НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Кафедра інформатики та програмної інженерії
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2022 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення комп'ютеризованих систем»

спеціальності «121 Інженерія програмного забезпечення»

на тему: Сервіс оптимального вибору рецепту в умовах змінюваності заданих параметрів (Комплексна тема). Клієнтська частина веб-застосунку та розгортання.

Виконав студент IV курсу, групи ІП-82
(шифр групи)
Шабанов Бунямін Шаміль Огли _____
(прізвище, ім'я, по батькові) (підпис)

Керівник ст. вик., Ковтунець О.В. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з графічної документації доцент, к.т.н., доц., Ліщук К. І. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент доцент кафедри ІСТ, к.т.н., доц., Остапченко К. Б. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ –2022

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 Інженерія програмного забезпечення

Освітньо-професійна програма – Інженерія програмного забезпечення комп'ютеризованих систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2022 р.

ЗАВДАННЯ
на дипломний проєкт студенту

Шабанову Буняміну Шаміль Огли
(прізвище, ім'я, по батькові)

1. Тема проєкту Сервіс оптимального вибору рецепту в умовах змінюваності заданих параметрів (Комплексна тема). Клієнтська частина веб-застосунку та розгортання.

керівник проєкту Ковтунець Олесь Володимирович, ст. викладач
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «10» червня 2022 р. №1033-с

2. Термін подання студентом проєкту « 15 » червня 2022 року

3. Вихідні дані до проєкту: технічне завдання

4. Зміст пояснювальної записки
1) Аналіз вимог до програмного забезпечення: основні визначення та терміни, опис предметного середовища, огляд існуючих технічних рішень та відомих програмних продуктів.
2) Моделювання та конструювання програмного забезпечення: моделювання та аналіз програмного забезпечення, засоби розробки, технічні рішення, архітектура програмного забезпечення.
3) Аналіз якості програмного забезпечення.
4) Впровадження та супровід програмного забезпечення

5. Перелік графічного матеріалу _____
1) Креслення вигляду екранних форм

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «10» березня 2022 року _____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення рекомендованої літератури	14.03.2022	
2	Аналіз існуючих методів розв'язання задачі	21.03.2022	
3	Постановка та формалізація задачі	28.03.2022	
4	Розробка інформаційного забезпечення	05.04.2022	
5	Алгоритмізація задачі	12.04.2022	
6	Обґрунтування вибору використаних технічних засобів	19.04.2022	
7	Розробка програмного забезпечення	13.05.2022	
8	Налагодження програми	19.05.2022	
9	Виконання графічних документів	24.05.2022	
10	Оформлення пояснювальної записки	08.06.2022	
11	Подання ДП на попередній захист	10.06.2022	
12	Подання ДП рецензенту	12.06.2022	
13	Подання ДП на основний захист	19.06.2022	

Студент _____ Бунямін ШАБАНОВ
(підпис) (ініціали, прізвище)

Керівник _____ Олесь КОВТУНЕЦЬ
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

Пояснювальна записка дипломного проєкту складається з чотирьох розділів, містить 45 таблиць, 6 рисунків та 10 джерел – загалом 56 сторінки.

До бакалаврської роботи дипломної роботи Шабанова Буняміна Шаміль Огли на тему: «Сервіс оптимального вибору рецепту в умовах змінюваності заданих параметрів (Комплексна тема). Клієнтська частина веб-застосунку та розгортання».

Мета: розробка надійного якісного фронтенд-додатку. Додаток необхідний для користувачів платформи, що дозволяє шукати кулінарні рецепти за різними інгредієнтами та переглядати детальної інформації про рецепти та інгредієнти.

Об'єкт дослідження: програмне забезпечення для пошуку рецепту по заданих інгредієнтах.

Предмет дослідження: розробка та модифікація браузерного програмного забезпечення для взаємодії користувача з бекендом сервісу пошуку рецептів та продуктів, налаштування процесів розробки та доставки програмного забезпечення.

У першому розділі описано вимоги до програмного забезпечення, сформовано сценарії використання і задачу дипломного проєкту.

В другому розділі детально описані задачі веб-додатку, описано основні технології та мови програмування, використані при розробці, описано технології розгортання, що були використані для розгортання сервісу.

В третьому розділі аргументовано вибір шляхів тестування програми, наведено покрокові сценарії тестування.

У четвертому розділі проінструктоване покрокове розгортання програмного забезпечення.

Програмне забезпечення розгорнуто на віддаленому сервері платформи Vercel.

КЛЮЧОВІ СЛОВА: ВЕБ-ДОДАТОК, FRONTEND, JAVASCRIPT, NEXTJS, GOOGLE MAPS API.

ABSTRACT

The explanatory note of the diploma project consists of four sections, contains 45 tables, 6 figures and 10 sources - a total of 56 pages.

To the bachelor's thesis of Shabanov Bunyamin on the topic: «Service of the optimal choice of the recipe in the conditions of variability of the set parameters (Complex theme). The client part of the web application and deployment».

Purpose: development of a reliable high-quality frontend application. The application is necessary for users of the platform, which allows you to search for recipes by various ingredients and view detailed information about recipes and ingredients.

Object of research: software for finding a recipe for a given ingredient.

Subject of research: development and modification of browser software for user interaction with the backend of the service of searching for recipes and products, setting up the processes of software development and delivery.

The first section describes the requirements for the software, generated usage scenarios and the task of the thesis project.

The second section describes in detail the tasks of the web application, describes the main technologies and programming languages used in the development, describes the deployment technologies that were used to deploy the service.

The third section argues the choice of ways to test the program, provides step-by-step testing scenarios.

The fourth section instructs step-by-step software deployment.

The software is deployed on a remote Vercel server.

KEYWORDS: WEBAPPLICATION, FRONTEND, JAVASCRIPT, NEXTJS, GOOGLE MAPS API

Пояснювальна записка
до дипломного проєкту

на тему: Сервіс оптимального вибору рецепту в умовах змінюваності заданих параметрів (Комплексна тема). Клієнтська частина веб-застосунку та розгортання.

КПІ.ІП-8134.045490.02.81

Київ – 2022

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ 3
ВСТУП 4
1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 5
1.1 Загальні положення 5
1.2 Постановка задачі 16
Висновки до розділу 16
2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 17
2.1 Моделювання та аналіз програмного забезпечення 17
2.2 Архітектура програмного забезпечення 23
Висновки до розділу 44
3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 45
3.1 Аналіз якості ПЗ 45
3.2 Опис процесів тестування 46
3.3 Опис контрольного прикладу 46
Висновки до розділу 51
4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ . 53
4.1 Розгортання програмного забезпечення 53
4.2 Робота з програмним забезпеченням 53
ВИСНОВКИ 54
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 55

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API	–	Application programming interface, прикладний програмний Інтерфейс..
IT	–	Інформаційні технології.
ОС	–	Операційна система.
БД(DB)	–	База даних(database).
BPMN	–	Business Process Model and Notation, модель та нотація бізнес-процесів.
WASM	–	WebAssembly, низькорівневий проміжний код для виконання в браузері застосунків, скомпільованих з різних мов програмування.
DOM	–	Document Object Model, об’єктна модель документу.
CI/CD	–	Continuous Integration and Continuous Development, безперервна інтеграція та безперервна розробка.
PaaS	–	Platform as a Service, платформа в якості сервісу

ВСТУП

Сучасна сфера розробки програмного забезпечення для веб-платформи надає безліч варіантів інструментів та технологій для розробки, тестування та розгортання додатків. Не дивлячись на те, що різноманіття мов для розробки фронтенд додатків скоротилось через суттєве домінування мови Javascript, все одно навіть в рамках однієї мови важко обрати правильній фреймворк чи бібліотеку. Такий вибір — велика відповідальність, адже ніколи не знаєш як довго проєкт розвиватиметься та як довго технологія буде актуальною. Слід, звичайно, зазначити, що, не дивлячись на домінування однієї мови програмування, сучасні браузери дозволяють використовувати інші мови програмування, що компілюються, через технологію WASM, тобто WebAssembly.

Серед сучасних технологій розробки фронтенд-додатків можна назвати декілька різних груп:

– фреймворки та сімейства бібліотек на Javascript(Angular, React, Vue, Svelte, WASP та інші);

– бібліотеки та препроцесори для CSS (Sass, Less, Bulma);

– шаблонізатори для HTML(ejs, handlebars, Mustache, Pug(в минулому Jade));

– різноманітні рішення на інших мовах програмування для роботи в WASM (наприклад yew на мові Rust).

Метою виконання цієї роботи є розробка надійного якісного фронтенд-додатку. Додаток необхідний для користувачів платформи, що дозволяє шукати кулінарні рецепти за різними інгредієнтами та переглядати детальної інформації про рецепти та інгредієнти. Основними задачами роботи є:

– вивчити можливі варіанти розробки програмного коду;

– сформувати вимоги для майбутнього продукту;

– розробити програмний код;

– розгорнути код на віддаленому сервері задля широкого доступу.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

Для розроблення коректного працюючого додатку необхідно сформувати набір варіантів використання, функціональних та нефункціональних вимог.

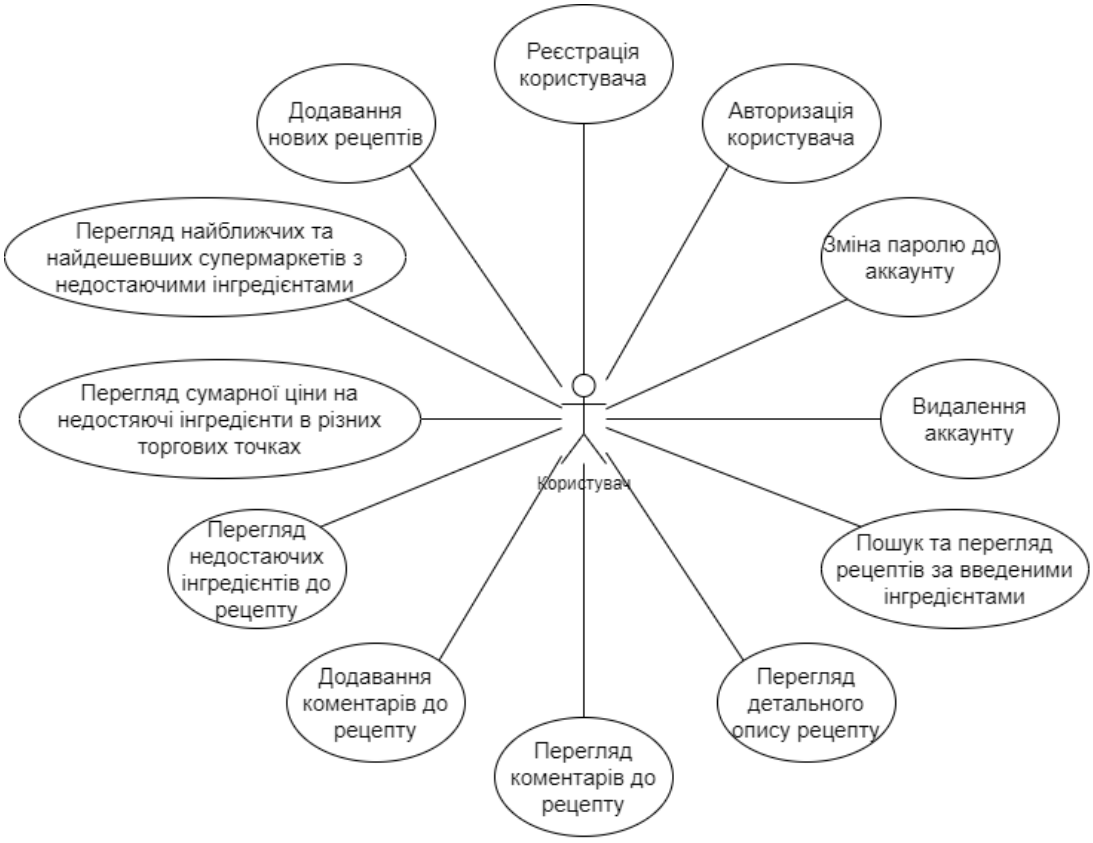


Рисунок 3.1 — Схема варіантів використання програмного забезпечення

1.1.1 Розроблення функціональних вимог

Таблиця 3.1 — Варіант використання UC001

Назва	Реєстрація користувача
Опис	Користувач сервісу має можливість отримати акаунт в системі шляхом надання логіну і паролю
Учасники	Користувач веб-додатку

Продовження таблиці 3.1

Передумови	Неавторизований користувач
Післяумови	Користувач отримує власний акаунт, яким може керувати
Основний сценарій	1) Користувач відкриває веб-додаток. 2) Користувач переходить на сторінку надання даних для реєстрації. 3) Користувач вводить електронну пошту, пароль та підтверджує ввід. 4) Користувач автоматично авторизовується та перенаправляється на головну сторінку.

Таблиця 3.2 — Варіант використання UC002

Назва	Авторизація користувача
Опис	Користувач має можливість зайти в свій попередньо створений акаунт(а також вийти з нього)
Учасники	Користувач веб-додатку
Передумови	Неавторизований користувач
Післяумови	Користувач отримує доступ до власного акаунту та до привелегій володіння акаунтом у системі
Основний сценарій	1) Користувач відкриває веб-додаток. 2) Користувач переходить на сторінку надання даних для авторизації. 3) Користувач вводить електронну пошту, пароль та підтверджує ввід. 4) Користувач авторизується та перенаправляється на головну сторінку.

Таблиця 3.3 — Варіант використання UC003

Назва	Зміна паролю до аккаунту
Опис	Користувач має можливість змінити дані для авторизації, а саме пароль
Учасники	Користувач веб-додатку
Передумови	Користувач авторизований
Післяумови	Користувач авторизований з новий паролем
Основний сценарій	1) Користувач переходить на сторінку керування акаунтом. 2) Користувач вводить старий пароль, новий пароль та підтверджує ввід. 3) Користувач бачить повідомлення про успішну зміну паролю.

Таблиця 3.4 — Варіант використання UC004

Назва	Видалення аккаунту
Опис	Користувач має змогу назавжди видалити свій акаунт з системи
Учасники	Користувач веб-додатку
Передумови	Користувач авторизований
Післяумови	Користувач не авторизований та більше не має змоги авторизуватись зі старими вхідними даними
Основний сценарій	1) Користувач переходить на сторінку керування акаунтом. 2) Користувач натискає на кнопку видалення акаунту. 3) Користувач підтверджує видалення вводом паролю від акаунту. 4) Користувач бачить повідомлення про успішну видалення акаунту та автоматично деавторизується.

Таблиця 3.5 — Варіант використання UC005

Назва	Пошук та перегляд рецептів за введеними інгредієнтами
Опис	Користувач має можливість переглядати короткі відомості про кулінарні рецепти що містять необхідні інгредієнти
Учасники	Користувач веб-додатку
Передумови	Введені наявні інгредієнти
Післяумови	Користувач бачить список рецептів
Основний сценарій	1) Користувач вводить необхідні інгредієнти в поле вводу та підтверджує ввід. 2) Користувачу демонструється список рецептів з зображеннями, короткими списками інгредієнтів та посиланнями на повний опис кожного рецепту.

Таблиця 3.6 — Варіант використання UC006

Назва	Перегляд детального опису рецепту
Опис	Користувач має можливість переглядати повну інформацію про рецепт у відповідному вікні
Учасники	Користувач веб-додатку
Передумови	Наявне посилання на рецепт або відкритий список рецептів
Післяумови	Відкрита сторінка з деталями рецепту
Основний сценарій	1) Користувач відкриває посилання на рецепт або відкриває посилання зі сторінки пошуку рецептів 2) Користувач переглядає детальну інформацію за рецептом

Таблиця 3.7 — Варіант використання UC007

Назва	Перегляд коментарів до рецепту

Продовження таблиці 3.7

Опис	Користувач має змогу переглядати коментарі, залишені іншими користувачами
Учасники	Користувач веб-додатку
Передумови	Наявне посилання на рецепт або відкритий список рецептів
Післяумови	Відкрита сторінка з деталями рецепту
Основний сценарій	1) Користувач відкриває посилання на рецепт або відкриває посилання зі сторінки пошуку рецептів 2) Користувач переглядає детальну інформацію за рецептом 3) Під деталями рецепту користувач бачить список коментарів, що включають в себе ім'я користувача, що залишив коментар та власне текст коментаря.

Таблиця 3.8 — Варіант використання UC008

Назва	Додавання коментарів до рецепту
Опис	Користувач має можливість залишати власні коментарі до наявних рецептів
Учасники	Користувач веб-додатку
Передумови	Відкрита сторінка з деталями рецепту, користувач авторизований
Післяумови	Рецепт містить коментар користувача
Основний сценарій	1) Користувач відкриває посилання на рецепт або відкриває посилання зі сторінки пошуку рецептів 2) Користувач переглядає детальну інформацію за рецептом 3) Під деталями рецепту користувач вводить новий коментар та підтверджує збереження коментаря. 4) Користувач бачить збережений коментар під переглядаємим рецептом.

Таблиця 3.9 — Варіант використання UC009

Назва	Перегляд недостаючих інгредієнтів до рецепту
Опис	Користувач має змогу побачити яких інгредієнтів йому не вистачає для приготування страви
Учасники	Користувач веб-додатку
Передумови	Введені інгредієнти
Післяумови	Наявна сторінка рецепту з кольоровим списком інгредієнтів
Основний сценарій	1) Користувач відкриває посилання на рецепт або відкриває посилання зі сторінки пошуку рецептів, попередньо ввівши список наявних інгредієнтів. 2) Користувач переглядає детальну інформацію за рецептом 3) Під деталями рецепту користувач бачить список інгредієнтів для рецепту, серед яких червоним виділені ті, які не були використані під час пошуку цього рецепту.

Таблиця 3.10 — Варіант використання UC0010

Назва	Перегляд сумарної ціни на недостаючі інгредієнти в різних торгових точках
Опис	Користувач може переглядати список торгових точок поблизу та ціну кошика з недостаючими інгредієнтами
Учасники	Користувач веб-додатку
Передумови	Введені інгредієнти

Продовження таблиці 3.10

Післяумови	Наявна сторінка рецепту зі списком торгових точок
Основний сценарій	1) Користувач відкриває посилання на рецепт або відкриває посилання зі сторінки пошуку рецептів, попередньо ввівши список наявних інгредієнтів. 2) Користувач переглядає детальну інформацію за рецептом 3) Під деталями рецепту користувач бачить список назв торгових точок та вартості кошиків в цих торгових точках, де кошики це набір недостаючих інгредієнтів, що можуть бути куплені в відповідному супермаркеті

Таблиця 3.11 — Варіант використання UC0011

Назва	Перегляд найближчих та найдешевших супермаркетів з недостаючими інгредієнтами
Опис	Користувач має можливість переглядати на інтерактивній мапі найближчі супермаркети у своєму регіоні
Учасники	Користувач веб-додатку
Передумови	Відкрита сторінка з деталями рецепту

Продовження таблиці 3.11

Післяумови	Відкрита сторінка з інтерактивною мапою
Основний сценарій	1) Користувач відкриває посилання на рецепт або відкриває посилання зі сторінки пошуку рецептів. 2) Після завантаження сторінки користувач натискає кнопку перегляду мапи. 3) Веб-додаток відкриває інтерактивну мапу, на якій позначками відображені найближчі супермаркети, а також вартості кошиків в цих супермаркетах. 4) Додатково виділяються особливими позначками два типи супермаркетів: найближчий до користувача та найдешевший за вартістю кошика.

Таблиця 3.12 — Варіант використання UC0012

Назва	Додавання нових рецептів
Опис	Користувач має можливість додавати і зберігати нові кулінарні рецепти
Учасники	Користувач веб-додатку

Продовження таблиці 3.12

Передумови	Користувач авторизований
Післяумови	Новий рецепт доступний для перегляду за посиланням або під час пошуку
Основний сценарій	1) Користувач відкриває головну сторінку веб додатку та авторизується(за необхідністю — реєструється). 2) На головній сторінці користувач натискає кнопку додавання нового рецепту, після чого веб-додаток відображає вікно для введення необхідних даних. 3) Користувач вводить дані нового рецепту, а саме: назва, список інгредієнтів, детальний процес приготування та зображення(за бажанням). 4) Користувач підтверджує створення нового рецепту. 5) Веб додаток зберігає інформацію про новий рецепт та відображає сторінку з новим рецептом на посиланням на нього. 6) В якості тестування користувач вводить в поле пошуку інгредієнти, що були використані в новому рецепті. 7) Користувач бачить свій рецепт серед перших у списку результатів пошуку, точна позиція може змінюватись через інші схожі рецепти.

Виходячи з вищеописаних варіантів використання можна сформувати наступні функціональні вимоги:

Таблиця 3.13 — Опис функціональної вимоги REQ001

Номер	REQ001

Продовження таблиці 3.13

Назва	Реєстрація нового користувача, авторизація та маніпуляції над аккаунтом
Опис	Додаток має надавати графічні інтерфейси для реєстрації користувача, авторизації користувача та редагування паролю облікового запису

Таблиця 3.14 — Опис функціональної вимоги REQ002

Номер	REQ002
Назва	Пошук рецептів за введеними інгредієнтами
Опис	Додаток має надавати графічний інтерфейс для введення списку інгредієнтів та надсилання запиту на пошук рецептів

Таблиця 3.15 — Опис функціональної вимоги REQ003

Номер	REQ003
Назва	Перегляд рецептів
Опис	Додаток має надавати графічний інтерфейс для перегляду знайдених за введеними параметрами рецептів

Таблиця 3.16 — Опис функціональної вимоги REQ004

Номер	REQ004

Продовження таблиці 3.16

Назва	Перегляд та додавання коментарів до рецептів
Опис	Додаток має надавати інтерфейс для введення користувачем нового коментаря до рецепту та перегляду вже введених іншими користувачами коментарів

Таблиця 3.17 — Опис функціональної вимоги REQ005

Номер	REQ005
Назва	Додавання нових рецептів
Опис	Додаток має надавати графічний інтерфейс для збереження нових рецептів.

Таблиця 3.18 — Опис функціональної вимоги REQ006

Номер	REQ006
Назва	Перегляд недостаючих інгредієнтів до рецептів
Опис	Додаток має надавати графічно зображену інформацію про неспівпадіння введеного списку інгредієнтів зі списком в переглядаємому рецепті

Таблиця 3.18 — Опис функціональної вимоги REQ007

Номер	REQ007

Продовження таблиці 3.18

Назва	Перегляд мапи з супермаркетами та цінами в них
Опис	Додаток має надавати графічний інтерфейс для перегляду супермаркетів з необхідними для покупки інгредієнтами на інтерактивній мапі

1.1.2 Розроблення нефункціональних вимог

Веб-додаток повинен відповідати наступним не функціональним вимогам:

– працювати у веб-браузері Chrome версії 49 та вище;

– мати інтерфейс англійською мовою;

– мати адаптивний інтерфейс (наповнення повинно бути доступним користувачеві на маленьких і великих екранах).

1.2 Постановка задачі

Метою виконання цієї дипломної роботи є розробка сучасного фронтенд-додатку. Цей додаток надаватиме користувачу веб-інтерфейс для пошуку кулінарних рецептів за різними інгредієнтами та перегляду детальної інформації про рецепти та інгредієнти.

Основними задачами для досягнення наведеної мети є:

– вивчити варіанти розробки програмного коду;

– сформувати вимоги для коду продукту;

– розробити програмний код;

– розгорнути код на віддаленому сервері задля широкого доступу.

Висновки до розділу

Сформувавши функціональні та нефункціональні вимоги можна почати моделювання архітектури та розробку додатку.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Перед початком обрання технологій та інструментів і реалізації програмного забезпечення необхідно сформувати моделі поведінки користувача, або моделі бізнес процесів. Для цього необхідно виділити основні задачі, які вирішуватиме веб-додаток. Серед задач виділимо наступні:

– необхідна можливість авторизуватись, маючи обліковий запис, для збереження в системі власних кулінарних рецептів і для коментування рецептів, процес детально описаний на рисунку 2.1.1;

– можливість шукати та переглядати список рецептів за введеними інгредієнтами, процес детально описаний на рисунку 2.1.2;

– користувач матиме можливість переглядати деталі конкретного рецепту а також переглядати кошики з інгредієнтами на інтерактивній мапі, процес детально описаний на рисунку 2.1.3;

– можливість залишити коментар під відкритим рецептом, процес детально описаний на рисунку 2.1.4;

– можливість зберегти інформацію про новий рецепт в системі на переглянути його пізніше, процес детально описаний на рисунку 2.1.5.

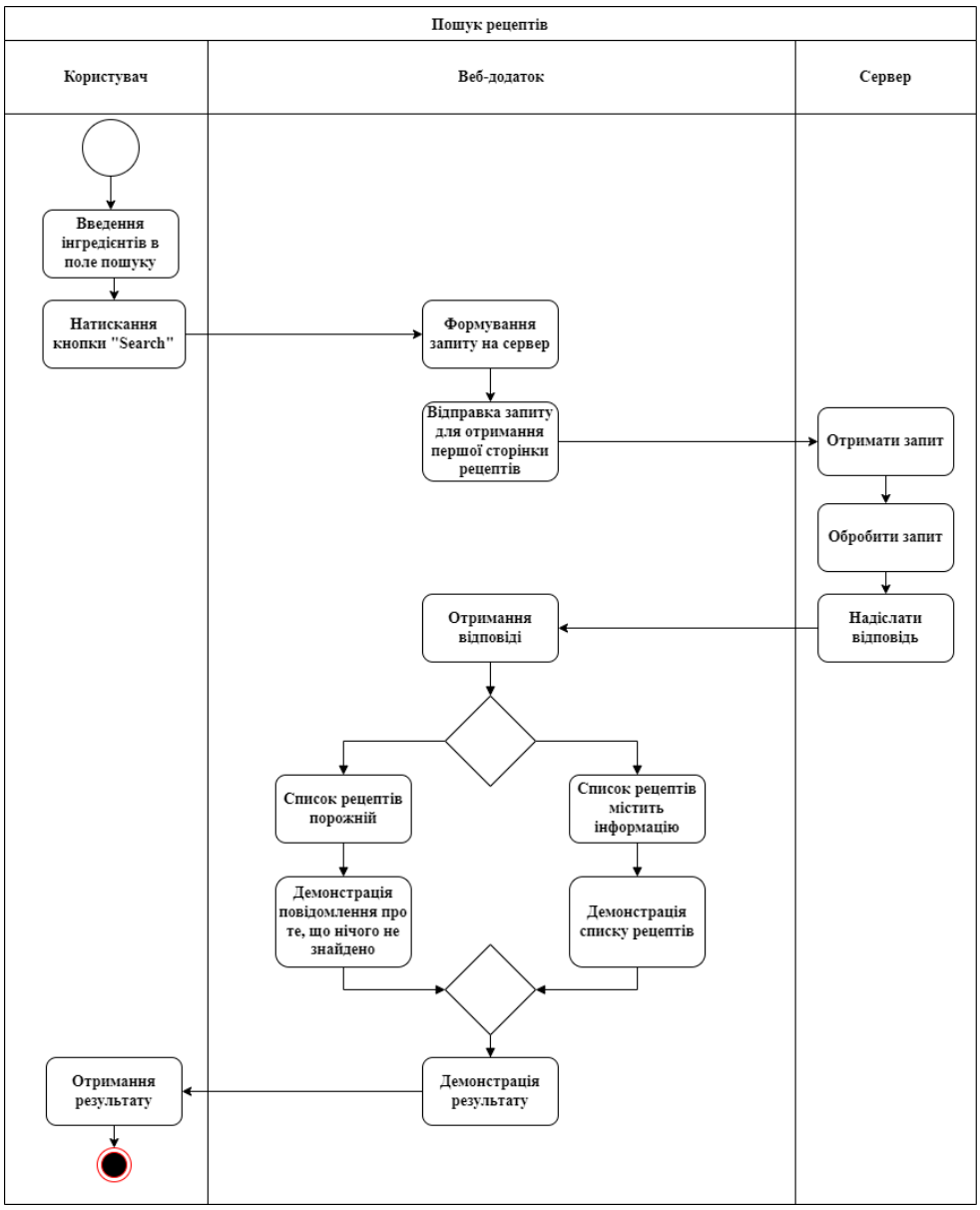


Рисунок 2.1.1 — Діаграма BPMN001

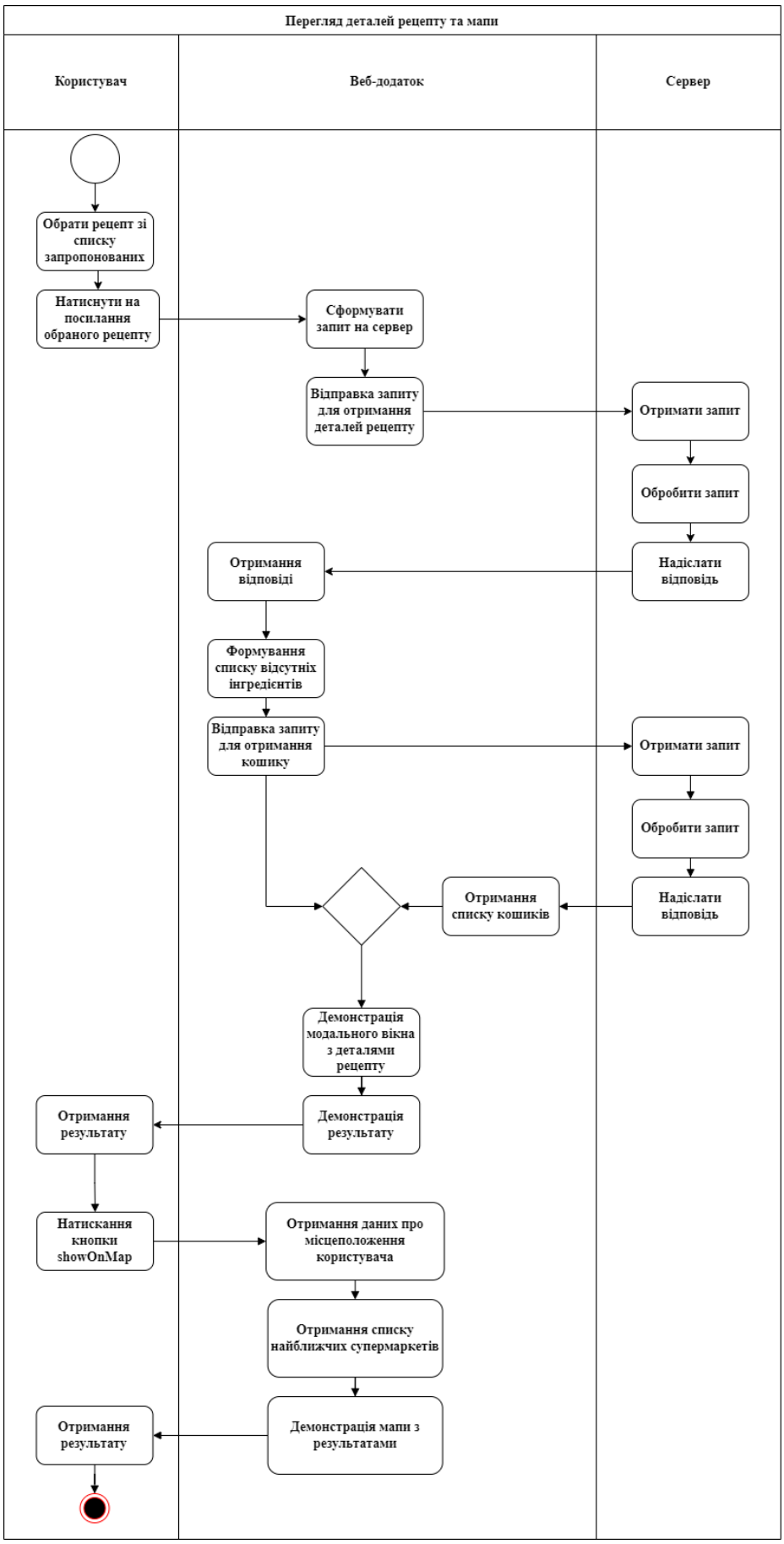


Рисунок 2.1.2 — Діаграма BPMN002

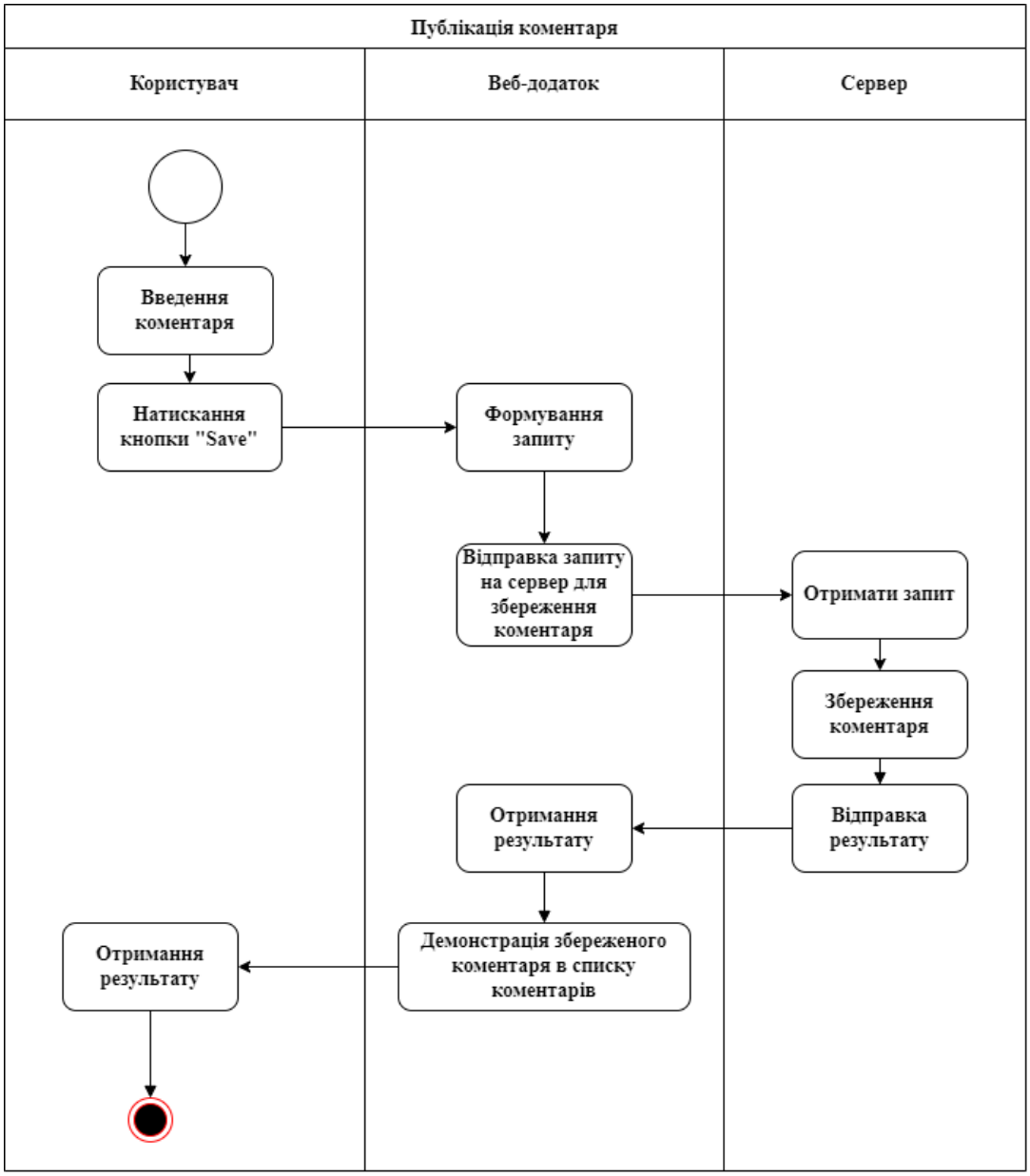


Рисунок 2.1.3 — Діаграма BPMN003

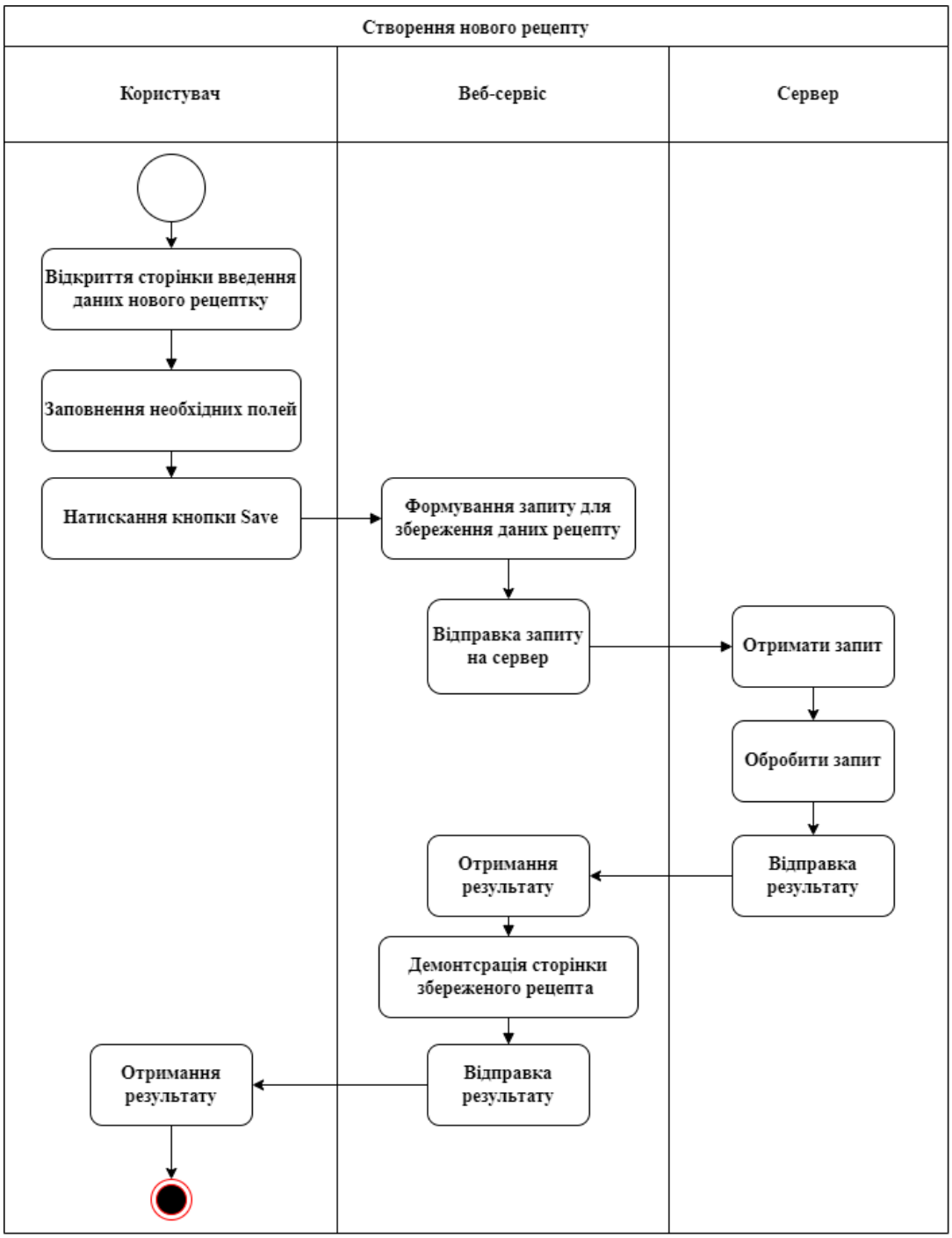


Рисунок 2.1.4 — Діаграма BPMN004

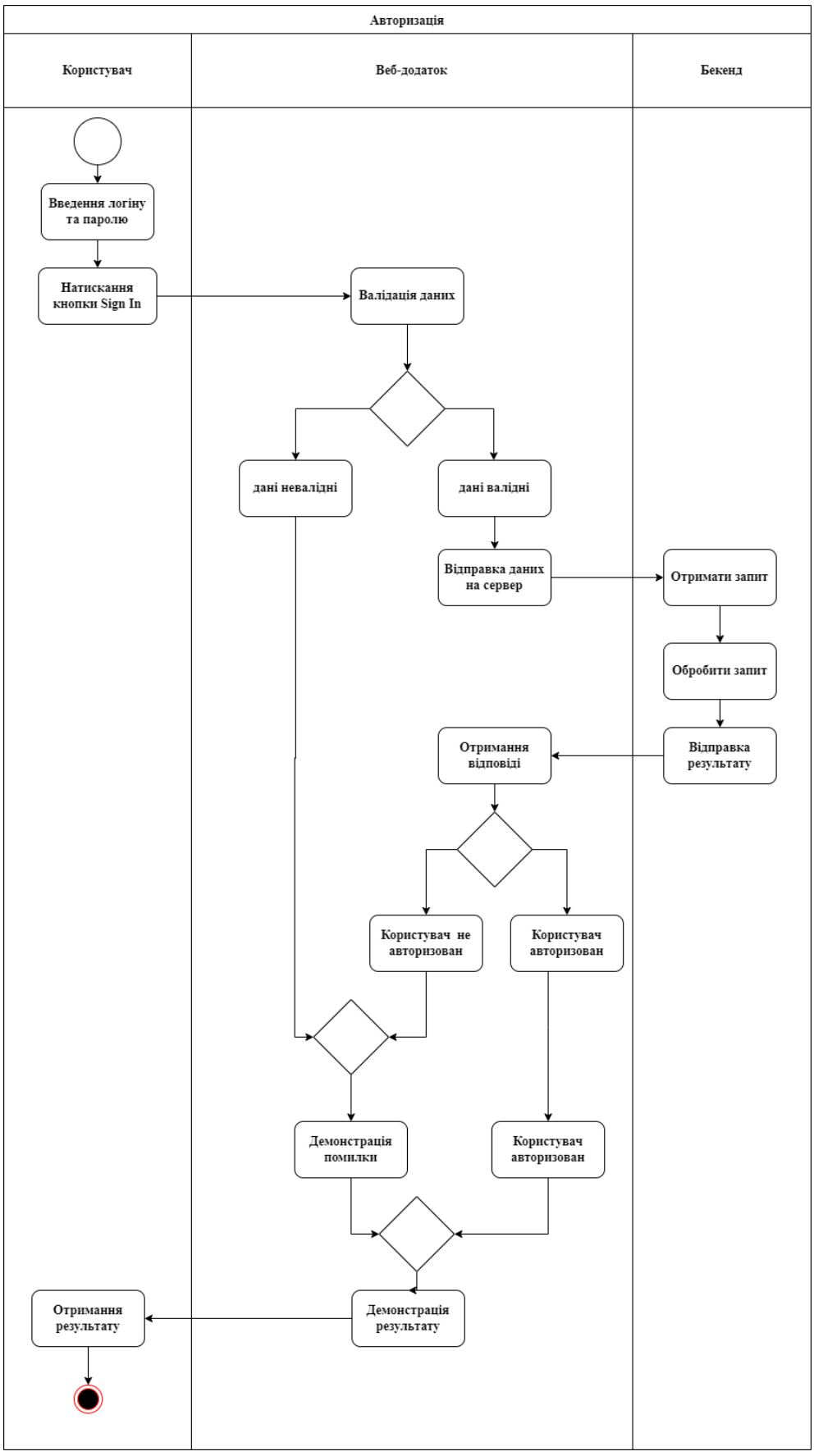


Рисунок 2.1.5 — Діаграма BPMN005

2.2 Архітектура програмного забезпечення

Для виконання завдання та побудови користувацького інтерфейсу у вигляді веб додатку були застосовані наступні технології, інструменти та мови програмування:

- мова програмування Javascript;
- мова програмування Typescript;
- бібліотека React;
- фреймворк Next.js;
- пакетний менеджер npm;
- Google Maps API;
- бібліотека Bootstrap 5;
- Github Actions.

2.2.1 Javascript

Мова програмування Javascript була розроблена у 1995-му році через бажання компанії Netscape отримати мову програмування в свій браузер. Не дивлячись на створення конкуруючої мови Jscript, яка була представлена компанією Microsoft в своєму браузері, обидві компанії(за ініціативою Netscpape) стандартизували мову та назвали її EcmaScript. На той час веб та інтернет представляли з себе XML та HTML документи з мінімальною стилізацією. Завдяки виникненню мови Javascript розробники веб-порталів отримали можливість додавати інтерактив на свої сторінки. Серед такого інтерактиву були падаючі сніжинки, біжучий рядок, та інші мінімальні прикраси. До початку створення бібліотек та повноцінних веб-додатків, побудованих цією мовою, Javascript оцінювався спільнотою розробників як іграшкова мова програмування. Варто зазначити, що стандарт EcmaScript мав на увазі не тільки власне стандартизацію, але і введення версіонування з доповненням стандарту новими фічами. Саме фіча однієї з таких версій призвела до тихої революції іграшкової мови, а саме концепція AJAX, яка дозволяла завантажувати дані з віддаленого вузла без перезавантаження

сторінки. Такої думки дотримуються Дуглас Крокфорд, постійний учасник розвитку мови Javascript, та Тимур Шемсединов, розробник і викладач КПІ ФІОТ. Завдяки поширенню цієї концепції великими проектами (Gmail, Yahoo Mail) почали з'являтися бібліотеки і фреймворки, які дозволяли спростити роботу розробника.

В цьому дипломному проєкті Javascript спочатку використовувався напряму без додаткових фреймворків. Це було обумовлено тим, що для оновлення інформації на сторінці достатньо було отримати відповідь з серверу та ввести її напряму в DOM, тому що сервер віддавав вже готові фрагменти HTML. Та під час розширення функціональних вимог було виявлено, що існуюча архітектура дуже громіздка для додавання нового функціоналу, і було вирішено почати використовувати додаткові бібліотеки і фреймворки, видаливши попередньо написаний код. В основному до ревізії мова використовувалась для виконання наступних задач:

– маніпуляції з об'єктною моделлю документу(DOM);
– додавання слухачів подій:
– формування HTML коду;
– взаємодія з зовнішніми системами(Google Maps API);
– отримання інформації про місцеположення користувача.

Після рефакторінгу мова все ще використовується для тих самих цілей, проте вже в оболонці бібліотек, про які буде зазначено далі.

2.2.2 Typescript

Typescript — це мультипарадигмальна статично типізована мова програмування, що розробляється і підтримується компанією Microsoft. Typescript стала логічним доповненням мови Javascript та була розроблена завдяки попиту розробників на типізовані мови програмування, проте не зважаючи на це велику кількість прихильників у цієї мови є і супротивники (серед впливових діячів також), які вважають цю мову надлишковою, а написання програмного коду нею — безглуздим. Тим не менш, в цьому

проєкті мова була використана для приборкання хаосу, адже можливість створення типів, інтерфейсів та більш розширені можливості для імплементації ООП дозволяють побачити слабкі місця архітектури програми.

Typescript не вміє напряму виконуватися в середовищі браузера, тому вона спочатку компілюється у нативний Javascript код, а потім, при необхідності, траспілюється такими інструментами як Babel або SWC. Слід зазначити що процес траспіляції(тобто переведення однієї версії EcmaScript в іншу) виконується для підтримки більш старих браузерів, в яких відсутня підтримка певних концепцій мови. Проте, в даному проєкті ми не працюємо безпосередньо з транспайлером, цим займається фреймворк.

В цьому проєкті Typescript стала в нагоді завдяки тому, що фреймворк, яким написаний додаток, об'єднує в собі і клієнтський код, і бекенд. Тому дуже зручно було декларувати інтерфейси та типи в одному місці і використовувати їх в обох індивідуальних частинах.

2.2.3 React

React — це бібліотека для побудови користувацьких інтерфейсів, розроблена компанією Meta (в минулому Facebook). Завдяки своїх декларативності та орієнтованості на компонентну архітектуру React затвердив себе як надійний інструмент для розробки фронтенд додатків.

Основними концепціями бібліотеки є:

- компоненти (функціональні та класові) і аргументи;
- JSX;
- стейт(стан) і життєвий цикл;
- композиція;
- Virtual DOM.

Компоненти представляють фундаментальні блоки, які є основою побудови додатків на React. Кожен компонент може представляти якийсь елемент на сторінці, бізнес-елемент або обгортку для інших компонентів. Компоненти до 16-ої версії були виключно представлені у вигляді класів

Javascript, проте згодом команда додала підтримку компонентів на базі звичайний функцій разом з хуками. Про хуки буде зазначено пізніше. Аргументи — це властивості компонентів, які передаються їм з батьківських компонентів. Аргументи фактично є аналогом аргументів функцій, де в залежності від аргументу може генеруватись різні види компоненту, наприклад, активна і неактивна кнопка або відкрите чи закрите діалогове вікно.

JSX — це спеціальне розширення мови Javascript, яке дозволяє писати HTML теги прямо в файлах з розширенням .js. Звичайно, React вміє створювати елементи і через вбудовану функцію React.createElement, проте застосування JSX спрощує цю процедуру а також спрощує умовний рендерінг. JSX також використовується з фреймворком Svelte.

Стан в React представляє з себе окреме поле в класових компонентах і спеціальний хук useState в функціональних компонентах. В бібліотеці React за правило взято передавання станів через аргументи вниз по дереву компонентів, тобто від батьківських до дочірніх компонентів, а передача подій навпаки, наверх по дереву. Таке правило призвело до набуття широкого поширення принципу підняття стану, або hoisting. Це процес, при якому стан дочірнього елемента підіймається до рівня батьківського і передається в дочірній через аргументи. Такий процес необхідний, коли два сусідніх дочірніх компоненти мають знати про стан один одного або мають спільний стан. Hoisting в свою чергу, через необхідність підняття стану на самий високий рівень, призвів до появи поняття контекстів. Контексти в React представляють з себе спеціальні компоненти які містять стан додатку і можуть надавати його будь-якому дочірньому компоненту будь-якої глибини за його бажанням. Цей механізм стає в нагоді коли є деякий стан, наприклад тема додатку (світла/темна), мова або інформація про авторизацію, який необхідний одночасно всім або більшості компонентів додатку. Без контекстів такий стан довелося б прокидати через всі компоненти через аргументи допоки

він не дійде до необхідного дочірнього компонента. Життєвий цикл компонента — це процес, який проходить компонент від створення до знищення. Розуміння життєвого циклу вкрай важливо для правильного налаштування поведінки компонента і правильної оптимізації. Неоптимізований життєвий цикл може призвести до витоку пам’яті додатку та непередбачуваним зависанням. Життєвий цикл класового компоненту описується спеціальними методами класу, що поділяються на групи:

– методи монтування;
– методи оновлення;
– метод демонтування;
– методи обробки помилок.

Більше детальний перелік цих методів і їх детальний опис можна знайти на сторінці офіційної документації бібліотеки React [1].

В функціональних компонентах теж є життєвий цикл, проте він контролюється спеціальними функціями — хуками, але, якщо конкретніше, то життєвий цикл в тут контролюється лише хуком useEffect, бо він одразу поєднує в собі декілька методів життєвого циклу, хоч і не покриває їх всі. Хуки були представлені разом з функціональними компонентами в версію react 16.8. Головною причиною їх появи було те, що неможливо було написати класовий компонент функцією, відвалювався великий пласт функціоналу. Хуки повертають цей функціонал та надають можливість керувати станом (useState, useRef, useContext), створювати спеціальні функції для роботи зі станом (useCallback, useMemo, useReducer) а також дозволяють створювати кастомні хуки об’єднуючи стандартні. Завдяки останній можливості існують безліч інструментів з відкритим доступом, що на повну використовують розробники в своїх проектах. Серед них популярні бібліотеки react-query (useQuery), react-hook-form (useForm), react-i18-next (useTranslation) та багато інших.

Композиція — це основний принцип створення нових компонентів в бібліотеці React. Щоб зрозуміти композицію слід спочатку визначити

альтернативний підхід — наслідування. Наслідування — це принцип ООП, за яким для створення схожого класу ми не копіюємо повністю поля і методи необхідного класу, а унаслідуємось від нього. Таким чином ми можемо мати один базовий клас (наприклад клас Автомобіль), а від нього створювати різні інші класи з іншими властивостями (наприклад Седан, Мікроавтобус, Пікап, Бензовоз). На відміну від наслідування композиція не плодить додаткові класи. Основна задача React в цьому процесі — створити компонент, тобто вже готовий об'єкт. Йому не важливо яку кількість класів ми для цього використали. Тому набагато вигідніше і швидше для розробника було б створити компонент через композицію, яка дозволяє кастомізувати елемент через аргументи (які виступають параметрами конструктора або аргументами функції). Таким чином попередній приклад перетворюється на один клас Автомобіль, який в конструктор приймає тип Автомобіля як параметр, і в залежності від нього рендерить необхідний тип автомобіля. Розробники react пояснюють цей підхід своїм досвідом використання бібліотеки всередині компанії та зазначають, що жодного разу не виникало ситуацій, при яких їм довелось би використовувати наслідування з компонентнами.

Virtual DOM — це концепт програмування, який преедбачає створення легковажного аналогу об'єктної моделі документі в середовищі Javascript. Цей концепт знаходиться у синхронізації зі справжнім DOMом завдяки процесу узгодження. Процес узгодження перебігає в нашому випадку в бібліотеці ReactDOM. Різні фреймворки та бібліотеки теж використовують схожу концепцію, наприклад Vue.js та Elm. Основна ідея використання цієї концепції в React в тому, що зміни стану моментально вносяться в Virtual DOM і накопичуються, допоки механізм всередині React не вирішить застосувати ці зміни до реального DOM. Цей механізм з версії 16 називається React Fiber. Таким чином важкі та часті зміни не навантажують машину користувача постійно і надають зміни поступово, чанками або патчами.

Поставлену задачу можна було вирішити і іншими інструментами також. Перш за все, на чистому Javascript або Typescript, проте вже експериментальним шляхом було визначено що це набагато важче, адже доводиться розписувати свої, більш прості аналоги механізмів фреймворків, такі як наприклад зберігання стану або оновлення DOM. Якщо говорити вже про бібліотеки і фреймворки, то в нас є деякий набір основних найпопулярніших технологій:

– React:
– Angular (або більш старий Angular.js):
– Vue.js;
– Svelte.

Серед всіх зазначених React був обраний через його простоту та підтримку спілкою, адже Angular хоч і підтримується та розширюється, проте є доволі складним для вивчення. Vue.js — навпаки, легкий в розумінні та використанні, проте має трішки інші концепції та меншу кількість розширень. Svelte на перший погляд покриває недоліки конкурентів. Він більш легкий через те, що не використовує концепцію Virtual DOM, він дуже легкий і зрозумілий в написанні, адже використовує суто нативний javascript код, проте він ще не набув необхідної популярності та не має достатньої кодової бази.

Більш того, React був обраний скоріше через наявність його як ядра у фреймворку Next.js, про це буде написано далі.

Як вже було зазначено, окрім самої бібліотеки React в проєкті було використано чимало додаткових бібліотек. Серед основних можемо назвати:

– react-bootstrap та bootstrap-icons;
– react-hook-form;
– sass та classnames.

React-bootstrap та bootstrap-icons використовуються разом з бібліотекою bootstrap для полегшення написання стилів, детальніше про неї буде в наступних пунктах.

React-hook-form використовується для полегшення створення форм в компонентах, завдяки хуку useForm та валідаторам типу yup або zod форми стають ще більше декларативними. Основна ідея цієї бібліотеки в тому, що вона сама обробляє будь-які зміни стану, якщо зареєструвати відповідний інпут, а потім надає весь стан форми в колбек-обробник.

Sass — це препроцесор, що інтерпретує та компілює свій код в код CSS. Sass зручний, тому що надає зручний синтаксис вкладеності класів та елементів, що робить код в компонентній архітектурі простішим для сприйняття, якщо дотримуватись відповідних правил. Classnames це бібліотека для зручного поєднання класів в JSX. Справа в тому, що в проєкти може виникнути необхідність перемикати класи елементу в залежності від стану компонента. Це легко зробити звичайним тернарним оператором коли таких класів два, проте якщо робити більше то код становиться важким для сприйняття через велику кількість вкладених тернарок. Тут на допомогу приходить classnames, яка надає функцію, що вміє комбінувати імена класів надаючи їх як, наприклад, об’єкт, в якому імена полів це назви класів, а значення — булеве значення, яке позначає включати клас чи ні.

Основні компоненти, що були створені в середовищі React, наведені в наступних таблицях.

Таблиця 2.2.1 — Компонент 1

Тип компонента	Сторінка
Назва	index.ts
Аргументи	
Детальний опис	Головна сторінка додатку, містить в собі необхідні компоненти для відображення (контейнер тегів, результати пошуку, шапка сайту) та хуки для збереження станів і поведінки сторінки.

Таблиця 2.2.2 — Компонент 2

Тип компонента	Сторінка
Назва	login.ts
Аргументи	csrfToken: string
Детальний опис	Сторінка для авторизації користувача, містить шапку вебсайту і форму для надання имейлу, паролю і підтвердження входу в обліковий запис та посилання на сторінку реєстрації. Проп csrfToken виступає прихованим значенням, що відправляється на сервер для коректної роботи сервісу авторизації.

Таблиця 2.2.3 — Компонент 3

Тип компонента	Сторінка
Назва	register.ts
Аргументи	
Детальний опис	Сторінка реєстрації користувача, містить в собі шапку сайту, форму для введення мейлу, паролю та підтвердження паролю, кнопку підтвердження реєстрації та посилання на сторінку авторизації.

Таблиця 2.2.4 — Компонент 4

Тип компонента	Сторінка
Назва	userManagement.ts

Продовження таблиці 2.2.4

Аргументи	
Детальний опис	Сторінка для оновлення даних користувача, містить шапку сайту та форму для оновлення пароля. Форма містить поле для старого паролю, для нового та для підтвердження нового паролю.

Таблиця 2.2.5 — Компонент 5

Тип компонента	Сторінка
Назва	newRecipe.ts
Аргументи	
Детальний опис	Сторінка для створення та збереження нового рецепту. Містить шапку сайту, форму для збереження даних нового рецепту. Після збереження автоматично відкриває сторінку збереженого рецепту.

Таблиця 2.2.6 — Компонент 6

Тип компонента	Сторінка
Назва	recipe/[id].ts

Продовження таблиці 2.2.6

Аргументи	
Детальний опис	Сторінка для окремого перегляду рецепту. Містить шапку сайту, поле для пошуку рецептів та секцію з детальною інформацією конкретного рецепту. Сторінка доступна лише при відкритті посилання на певний рецепт.

Таблиця 2.2.7 — Компонент 7

Тип компонента	Компонент
Назва	Header.ts
Аргументи	
Детальний опис	Шапка веб-сайту, містить в собі посилання на головну сторінку. Також в залежності від поточної сторінки та авторизованості користувача показує посилання на сторінку авторизації або посилання на сторінку керування обліковим записом.

Таблиця 2.2.8 — Компонент 8

Тип компонента	Компонент
Назва	Loading.ts

Продовження таблиці 2.2.8

Аргументи	
Детальний опис	Анімоване коло, що обертається навколо центру компонента, виступає в ролі індикатора завантаження даних в інших компонентах, наприклад під ас завантаження списку рецептів або списку кошиків до рецепту.

Таблиця 2.2.9 — Компонент 9

Тип компонента	Компонент
Назва	RecipePopup.ts
Аргументи	- recipeId: string; - showModal: boolean; - handleCloseModal: () => void; - ingredients: string[].

Продовження таблиці 2.2.9

Детальний опис	Модальне вікно, що демонструє деталі певного рецепту. Деталі рецепту отримуються з серверу всередині компонента. В деталі входять: - фото страви; - назва страви; - детальний опис; - детальний список інгредієнтів. Після отримання деталей отримуються і демонструються кошики для рецепта окремому компоненті. Компонент також містить кнопку, що відкриває інтерактивну мапу і кнопку закриття модального вікна.

Таблиця 2.2.10 — Компонент 10

Тип компонента	Компонент
Назва	ResultsContainer.ts
Аргументи	ingredients: string[]
Детальний опис	Контейнер, який за введеними інгредієнтами отримує з сервера список рецептів та демонструє їх у вигляді карточок ResipeSearchResultCard.

Таблиця 2.2.11 — Компонент 11

Тип компонента	Компонент
Назва	ResipeSearchResultCard.ts
Аргументи	- image: string; - ingrFast: string[]; - link: string; - name: string; - openRecipe: () => void.
Детальний опис	Карта з короткою інформацією про рецепт та посиланням на детальний опис рецепта.

Таблиця 2.2.12 — Компонент 12

Тип компонента	Компонент
Назва	ShoppingBasketsList.ts
Аргументи	- existingIngredients: string[]; - recipeIngredients: RecipeDetails["ingredients"].
Детальний опис	Компонент що відображає кошики та ціну кошика для кожного супермаркету. Для отримання кошиків з сервера обробляє вхідні дані для отримання списку інгредієнтів, яких немає в наявності у користувача.

Таблиця 2.2.13 — Компонент 13

Тип компонента	Компонент

Продовження таблиці 2.2.13

Назва	Tag.ts
Аргументи	- tagName: string - closeTag: ()=>void
Детальний опис	Компонент для відображення введеного користувачем інгредієнту. Показує назву інгредієнту та кнопку (хрестик) для видалення інгредієнту зі списку.

Таблиця 2.2.14 — Компонент 14

Тип компонента	Компонент
Назва	TagInput.ts
Аргументи	getRecipes: (tags: string[]) => void
Детальний опис	Представляє з себе поле для введення інгредієнтів. Містить в собі власне саме поле, список вже введених інгредієнтів та кнопку для запуску пошуку рецептів за інгредієнтами. Під час збереження списку в стан додатково зберігає список в LocalStorage для відновлення після перезавантаження сторінки.

Таблиця 2.2.15 — Компонент 15

Тип компонента	Компонент
Назва	MapSidebar.ts

Продовження таблиці 2.2.15

Аргументи	storeNames: string[]
Детальний опис	Сайдбар, що відображається при взаємодії користувача, містить повністю логіку взаємодії з інтерактивною мапою та інтеграцію з Google Maps API.

Таблиця 2.2.16 — Компонент 16

Тип компонента	Компонент
Назва	IngredientsInPopup.ts
Аргументи	existingIngredients: string[]; recipeIngredients: RecipeDetails["ingredients"];
Детальний опис	Список інгредієнтів в модальному вікні з деталями рецепту. Позначає червоним кольором інгредієнти, яких немає в списку користувача.

Таблиця 2.2.17 — Компонент 17

Тип компонента	Компонент
Назва	NewRecipeForm.ts
Аргументи	
Детальний опис	Форма для введення назви рецепту, списку інгредієнтів, детального опису, завантаження зображення та кнопку підтвердження збереження рецепту.

Таблиця 2.2.18 — Компонент 18

Тип компонента	Компонент
Назва	AuthFormErrorMessage.ts
Аргументи	errorType: string;
Детальний опис	Компонент, що відображає помилки авторизації. Через те що замість повідомлень сервер віддає короткі типи помилки потрібно мати маппінг або єнам для співставлення кодів та повідомлень. Найкращим рішенням було винести таку логіку в окремий компонент.

Основними допоміжними об’єктами були mockedApi та realAPI, вони мають спільний інтерфейс, що представлений на наступній таблиці:

Таблиця 2.2.19 — Інтерфейс методів API

Назва і тип метода	**Призначення**
getRecipes(recipeIngredients: **Array**[string],page: number): **Promise**<**Array**<**RecipeBaseDetails**>>	Отримання списку рецептів за введеними інгредієнтами
getRecipeDetails(recipeId: string): **Promise**<**RecipeFullDetails**>	отримання деталей інгредієнтів за ідентифікаційним номером рецепту
getRecipeCartPrices(whatToBuy: string[]): **Promise**<**RecipeCartPrices**>	Отримання цін кошиків для списку продуктів

Продовження таблиці 2.2.19

createNewUser(email: string, password: string): **Promise**<void>	Створення нового облікового запису в системі
updateUserPassword: (oldPassword: string, newPassword: string) => **Promise**<void>	Оновлення пароля облікового запису в системі
createNewRecipe: (recipeDetails: {image: string;description: string;}) => **Promise**<string>	Збереження даних нового рецепту в сховищі даних
getRecipeComments: (recipeId: string) => **Promise**<string[]>	Отримання коментарів до рецепту за ідентифікаційним номером
postNewComment: (recipeId: string,comment: string,user: string) => **Promise**<void>	Збереження нового коментаря до рецепту в сховищі даних

Об’єкти mockedApi та realApi відрізняються тим, що в mocked версії методи повертають фіктивні дані, а в real версії відбуваються справжні запити на сервер. Ця поведінка регулюється index.ts файлом, який в залежності від типу середовища (dev, prod, test) визначає яке API надавати в користування компонентам. Такий підхід можна назвати неповною імплементацією Dependency Injection, адже немає можливості перемикати API між різними компонентами. Проте в цьому немає потреби.

2.2.4 Next.js

Next.js — це фреймворк, побудований поверх Node.js і React для розширення функціоналу React рендерингом на стороні сервера (SSR) та генерацією статичних веб-сайтів. Альтернативою цьому фреймворку саме на

базі React є Gatsby, проте він більш громіздкий і важчий в обслуговуванні. З інших платформ серед аналогів можна перелічити Angular Universal або Scully для Angular та Nuxt.js для бібліотеки Vue.

Next.js був обраний як альтернатива попередній «голій» архітектурі. Першочергово задача була перевести архітектуру фронтенда на будь-яку іншу, але компонентну, тому підходили як React, так і Vue і Svelte. Проте був обраний саме React через наявний досвід роботи з ним. Не зважаючи на підтримку SSR та схожих технологій, основна задача Next.js для цього проєкту була об'єднати два окремих проєкти під одною спільною архітектурою та максимізувати перевикористання коду. Та з цим фреймворком надійшов також модуль next-auth, який дозволив мінімізувати написання коду для авторизації користувача.

2.2.5 Пакетний менеджер npm

Пакетний менеджер npm став основним джерелом отримання готового коду, бібліотек та фреймворків на платформі Node.js. Серед його альтернатив — yarn, який був швидшим за npm поки останній не оновився (тепер вони однакові по швидкості встановлення пакетів), і pnpm. Pnpm набув популярності за виправлення одного з ключових недоліків npm, а саме нагромодження одних і тих самих пакетів в різних проектах. Багато розробників з інших мов програмування не люблять Javascript та npm через це. Папка node_modules, яка використовується проєктами для зберігання залежностей може досягати декількох гігабайтів у великих корпоративних проєктах. Це призводить до сповільнення деплойменту софта а також, очевидно, займає місце на машинах розробників. Особливо це критично для машин з невеликим дисковим простором які використовуються для розробки декількох продуктів одночасно. Під час деплойменту проблема вирішується звичайним кешуванням залежностей. Pnpm вирішує цю проблему, встановлюючи всі пакети в теку node_modules не конкретного проєкту, а глобально, в кореневу теку користувача, В теку проєкту залежності додаються

через symlinki. Це дозволяє не встановлювати одні і ті ж залежності кілька разів на різних проектах.

Не зважаючи на вищезазначені переваги npm все ще є домінуючою платформою на ринку, тому що більшість утиліт (наприклад npx) за замовчуванням встановлюють пакети через npm а також через те, що при встановленні Node.js менеджер npm встановлюється разом з ним.

2.2.6 Google Maps API

API від Google Maps це зручне АПІ від компанії Google, яке дозволяє інтегрувати їх сервіс мап в будь-який додаток. Більш того, API дозволяє кастомізувати мапу, власноруч керувати мітками та відображати ту інформацію, яку розробник вважатиме необхідною. В цьому проєкті використано API для веб-додатків написане на Javascript. Більш того, враховуючи те, що основною бібліотекою фреймворку є React можна використати офіційну бібліотеку карт спеціально написану під React. Таким чином не доведеться власноруч писати адаптери та код зв’язки між React та скриптом мапи.

2.2.7 Bootstrap 5

Bootstrap — це UI бібліотека, що суттєво спрощує стилізацію додатків та створення прототипів. Ця бібліотека надає вже написаний набір класів та утиліт, які розробнику достатньо застосувати до тих чи інших елементів для отримання необхідного результату. Серед таких класів виділяють дві групи:

– утилітарні;

– компонентні.

Утилітарні класи надаються для застосування якогось форматування до елементу, наприклад вирівнювання за шириною по центру (self-align-center), створення рядку з колонками (row та col) або надання кольору елементу (btn-danger, alert-success).

Компонентні класи повністю надають стилізацію певному елементу, наприклад form-input, button або modal. Утилітарні класи можуть доповнювати

компонентні класи елементу для варіативності (кнопка може бути жовтою, червоною або вимкненою).

Головними відмінностями п’ятої версії від четвертої є відмова від jQuery та наявність пакету іконок, які також стали в нагоді в даному проєкті.

Основними альтернативами Bootstrap`а є Bulma, TailwindCSS та MaterialUI, проте їх набагато більше. Bootstrap був обраний лише через вже наявну кодову базу, яка використовувала цю бібліотеку, подібна задача могла бути вирішена будь-яким з вищезазначених інструментів.

2.2.8 Github Actions

Це безкоштовна система безперервної інтеграції розроблена компанією Github. Варто сказати, що безкоштовно вона надається тільки в рамках деякої кількості запусків на місяць, але невеликим проектам цього достатньо.

Actions дозволяє автоматизувати багато різних процесів. Завдяки простому синтаксису yaml можна декларативно описувати кроки виконання будь-якого процесу і запускати їх на наданих машинах. Серед таких кроків можуть бути:

– тестування додатку багатьма способами(e2e, load-testing, unit testing);

– репортінг результатів тестування;

– лінтинг;

– деплой додатку;

– маніпуляції з репозиторієм(додавання коментарів до пул реквестів, створення релізів, тощо).

Системи інтеграції дозволяють спростити і автоматизувати рутинні задачі а також здатні зберігати необхідну інформацію для звітування прогресу та ведення статистики розробки. Такі системи автоматизації процесів є дзеркалом роботи програмістів.

Серед аналогів Actions найпопулярнішими на сьогодні є TravisCI, Jenkins, JetBrains TeamCity, GitlabCI. Головною перевагою Github Actions

перед попередньо зазначеними є те, що ця система вже вбудована в існуючу платформу Github, тобто не треба мати і обслуговувати свій власний сервер для запуску необхідних процесів. Більш того, Actions є керованим спілкою розробників, тобто будь-який розробник може встановити вже готовий написаний крок автоматизації і використати його в своєму пайплайні завдяки маркетплейсу. Це дуже спрощує розробку однотипних додатків.

В даній роботі Github Actions був використаний для автоматизації деплойменту на платформу Vercel. Під час міграції на Next,js пайплайн був налаштований на деплоймент для кожного апдейту на кожній гілці. Після завершення міграції конфігураційний файл було оновлено таким чином, щоб деплоймент відбувався тільки при додаванні змін в гілку master.

Висновки до розділу

Розписавши основні вимоги до програмного забезпечення, визначившись з основними технологіями, проаналізувавши їх альтернативи та сформувавши принципи розробки та розгортання була розпочата розробка програмного продукту. Одночасно з розробкою ми почати писати тестові сценарії, про які буде написано в наступному розділі.

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз якості ПЗ

Тестування програмного забезпечення є невід'ємною частиною якісної розробки. Важливість цього кроку настільки велика, що існують навіть окремі процеси розробки, за яких софт пишеться тільки після написання тестів. Цей процес називається TDD або Test Driven Development. В даному дипломному проєкті цей процес не був використаний, адже він потребує багато часових ресурсів для коректної розробки вимог для їх подальшого перетворення на тести.

Основною задачею тестування є перевірка виконання всіх функціональних вимог за різних обставин. Це додає додаткові складності, адже програмне забезпечення може перестати працювати не завжди за причинами, очевидними для розробника. Серед джерел таких причин перш за все може бути сервіс, який обслуговує програмний код та надає його користувачам за запитом.

В нашому випадку помилки можуть виникнути на трьох етапах роботи програмного забезпечення:

– під час розробки;
– під час збірки проєкту(компіляції);
– під час роботи на сервері.

Помилки під час розробки автоматично відображаються розробнику локальним сервером. Більш того мова Typescript завдяки типізації дозволяє виявляти та знешкоджувати помилки ще до того як вони з'являться. Компілятор Typescript попереджає про помилки і під час збірки теж, що потенційно зменшує вірогідність появи помилки в рантаймі (під час роботи програми на сервері).Таким чином залишається лише два джерела появи помилок: сервер, що обслуговує код та бекенд. Завдяки тому, що сервер нам надається PaaS платформою ми можемо не турбуватися про стабільність його

роботи, адже стабільна робота входить в умови користування сервісом Vercel. Помилки з бекенду обробляються самим кодом таким чином, що або користувач про них не знає взагалі, або отримує відповідні повідомлення без переривання нормального користувацького досвіду.

3.2 Опис процесів тестування

Не дивлячись на зазначені в попередньому пункті аргументи тестування все одно необхідно проводити, адже тестування програмного коду це не стільки перевірка якості продукту, скільки перевірка якості виконання своєї роботи розробником. Правильно написані тести можуть також не тільки допомогти розробнику зосередитися на нагальних проблемах коду але й можуть виступати в якості звіту для замовника.

3.3 Опис контрольного прикладу

Для тестування програмного забезпечення були використані, тестові сценарії, описані в таблицях 3.3.1-3.3.7.

Таблиця 3.3.1 — Опис контрольного прикладу 1

Мета тесту	Перевірити створення облікового запису і авторизації
Початковий стан	
Вхідні дані	Вхідні дані створюються під час реєстрації.
Схема проведення тесту	1) Відкрити сторінку авторизації; 2) Ввести логін та пароль від існуючого акаунту; 3) Натиснути кнопку "Log in".

Продовження таблиці 3.3.1

Очікуваний результат	Повинно відбутись перезавантаження сторінки та перенаправлення користувача на головну сторінку з індикацією авторизованості.
Кінцевий результат	Відображається головна сторінка з іконкою користувача.

Таблиця 3.3.2 — Опис контрольного прикладу 2

Мета тесту	Перевірити можливість введення рецептів
Початковий стан	
Вхідні дані	Вхідні дані користувач придумує сам.
Схема проведення тесту	1) Відкрити головну сторінку; 2) В поле вводу ввести слово "мясо" та натиснути Enter; 3) В поле вводу ввести слово "мука" та натиснути Enter; 4) Натиснути хрестик поряд зі словом "мясо"; 5) Перезавантажити сторінку.
Очікуваний результат	Після перезавантаження список має містити тільки інгредієнт під назвою "мука".
Кінцевий результат	Інгредієнт "мука" відображається один в списку для пошуку.

Таблиця 3.3.3 — Опис контрольного прикладу 3

Мета тесту	Перевірити можливість переглядання списку рецептів
Початковий стан	Введений список інгредієнтів
Вхідні дані	Вхідні дані вже введені користувачем.
Схема проведення тесту	1) Маючи валідний введений список інгредієнтів натиснути кнопку пошуку рецептів. 2) Дочекатись завантаження сторінки.
Очікуваний результат	Повинен відобразитись список карточок рецептів під полем пошуку. Кожна карточка має містити зображення рецепту, назву та список інгредієнтів.
Кінцевий результат	Відображається декілька карток рецептів з необхідною інформацією.

Таблиця 3.3.4 — Опис контрольного прикладу 4

Мета тесту	Перевірити можливість переглядання деталей рецепту
Початковий стан	Відкритий список рецептів
Вхідні дані	Вхідні дані вже введені користувачем та оброблені додатком.

Продовження таблиці 3.3.4

Схема проведення тесту	1) Обрати рецепт зі списку запропонованих; 2) Натиснути на посилання обраного рецепту; 3) Дочекатися завантаження всієї інформації про рецепт.
Очікуваний результат	Після завантаження повинні відображатися: - кнопка для закриття модального вікна; - фото рецепту; - детальний опис; - список інгредієнтів з позначенням відсутніх інгредієнтів; - список кошиків і цін; - кнопка для перегляду мапи.
Кінцевий результат	Відображаються всі вищезазначені елементи.

Таблиця 3.3.5 — Опис контрольного прикладу 5

Мета тесту	Перевірити можливість перегляду мапи
Початковий стан	Відкрита сторінка деталей рецепту
Вхідні дані	Вхідні дані вже введені користувачем та оброблені додатком.
Схема проведення тесту	1) Натиснути на кнопку перегляду мапи. 2) Дочекатись завантаження мапи.

Продовження таблиці 3.3.5

Очікуваний результат	Після взаємодії повинна бути продемонстрована мапа з мітками найближчих супермаркетів. Користувач повинен мати змогу взаємодіяти з картою.
Кінцевий результат	Відкривається мапа з мітками, користувач може переміщувати або змінювати масштаб мапи.

Таблиця 3.3.6 — Опис контрольного прикладу 6

Мета тесту	Перевірити можливість додавання коментарів до рецепту
Початковий стан	Користувач авторизований
Вхідні дані	Вхідні дані користувач придумує сам.
Схема проведення тесту	1) Ввести інгредієнти в поле пошуку та зробити пошук рецептів. 2) Обрати рецепт з результатів пошуку та натиснути на його посилання. 3) Дочекатися завантаження деталей рецепту. 4) Прогортати до секції коментарів. 5) Ввести в поле вводу коментаря будь-який текст та підтвердити ввід. 6) Дочекатися завантаження.
Очікуваний результат	Введений коментар повинен зберегтися в блок з коментарями та відобразитися під деталями рецепта.
Кінцевий результат	Коментар відображається під деталями рецепта.

Таблиця 3.3.7 — Опис контрольного прикладу 7

Мета тесту	Перевірити можливість створення та збереження нового рецепту
Початковий стан	Користувач авторизований
Вхідні дані	Вхідні дані користувач придумує сам.
Схема проведення тесту	1) Перейти на сторінку створення рецепту. 2) Ввести назву рецепту, детальний опис, список інгредієнтів в відповідні поля, прикріпити зображення. 3) Підтвердити заповнену форму. 4) Дочекатися завантаження.
Очікуваний результат	Щойно введені деталі рецепту повинні зберегтися в системі як новий рецепт та відобразитись користувачеві.
Кінцевий результат	Рецепт відображається користувачеві.

Висновки до розділу

За результатами тестування можна зробити висновок, що програмне забезпечення розроблене якісно на надійно, адже всі основні тестові сценарії були успішними. Звичайно, слід не забувати такі речі, як публічне тестування та технічний борг. Публічне тестування може виявити деякі баги в користувачів на специфічних приладах, які не були виявлені командою розробки або командою тестування, проте це питання вже розглядається під час впровадження та підтримки програмного забезпечення. Технічний борг — це річ неминуча в сучасних веб-застосунках, адже технології та інструменти розробки постійно оновлюються, створюються нові. Angular.js, який був популярним ще всього лише 8 років тому вже витіснений такими рішеннями як React та Vue. Тому найкраще, що можна зробити для його мінімізації — це

писати коректну підтримувану та здатної до масштабізації архітектуру додатку.

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Розгортання програмного забезпечення було виконано на PaaS платформі Vercel. Серед альтернатив цієї платформи можна виділити:

– Heroku;

– Google App Engine;

– деякі з сервісів AWS.

Фактично Vercel був обраний через простоту взаємодії, розгортання а також через те що ця компанія є автором фреймворку Next.js, на якому був написаний цей проєкт. Це дозволяє інтегрувати та розгорнути додаток з мінімальними втратами по часу та максимальною сумісністю між платформою і додатком, адже Vercel надає готовий шаблон розгортання додатку, для якого немає необхідності нічого змінювати в конфігурації додатку.

4.2 Робота з програмним забезпеченням

Для роботи з програмним забезпеченням, для його оновлення та обслуговування необхідно мати:

– комп’ютер зі встановленою операційною системою Linux або Windows;

– будь-який редактор файлів або коду (наприклад nano, vim, Emacs, VSCode, IntelijIDEA та інші);

– встановлені на комп’ютері Node.js та npm.

Подальші інструкції для розробників описані в файлі Readme.md в репозиторії проєкту.

ВИСНОВКИ

В цьому проєкті був розроблений додаток, необхідний для користувачів платформи, що дозволяє шукати кулінарні рецепти за різними інгредієнтами та перегляду детальної інформації про рецепти та інгредієнти. Було вивчено можливі варіанти розробки програмного коду, фреймворки, бібліотеки та архітектури розгортання. В якості основної платформи було використано платформу Node.js та фреймворк Next.js. Компоненти фронтенду були написані використовуючи бібліотеку React. Для написання стилів використовувався препроцесор Sass. Визначивши основний інструментарій було сформовано вимоги для майбутнього продукту та розроблено програмний код. Розроблений код було розгорнуто на платформі Vercel.

Відповідний код та продукт був протестований за основними функціональними вимогами та пройшов процес тестування з позитивним результатом.

Цей дипломний проєкт став не тільки зручним застосунком для студентів, але і чудовим прикладом розробки та удосконалення сучасних веб-застосунків. Завдяки легкості його бізнес-логіки основні конфігураційні файли проєкту можна використовувати в майбутніх проєктах як основу або шаблон для швидкого початку роботи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) React.Component [Електронний ресурс]. — 2022 — Режим доступу до ресурсу: https://uk.reactjs.org/docs/react-component.html#the-component-lifecycle

2) Simpson K. You don’t know Javascript [Електронний ресурс] / Kyle Simpson. — 2021 — Режим доступу до ресурсу: https://github.com/getify/You-Dont-Know-JS

3) Duckett J. Javascript and JQuery [Електронний ресурс] / John Duckett. — 2020 — Режим доступу до ресурсу: https://archive.org/details/javascriptjqueryjonduckett/mode/2up

4) Next.js. [Електронний ресурс] – 2022 –: https://nextjs.org/docs/

5) Github Actions. [Електронний ресурс] – 2022 –: https://docs.github.com/en/actions

6) Sass. [Електронний ресурс] – 2022 –: https://sass-lang.com/documentation/

7) Google Cloud Documentation. [Електронний ресурс] – 2022 –: https://cloud.google.com/docs

8) Goldberg J. Learning Typescript / Josh Goldberg, 2022

9) Crockford D. The World's Most Misunderstood Programming Language Has Become the World's Most Popular Programming Language [Електронний ресурс] / Douglas Crockford — 2008 — Режим доступу до ресурсу: http://javascript.crockford.com/popular.html

10) Стефанов С. React.js, Швидкий старт / Стоян Стефанов, 2016

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2022 р.

Сервіс оптимального вибору рецепту в умовах змінюваності заданих параметрів (Комплексна тема). Клієнтська частина веб-застосунку та розгортання.

Опис програми

КПІ.ІП-8234.045490.09.13

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олесь КОВТУНЕЦЬ

Нормоконтроль:
_____ Катерина ЛІЩУК

Виконавець:
_____ Бунямін ШАБАНОВ

Київ – 2022

ТЕКСТ ПРОГРАМНОГО КОДУ

Тексти програмного коду

Сервіс оптимального вибору рецепту в умовах змінюваності заданих параметрів (Комплексна тема). Клієнтська частина веб-застосунку та розгортання.

(Найменування програми (документа))

CD-R

(Вид носія даних)

15 арк, 118784 Кб

(Обсяг програми, арк., Кб)

Київ – 2022

```
import { NextPage } from "next";
import Head from "next/head";
import Script from "next/script";
import Image from "next/image";
import styles from "../styles/Home.module.scss";
import TagInput from "../components/TagInput/TagInput";
import Header from "../components/Header/Header";
import { useEffect, useState } from "react";
import ResultsContainer from "../components/ResultsContainer/ResultsContainer";
import RecipePopup from "../components/RecipePopup";
import Loading from "../components/Loading/Loading";

import API from "../components/API";
const Home: NextPage = () => {
  // null: for search not started yet.
  // [data]: for something found
  // []: for nothing found
  // endless search will be somewhere else
  const [recipes, setRecipes] = useState<any[] | null>(null);
  const [tags, setTags] = useState<string[]>([]);
  const [loading, setLoading] = useState(false);
  const [currentRecipePopupId, setCurrentRecipePopupId] = useState<
    null | string
  >(null);

  const fetchRecipes = async (tagsToSearch: string[], pages = 1) => {
    setLoading(true);
    setTags(tagsToSearch);
    if (tagsToSearch && tagsToSearch.length) {
      try {
        const newRecipes = await API.getRecipes(tagsToSearch, pages);
        if (pages > 1) {
          setRecipes([...(recipes ?? []), newRecipes]);
          return;
        }
```

```
      console.log(newRecipes);
      setRecipes(newRecipes);
    } catch (error) {
      console.debug(error);
    }
  }
  setLoading(false);
};
const dropRecipeId = () => {
  setCurrentRecipePopupId(null);
};

return (
  <div className={styles.container}>
    <Head>
      <title>Fill the Fridge</title>
      <meta name="description" content="Generated by create next app" />
      <link rel="icon" href="/favicon.ico" />
    </Head>

    <Header />
    <TagInput getRecipes={fetchRecipes} />
    <div className="container mt-4 container-infinite d-flex justify-content-center">
      {loading ? (
        <Loading />
      ) : (
        <ResultsContainer
          recipes={recipes}
          openRecipe={(id) => setCurrentRecipePopupId(id)}
          // updateRecipes={fetchRecipes}
        />
      )}
    </div>
    <div className="d-flex justify-content-center mt-4" id="reloading"></div>
    <div className="container mt-4 container-infinite">
```

```
        <div
          className="row row-cols-1 row-cols-sm-2 row-cols-md-3"
          id="search-results-container"
        ></div>
        <div className="d-flex justify-content-center" id="loading"></div>
      </div>
      {/* <div className="container container-popup"> */}
      <RecipePopup
        recipeId={currentRecipePopupId}
        ingredients={tags}
        dropRecipeId={dropRecipeId}
      />
      {/* </div> */}
    </div>
  );
};

export default Home;
import { NextPage } from "next";
import Link from "next/link";
import Header from "../components/Header/Header";
import { getCsrfToken } from "next-auth/react";
import { CtxOrReq } from "next-auth/client/_utils";
import { useRouter } from "next/router";
import AuthErrorContainer from "../components/AuthErrorContainer/AuthErrorContainer";
type Props = {
  csrfToken: string;
};

const Login: NextPage<Props> = ({ csrfToken }) => {
  const { error } = useRouter().query;
  return (
    <>
      <Header />
      <div className="container h-100 mt-4">
```

```
      <form method="post" action="/api/auth/callback/credentials">
        <AuthErrorContainer error={error} />
        <input name="csrfToken" type="hidden" defaultValue={csrfToken} />

        <div className="form-group">
          <label htmlFor="exampleInputEmail1">Email address</label>
          <input
            type="email"
            className="form-control"
            name="email"
            id="exampleInputEmail1"
            aria-describedby="emailHelp"
            autoComplete="email"
          />
        </div>
        <div className="form-group">
          <label htmlFor="exampleInputPassword1">Password</label>
          <input
            type="password"
            name="password"
            className="form-control"
            id="exampleInputPassword1"
            autoComplete="current-password"
          />
        </div>
        <button type="submit" className="btn btn-primary">
          Log In
        </button>
        <Link href="/register">Don&apos;t have an account?</Link>
      </form>
    </div>
  </>
 );
};
```

```
export async function getServerSideProps(context: CtxOrReq | undefined) {
  const csrfToken = await getCsrfToken(context);
  return {
    props: { csrfToken },
  };
}
export default Login;
import { NextPage } from "next";
import Head from "next/head";
import React, { useState } from "react";
import { useForm } from "react-hook-form";
import API from "../components/API";
import Header from "../components/Header/Header";
import IngredientsInput from "../components/IngredientsInput/IngredientsInput";
import { newRecipeForm, newRecipeIngredient } from "../types";

const NewRecipe: NextPage = () => {
  const {
    register,
    handleSubmit,
    watch,
    formState: { errors },
  } = useForm<newRecipeForm>();
  const [ingredients, setIngredients] = useState<newRecipeIngredient[]>([]);
  const onSubmit = (data: newRecipeForm) => {
    const formToSend: Record<string, any> = {
      ...data,
      ingredients,
      image: data.image[0],
    };
    const formData = new FormData();
    for (const key in formToSend) {
      if (Object.prototype.hasOwnProperty.call(formToSend, key)) {
        const element = formToSend[key];
        if (Array.isArray(element)) {
```

```
        formData.append(
          key,
          `[${element.map((curr) => JSON.stringify(curr)).join(",")}]`
        );
      } else {
        formData.append(key, element);
      }
    }
  }
  API.createNewRecipe(formData).then((resultId) => {
    console.log(resultId);
  });

  console.log(formToSend);
  return null;
};
const setNewIngredient = (newIngredient: newRecipeIngredient) => {
  setIngredients([...ingredients, newIngredient]);
};
const removeIngredient = (ingrIndex: number) => {
  const ingredientsWithoutDeleted = [
    ...ingredients.slice(0, ingrIndex),
    ...ingredients.slice(ingrIndex + 1),
  ];

  setIngredients(ingredientsWithoutDeleted);
};
return (
  <>
    <Head>
      <title>Fill the Fridge: New Recipe</title>
      <meta
        name="description"
        content="Create a new recipe in Fill the Fridge"
      />
```

```
</Head>

<Header />
<div className="container h-100 mt-4">
  <form onSubmit={handleSubmit(onSubmit)}>
    <div className="form-group">
      <label htmlFor="inputName">Recipe Name</label>

      <input
        type="text"
        className="form-control"
        id="inputName"
        aria-describedby="recipeNameHelp"
        {...register("name")}
      />
    </div>
    <div className="form-group">
      <label htmlFor="inputDescription">Description</label>

      <textarea
        className="form-control"
        id="inputDescription"
        aria-describedby="recipeHelp"
        {...register("recipe")}
      />
    </div>
    <div className="form-group">
      <label htmlFor="inputImage">Recipe Image</label>

      <input
        type={"file"}
        className="form-control"
        id="inputImage"
        aria-describedby="recipeImageHelp"
        {...register("image")}
```

```
          />
        </div>
        <IngredientsInput
          ingredients={ingredients}
          setNewIngredient={setNewIngredient}
          removeIngredient={removeIngredient}
        />
        <button type="submit" className="btn btn-primary">
          Save Recipe
        </button>
      </form>
    </div>
  </>
 );
};

export default NewRecipe;
import { GetServerSideProps, NextPage } from "next";
import Head from "next/head";
import Link from "next/link";
import Header from "../../components/Header/Header";
import TagInput from "../../components/TagInput/TagInput";
import API from "../../components/API";
import { RecipeFullDetails } from "../../types";
import { useEffect, useState } from "react";
import MapSideBar from "../../components/RecipePopup/MapSidebar/MapSideBar";
import WholeRecipeContent from "../../components/WholeRecipeContent/WholeRecipeContent";
import NavigateBeforeIcon from "@mui/icons-material/NavigateBefore";
import styles from "./styles.module.scss";
import classnames from "classnames";

const Recipe: NextPage<RecipeFullDetails> = (props) => {
  const [mapIsShowing, setMapIsShowing] = useState(false);
  const [baskets, setBaskets] = useState([]);
  const [tags, setTags] = useState<string[]>([]);
```

```
  useEffect(() => {
    const localTags = localStorage.getItem("tags");
    const parsedTags = localTags ? JSON.parse(localTags) : [];
    setTags(parsedTags);
  }, []);

  return (
    <>
      <Head>
        {/* <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <!-- og tags -->
    <meta property="og:title" content="<%= `Fill the Fridge: ${name}` %>" />
    <meta property="og:url" content=<%=link %> />
    <meta property="og:image" content=<%=image %> />
    <meta property="og:type" content="article" />
    <meta property="og:description" content="<%= `${reciep.substr(0, 127)}...` %>" />
    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"
      integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh"
crossorigin="anonymous" />
    <link rel="stylesheet" href="/assets/css/input.css" />
    <link rel="stylesheet" href="/assets/css/fullreciep.css" />
    <!-- Bootstrap JavaScript and JQuery-->
    <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js"
      integrity="sha384-J6qa4849blE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1yYfoRSJoZ+n"
      crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
      integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"
      crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js"
      integrity="sha384-wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl3Og8ifwB6"
      crossorigin="anonymous"></script>
    <script
```

```
src="https://maps.googleapis.com/maps/api/js?key=AIzaSyCYsL3NtRxHucdRBUKgnmP5m0QQcTnjM3s&librarie
s=places"></script>


    <script src="/assets/js/main.js" type="module"></script>
    <title>
      <%= `Fill the Fridge: ${name}` %>
    </title> */}
      </Head>
      <Header />
      <TagInput
        getRecipes={function (tags: string[]): void {
          throw new Error("Function not implemented.");
        }}
      />


      <div className="container mt-4 container-infinite mb-4">
        <div className={classnames({ "d-flex": mapIsShowing })}>
          <div
            className={styles.fullrecipe}
            id="reciepContent"
            style={{ width: mapIsShowing ? "3.5rem" : undefined }}
          >
            <div
              className={classnames(styles.backbutton, {
                "d-none": !mapIsShowing,
              })}
              style={{ marginTop: "0.5rem" }}
              id="dropmarkers"
            >
              <button
                type="button"
                className="rounded-circle btn btn-light"
                style={{ padding: 0, width: "2rem", height: "2rem" }}
                onClick={() => setMapIsShowing(false)}
              >
```

```
            <NavigateBeforeIcon />
          </button>
        </div>
        <WholeRecipeContent
          ingredients={tags}
          mapIsShowing={mapIsShowing}
          openMap={() => setMapIsShowing(true)}
          baskets={baskets}
          setBaskets={setBaskets}
          recipeData={props}
        />
      </div>
      {mapIsShowing && <MapSideBar mapIsShowing stores={baskets || []} />}
    </div>
  </div>
 </>
 );
};

export const getServerSideProps: GetServerSideProps<RecipeFullDetails> = async (
 context
) => {
 if (!context.params) {
  return { props: {} };
 }
 const { id } = context.params;

 const res = await API.getRecipeDetails(id as string);
 console.log(res);
 return {
  props: {
   ...res,
  },
 };
};
```

```
export default Recipe;
import { NextPage } from "next";
import error from "next/error";
import Link from "next/link";
import Header from "../components/Header/Header";
import { useForm } from "react-hook-form";
import API from "../components/API";
import { signIn } from "next-auth/react";

interface RegisterForm {
  email: string;
  password: string;
  passwordSubmit: string;
}
const Register: NextPage = () => {
  const {
    register,
    handleSubmit,
    watch,
    formState: { errors },
  } = useForm<RegisterForm>();
  const onSubmit = async (data: RegisterForm) => {
    console.log(data);
    return API.createNewUser(data)
      .catch((e) => {
        alert(e);
      })
      .then(() => {
        return signIn(
          "credentials",
          {},
          {
            email: data.email,
            password: data.password,
```

```
        }
      );
    });
  };
  return (
    <>
      <Header />
      <div className="container h-100 mt-4">
        <form onSubmit={handleSubmit(onSubmit)}>
          <div className="form-group">
            <label htmlFor="inputEmail">Email address</label>
            <input
              type="email"
              className="form-control"
              id="inputEmail"
              aria-describedby="emailHelp"
              autoComplete="email"
              {...register("email")}
            />
          </div>
          <div className="form-group">
            <label htmlFor="inputPassword">Password</label>
            <input
              type="password"
              className="form-control"
              id="inputPassword"
              autoComplete="new-password"
              {...register("password")}
            />
          </div>
          <div className="form-group">
            <label htmlFor="inputPasswordSubmit">Submit Password</label>
            <input
              type="password"
              className="form-control"
```

```
            id="inputPasswordSubmit"
            autoComplete="new-password"
            {...register("passwordSubmit")}
           />
          </div>
          <button type="submit" className="btn btn-primary">
           Sign Up
          </button>
          <Link href="/login">Already have an account?</Link>
        </form>
      </div>
    </>
  );
};
export default Register;
import "bootstrap/dist/css/bootstrap.css";
import "../styles/globals.scss";
import { SessionProvider } from "next-auth/react";

import type { AppProps } from "next/app";

function MyApp({ Component, pageProps: { session, ...pageProps } }: AppProps) {
  return (
    <SessionProvider session={session}>
      <Component {...pageProps} />
    </SessionProvider>
  );
}

export default MyApp;
```

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“___” _____ 2022 р.

Сервіс оптимального вибору рецепту в умовах змінюваності заданих параметрів (Комплексна тема). Клієнтська частина веб-застосунку та розгортання

Графічний матеріал

КПІ.ІП-8234.045490.10.99

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Олесь КОВТУНЕЦЬ

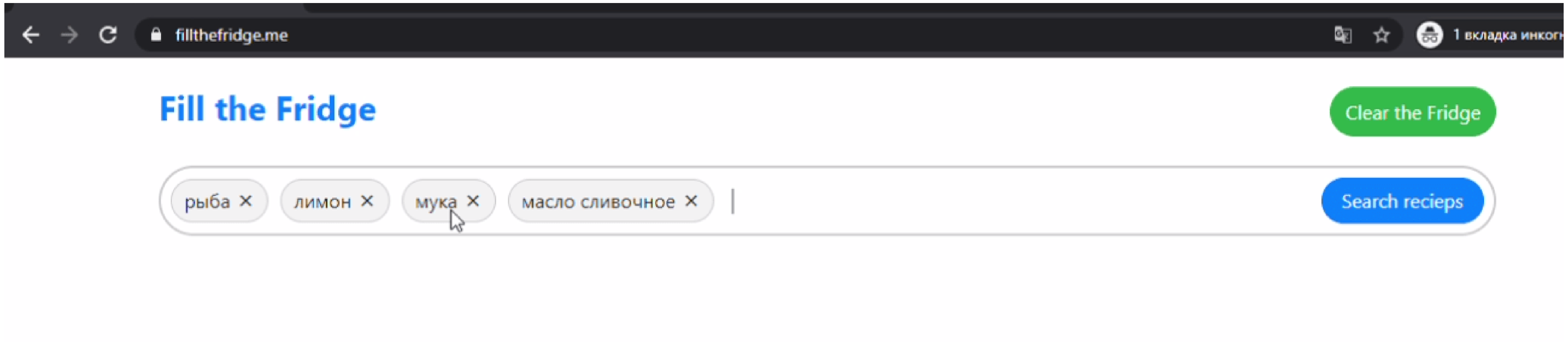
Нормоконтроль:

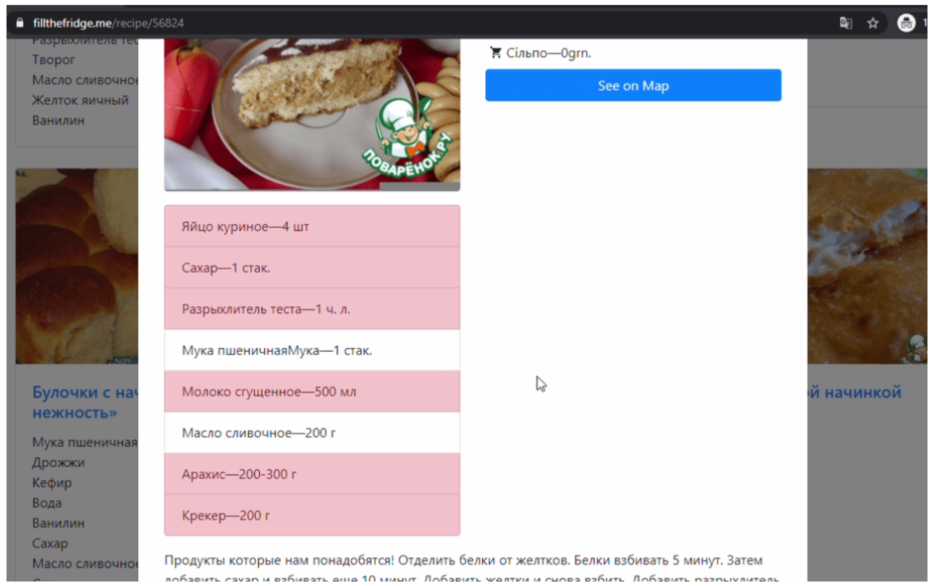
_____ Катерина ЛІЩУК

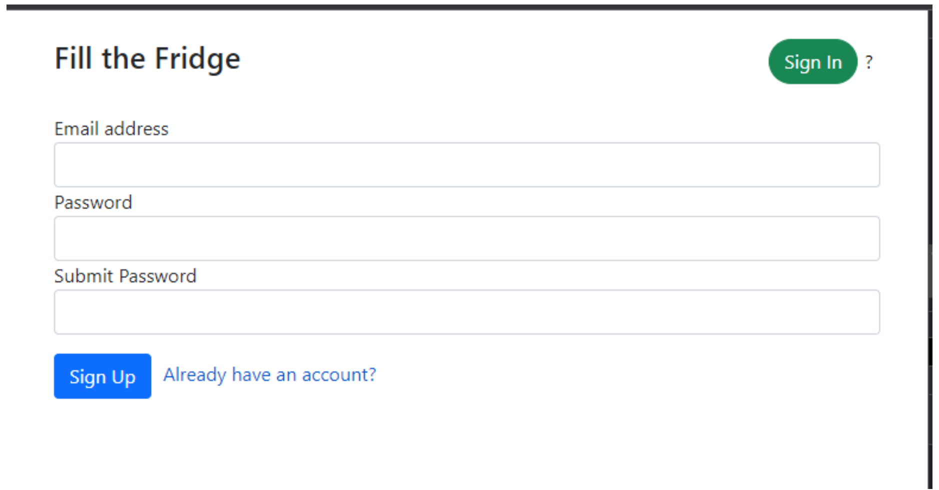
Виконавець:

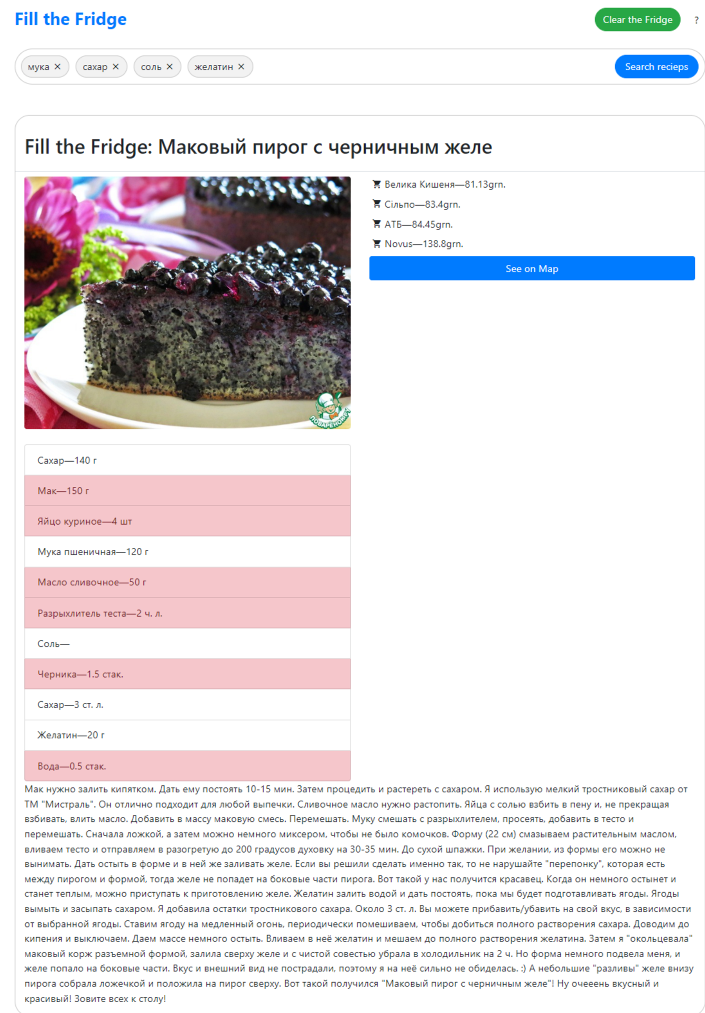
_____ Бунямін ШАБАНОВ

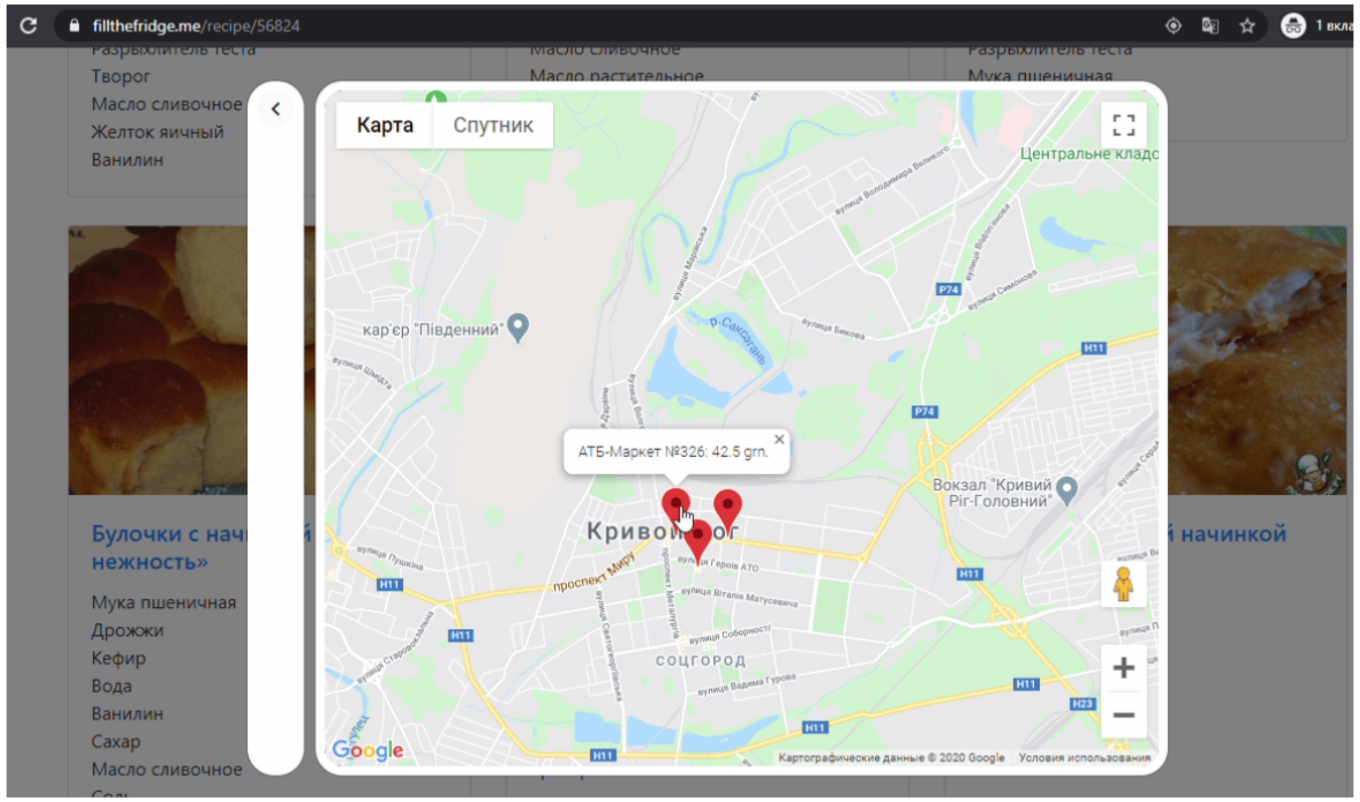
Київ – 2022











Зм.	Арк.	№ документа	Підпис	Дата
Розробив		Шабанов Б. Ш. о		
Перевірив		Ковтунець О. В.		
Т. кон.				
Н. кон.		Ліщук К.І.		
Затвердив		Ковтунець О. В.		

КПІ.ІП-8234.045490.10.59.КЕ

Креслення вигляду екранних форм індивідуальної частини №1

Літера	Маса	Масштаб

Аркуш	Аркушів

Сервіс оптимального вибору рецепту в умовах змінюваності заданих параметрів (Комплексна тема). Клієнтська частина веб-застосунку та розгортання.

КПІ ім.Ігоря Сікорського
Кафедра ІПІ
гр. ІП-82

UNICHECK

Имя пользователя:
Лісовиченко Олег Іванович

ID проверки:
1011600320

Дата проверки:
17.06.2022 07:12:16 EEST

Тип проверки:
Doc vs Internet + Library

Дата отчета:
17.06.2022 07:12:43 EEST

ID пользователя:
76913

Название файла: **ІП-82_Шабанов_ПЗ**

Количество страниц: **50** Количество слов: **7302** Количество символов: **55324** Размер файла: **403.87 KB** ID файла: **1011468906**

9.33%
Совпадения

Наибольшее совпадение: **7.46%** с источником из Библиотеки (ID файла: **1011468905**)

1.36% Источники из Интернета	1	Страница 52
9.33% Источники из Библиотеки	116	Страница 52

0% Цитат

Исключение цитат выключено

Исключение списка библиографических ссылок выключено

0% Исключений

Нет исключенных источников

Модификации

Обнаружены модификации текста. Подробная информация доступна в онлайн-отчете.

Замененные символы	10