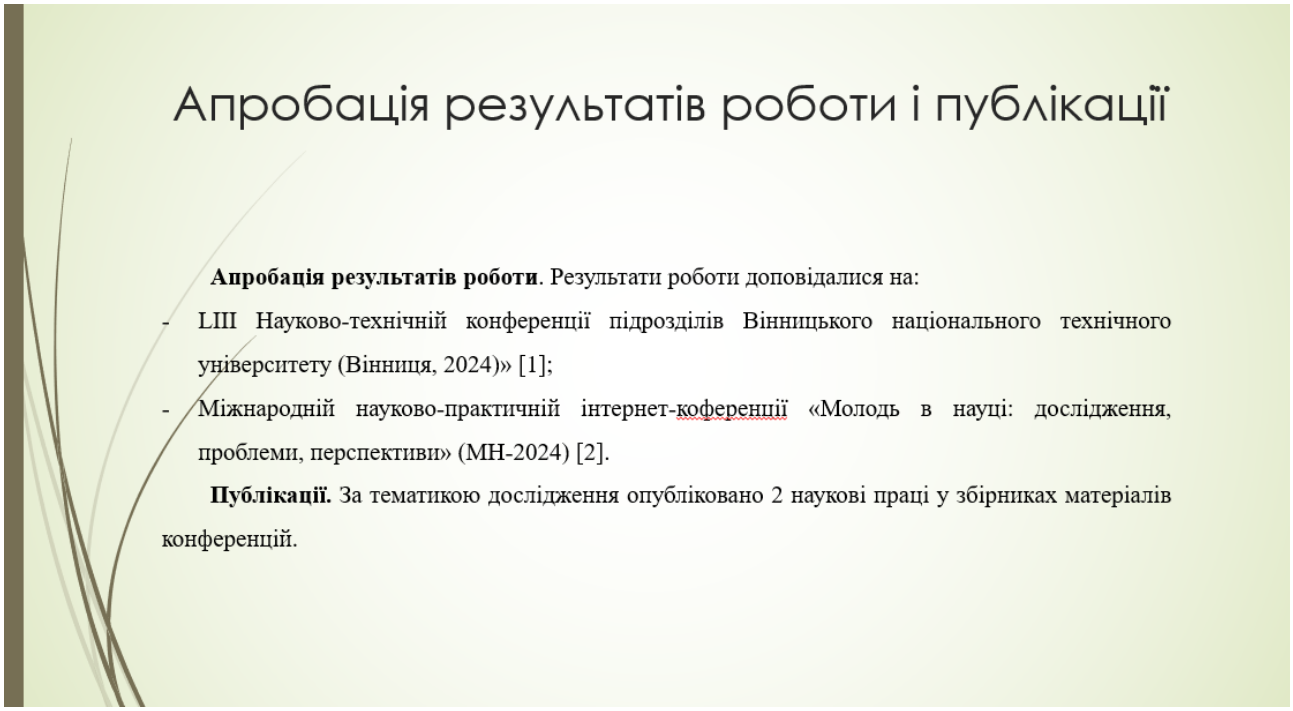

Рисунок Г.20 – Апробація результатів роботи і публікації