ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка Web-застосунку для викладення, пошуку та оцінки рецептів з використанням Node.js та React»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Максим ТИМОШЕНКО
(підпис)

Виконав: здобувач(ка) вищої освіти групи ПД-42

Максим ТИМОШЕНКО

Керівник: Світлана ШЕВЧЕНКО
к.п.н., доцент

Рецензент: _____

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Тимошенку Максиму Петровичу

1. Тема кваліфікаційної роботи: «Розробка Web-застосунку для викладення, пошуку та оцінки рецептів з використанням Node.js та React»
керівник кваліфікаційної роботи к.п.н., доцент Світлана ШЕВЧЕНКО,
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» лютого 2024 р. № 36.

2. Строк подання кваліфікаційної роботи «28» травня 2024 р.

3. Вихідні дані до кваліфікаційної роботи: теоретичні відомості та програмні реалізації для викладення, пошуку та оцінки рецептів; технічна документація з описом Node.js та React.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Аналіз технічного завдання
2. Вибір засобів програмної реалізації
3. Програмна реалізація web-застосунку для викладення, пошуку та оцінки рецептів
4. Тестування

5. Перелік графічного матеріалу: *презентація*

1. Аналіз аналогів.
2. Вимоги до програмного забезпечення.
3. Програмні засоби реалізації.
4. Діаграма варіантів використання.
5. Схема бази даних.
6. Мапа web-застосунку.
7. Екранні форми.
8. Апробація результатів дослідження

6. Дата видачі завдання «28» лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	28.02-06.03.2024	
2	Аналіз та дослідження існуючих аналогів	07.03-13.03.2024	
3	Створення серверної частини застосунку	14.03-26.03.2024	
4	Створення клієнтської частини застосунку	27.03-10.04.2024	
5	Інтеграція серверної та клієнтської частини	11.04-17.04.2024	
6	Тестування	18.04-28.04.2024	
7	Оформлення роботи: вступ, висновки, реферат	29.04-05.05.2024	
8	Розробка демонстраційних матеріалів	06.05-12.05.2024	
9	Попередній захист роботи	13.05-31.05.2024	

Здобувач(ка) вищої освіти _____ Максим ТИМОШЕНКО
(підпис)

Керівник
кваліфікаційної роботи _____ Світлана ШЕВЧЕНКО
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 50 стор., 2 табл., 39 рис., 14 джерел.

Мета роботи – спрощення процесу викладення, пошуку та оцінки рецептів за допомогою технологій Node.js та React.

Об'єкт дослідження – процес викладення, пошуку та оцінки рецептів.

Предмет дослідження – розробка web-застосунку для викладення, пошуку та оцінки рецептів з використанням технологій Node.js та React.

Короткий зміст роботи: Дане дослідження присвячене розробці web-застосунку для викладення, пошуку та оцінки рецептів. У роботі проведено аналіз аналогів серед існуючих застосунків: Cookpad, Kitchen Stories, FoodYub, визначено їх переваги та недоліки, які стали основою для формулювання вимог до нового застосунку та його проєктування. Розроблено алгоритм роботи застосунку та програмно реалізовані ключові функціональні можливості, зокрема: завантаження рецептів, їх пошук та оцінку; можливість отримати випадковий рецепт та пропозиція популярних рецептів на основі кількості лайків. У роботі використано бібліотеку Express для створення серверної частини, Bcrypt для забезпечення безпеки паролів, Mongoose для взаємодії з базою даних MongoDB, MUI для розробки користувацького інтерфейсу в середовищі React, Axios для виконання HTTP-запитів.

Користуватися застосунком може будь-яка людина, яка цікавиться кулінарією. За допомогою цього застосунку користувачі зможуть ефективно обмінюватися рецептами, що сприятиме розширенню їхнього кулінарного досвіду та розвитку кулінарних навичок. Також даний Web-застосунок можливо застосувати у навчальному процесі студентів-кулінарів для інтерактивних занять.

КЛЮЧОВІ СЛОВА: WEB-ЗАСТОСУНОК, ВИКЛАДЕННЯ РЕЦЕПТІВ, ПОШУК РЕЦЕПТІВ, ОЦІНКА РЕЦЕПТІВ, NODE.JS, REACT

ЗМІСТ

ВСТУП

Обґрунтування вибору теми та її актуальність. Розробка web-застосунку для обміну рецептами виправдовується не лише потребами шанувальників кулінарії, але й актуальністю самої теми у сучасному цифровому світі. Споживачі все більше віддають перевагу онлайн-ресурсам для отримання кулінарних інструкцій та інгредієнтів. Створення такого застосунку відповідає прагненню до зручності та доступності інформації.

У цьому контексті, застосунок для обміну рецептами стає невід'ємною частиною кулінарного життя. Зростаючий інтерес до готування вдома, пошук нових кулінарних ідей та дотримання певних дієт створює попит на зручні інструменти для обміну рецептами. Такий застосунок може значно спростити цей процес, забезпечуючи швидкий доступ до різноманітних рецептів та можливість обміну досвідом між користувачами.

Нарешті, враховуючи розвиток сучасних технологій, web-застосунок є важливим інструментом для зближення спільноти кулінарів. Він створює можливість для взаємодії та обміну ідеями, що сприяє зростанню кулінарної культури та розширенню горизонтів у готуванні. Такий застосунок стає відповіддю на сучасні потреби спільноти та допомагає зробити кулінарний досвід більш доступним і захопливим для всіх.

Ступінь вивчення проблеми. Наразі вже існує багато сайтів та додатків, які надають користувачам можливість ділитися рецептами. Але в багато з них мають проблему - вони не пропонують комплексного підходу до обміну рецептами, обмежуючись лише базовим функціоналом. Більшість таких платформ не надають користувачам можливості взаємодії з іншими користувачами, коментування та обговорення рецептів, що обмежує можливості спільного навчання та вдосконалення в галузі кулінарії або, навіть, мати функції підписок, що можуть обмежувати функціонал.

Все вище викладене підтвердило актуальність даного дослідження і його важливість.

Метою роботи є спрощення процесу викладення, пошуку та оцінки рецептів за допомогою технологій Node.js та React.

Для досягнення цієї мети в роботі необхідно вирішити такі *завдання*:

1. Здійснити огляд літературних джерел, існуючих засобів розв'язання завдань предметної галузі.
2. Здійснити порівняльний аналіз аналогів програмного забезпечення (Cookpad, Kitchen Stories, FoodYub); визначити функціональні та нефункціональні вимоги на основі отриманих результатів порівняння аналогів.
3. Обґрунтувати вибір технологій для розробки web-застосунку викладення, пошуку та оцінки рецептів.
4. Спроєктувати архітектуру системи та інтерфейс застосунку з викладення, пошуку та оцінки рецептів.
5. Розробити web-застосунок з викладення, пошуку та оцінки рецептів за допомогою технологій Node.js та React.
6. Провести тестування розробленого застосунку.

Виходячи з цього, *об'єктом* дослідження є процес викладення, пошуку та оцінки рецептів, *предметом* дослідження – web-застосунок для викладення, пошуку та оцінки рецептів з використанням технологій Node.js та React.

Методи дослідження. Для вирішення вищезгаданих завдань у роботі використано наступні методи: системно-структурні методи, порівняльний аналіз, методи передачі, зберігання та обробки інформації. Для розробки застосунку використано бібліотеку Express для створення серверної частини, Bcrypt для забезпечення безпеки паролів, Mongoose для взаємодії з базою даних MongoDB, MUI для розробки користувацького інтерфейсу в середовищі React, Axios для виконання HTTP-запитів

Наукова новизна полягає в створенні web-застосунку, який дозволить користувачам ділитися рецептами та взаємодіяти один з одним, при цьому який не буде мати реклами та підписок.

Практична значущість результатів полягає у вирішенні практичної задачі – створення програмного продукту, що надає можливість обміну рецептами, мовою програмування JavaScript з використанням Node.js та React, що дозволяє забезпечити швидкий та ефективний обмін рецептами між користувачами. Розроблений web-застосунок може використовуватися для персональних цілей в кулінарії.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Актуальність проєкту

Сучасні споживачі все більше надають перевагу онлайн-ресурсам для отримання кулінарних інструкцій та придбання інгредієнтів. Ця тенденція підсилюється зростаючою популярністю кулінарних блогів, відеоуроків та додатків, що пропонують швидкий та зручний доступ до рецептів.

Розробка web-застосунку для обміну рецептами відповідає прагненню сучасних користувачів до зручності та швидкого доступу до інформації. Web-застосунок дозволяє користувачам шукати та ділитися рецептами у будь-який час і з будь-якого місця, що особливо важливо у швидкому ритмі життя.

Зі зростанням доступу до Інтернету та нових технологій, web-застосунки стають все більш функціональними та орієнтованими на користувачів. Вони пропонують можливості для інтеграції з іншими платформами та сервісами, такими як соціальні мережі, що підвищує їх привабливість і функціональність.

Кулінарні хобі та приготування їжі вдома стали популярнішими, особливо в результаті пандемії COVID-19. Люди шукають нові рецепти та кулінарні ідеї, щоб урізноманітнити своє меню, що робить web-застосунок для обміну рецептами актуальним та затребуваним.

Web-застосунки для обміну рецептами сприяють покращенню кулінарних навичок користувачів, дозволяючи їм експериментувати з новими стравами та методами приготування. Крім того, такі платформи підтримують активну спільноту, де користувачі можуть обмінюватися порадами та відгуками.

Web-застосунки для обміну рецептами забезпечують розширений доступ до кулінарної інформації, зокрема для тих, хто має обмежений доступ до традиційних джерел. Це включає людей з віддалених регіонів, з обмеженими можливостями пересування або тих, хто прагне навчатися самостійно.

Таким чином, розробка web-застосунку для обміну рецептами є актуальною, оскільки вона задовольняє зростаючі потреби споживачів у отриманні кулінарної інформації.

1.2 Аналіз та загальна характеристика web-застосунку для викладення, пошуку та оцінки рецептів

Web-застосунок (web-програма) - це прикладна програма, яка зберігається на віддаленому сервері і доставляється через Інтернет за допомогою інтерфейсу браузера. Web-сервіси за визначенням є web-додатками, і багато, хоча й не всі, web-сайти містять web-додатки.

Web-застосунок можна розгорнути на сервері, забезпечивши доступ до нього через браузер або інші клієнтські програми. Користувачі можуть використовувати функції web-сервісу, такі як реєстрація, вхід до системи, пошук, надсилання даних і отримання відповідей.

Розробники створюють web-програми для найрізноманітніших цілей і користувачів, від організацій до приватних осіб, з багатьох причин. Найпоширенішими web-програмами можуть бути web-пошта, онлайн-калькулятори або магазини електронної комерції. Хоча користувачі можуть отримати доступ до деяких web-додатків лише за допомогою певного браузера, більшість з них доступні незалежно від браузера.

Функціональність web-додатків для викладення, пошуку та оцінки рецептів може включати:

1. Реєстрацію користувачів: можливість створення облікових записів користувачів.
2. Перегляд рецептів: можливість переглядати рецепти інших користувачів.
3. Створення рецептів: можливість створювати та наповнювати рецепт змістом.
4. Взаємодія: можливість користувачів оцінювати та коментувати рецепти.

1.3 Аналіз вже існуючих додатків для викладення, оцінки та коментування рецептів

Cookpad [1] - це глобальна платформа для обміну рецептами, де користувачі можуть шукати, зберігати та ділитися рецептами домашнього приготування. На рисунку 1.3.1 представлена головна сторінка цієї платформи.

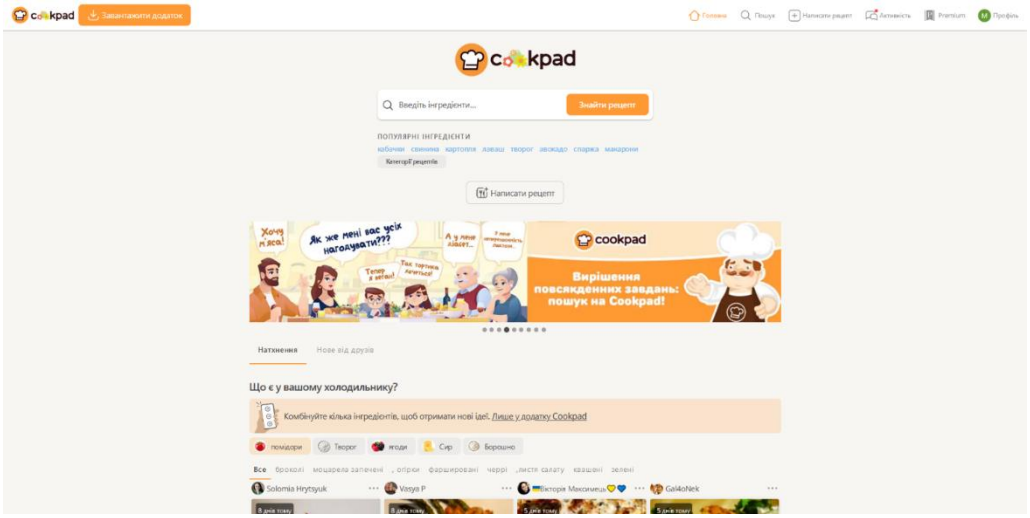


Рисунок 1.3.1 – головна сторінка Cookpad

Однією з головних переваг Cookpad є можливість створення рецептів власноруч, що дозволяє користувачам додавати свої унікальні страви до платформи. Крім того, Cookpad доступний на різних платформах, включаючи Web, Android та iOS, що забезпечує користувачам зручний доступ до рецептів з будь-якого пристрою.

Cookpad також підтримує систему оцінювання рецептів за допомогою емодзі, що додає елемент веселощів і дозволяє користувачам легко виражати свої враження від рецептів. Платформа підтримує коментування рецептів, що сприяє активному обговоренню та обміну порадами між користувачами. Пошук за назвою полегшує знаходження конкретних рецептів.

Однак, Cookpad має кілька недоліків. Наприклад, створення рецепту вимагає переходу на інші сторінки, що може бути незручним для користувачів. Також на платформі відсутня функція випадкового рецепту, яка могла б запропонувати

користувачам нові ідеї для приготування страв. Крім того, Cookpad не пропонує можливість відображення популярні рецепти на основі кількості лайків, що може ускладнювати пошук найпопулярніших страв на платформі.

FoodYub [2] – це web-застосунок, призначений для любителів їжі та кулінарії. Цей сервіс дозволяє користувачам знаходити рецепти, а також ділитися своїми власними рецептами з іншими. На рисунку 1.3.2 представлена головна сторінка цієї платформи.

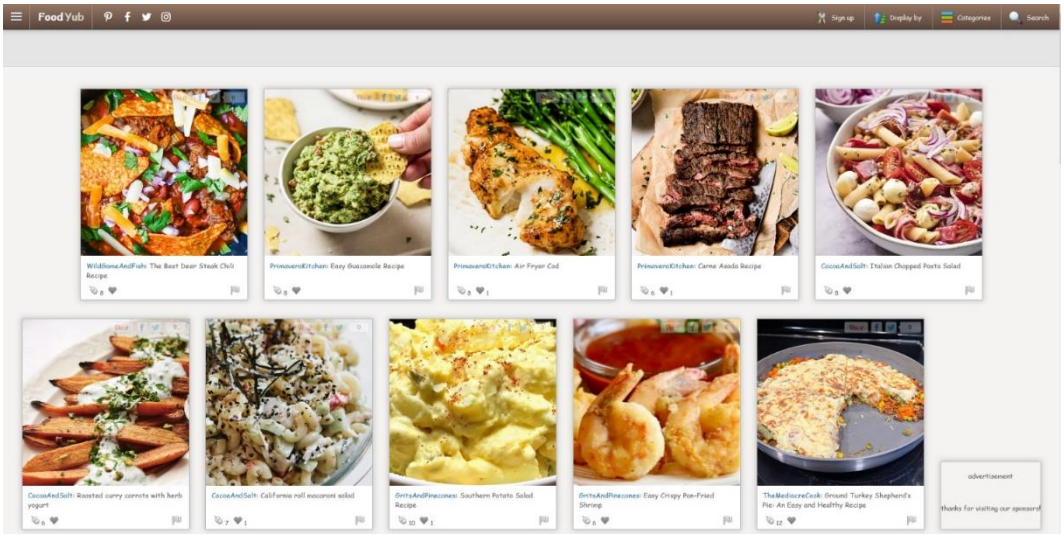


Рисунок 1.3.2 – головна сторінка FoodYub

Серед його переваг можна відзначити можливість створення власних рецептів, що стимулює творчість і сприяє розширенню кулінарних навичок користувачів. Також варто підкреслити наявність пошуку за назвою, що робить процес пошуку конкретних рецептів більш зручним та ефективним. Функціонал оцінки рецептів за допомогою лайків дає змогу користувачам виражати свої вподобання.

Однак, серед недоліків FoodYub слід відзначити відсутність спрощеного доступу до створення рецептів без переходу на інші сторінки, що може призвести до зменшення зручності та швидкості процесу створення. Також слід відзначити відсутність можливості коментування рецептів, що обмежує взаємодію користувачів та унеможливлює обмін корисними порадами та враженнями щодо страв.

Kitchen Stories [3] - це платформа для любителів готувати та пекти, на якій представлено понад 11 000 рецептів від професійних шеф-кухарів та редакторів харчування. На ньому можна знайти прості, вегетаріанські, веганські, бранч та популярні страви, а також замовити інгредієнти онлайн за допомогою Getir та Gorillas. На рисунку 1.3.3 представлена головна сторінка цієї платформи.

Рисунок 1.3.3 – головна сторінка Kitchen Stories

Проте, не всі функції Kitchen Stories є ідеальними. Наприклад, хоча є можливість коментувати рецепти, відсутня можливість підписки на інших користувачів, що може обмежити спільноту та обмін досвідом. Також, хоча пошук за назвою присутній, відсутня можливість знаходити випадкові рецепти або переглядати найкращі рецепти на основі кількості лайків безпосередньо на платформі, що може обмежити можливості користувачів зі спробою відкрити для себе нові страви. Також, хоча є можливість створення рецептів власноруч, процес створення може бути складнішим через неспрощений доступ до цієї функції, що може змусити користувачів шукати альтернативні шляхи.

Таблиця 1.3.1

Зведені результати аналізу характеристик додатків для викладення, оцінки та коментування рецептів

Назва	Kitchen Stories	FoodYub	Cookpad
Платформа	Web, Android, iOs	Web	Web, Android, iOs
Можливість створення рецепту власноруч	+	+	+
Спрощений доступ до створення рецепту (без переходу на інші сторінки)	-	-	-
Випадковий рецепт	-	-	-
Формат оцінки рецептів	Лайки	Лайки	Емодзі
Можливість підписки на інших користувачів	-	+	+
Можливість коментування рецептів	+	-	+
Можливість пошуку рецепту за назвою	+	+	+
Рейтинг найкращих рецептів на основі кількості лайків	-	-	-

2 ВИБІР ЗАСОБІВ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Постійне оновлення і розвиток технологій призводить до появи нових мов програмування та фреймворків, що робить вибір правильної технології для проєкту важким завданням. Від правильного вибору мови програмування або фреймворку залежить багато аспектів проєкту, включаючи його продуктивність, масштабованість, безпеку та швидкість розробки. Для реалізації даного проєкту було розглянуто декілька технологій, а саме Python, PHP, Java, Node.js, React та Go.

2.1 Python

Python є досить легкою мовою через її синтаксис, який є чітким і лаконічним [4]. Саме через це Python є популярним вибором початківців, але ця мова все ще залишається дуже потужною, щоб підтримувати деякі з найпопулярніший продуктів і додатків таких відомих компаній, як NASA, Google, Cisco, Microsoft тощо.

Крім того, Python славиться своєю сферою web-розробки. Він має багато фреймворків, серед яких bottle.py, Flask, CherryPy, Pyramid, Django та web2py. Ці фреймворки були використані для створення таких відомих сайтів, як Spotify, Reddit, Washington Post.

Але, не дивлячись на це, Python має багато недоліків. Деякими з них є:

- повільна обробка запитів через єдиний потік коду;
- він синхронний, тому працює повільно;
- коли система Python зростає, її також стає важко підтримувати і вона стає невиправдано складною.

2.2 PHP

PHP є серверною мовою програмування з відкритим кодом, яку використовують для створення web-сайтів та додатків. Хоч і маючи більш ніж 26-

річну історію, популярність PHP поступово зменшується (рис. 2.2.1). Не зважаючи на це, потужна онлайн-спільнота та вичерпна документація допомагають розробникам ефективно використовувати можливості PHP.

Але не можна не виділити такі недоліки:

1. Як і Python, PHP є синхронною мовою.
2. Використання більшої кількості функцій фреймворку PHP та інструментів призводить до низької продуктивності онлайн-додатків [5].
3. Він дуже складний в управлінні, тому що не має грамотної модульності.

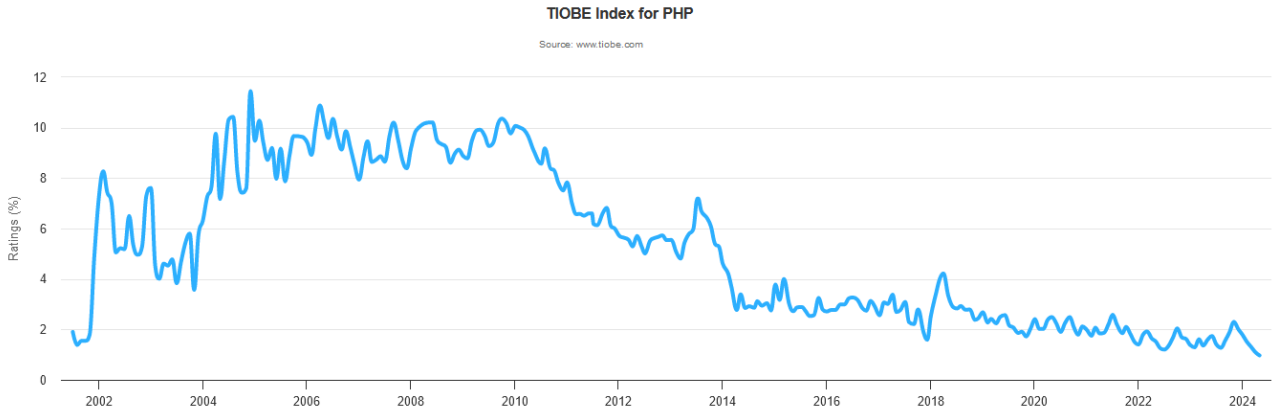


Рисунок 2.2.1 - Рейтинг популярності мови PHP

2.3 Java

Java була створена компанією Oracle у 1995 році і була спробою запустити нову мову, яка б дозволила побутовій електроніці ефективно взаємодіяти між собою [6]. Java мала чітку здатність забезпечувати інтерактивність і підтримку мультимедіа, що зробило її особливо придатною для іншої нової, на той час, технології – Інтернету. Це дозволило Java стати однією з провідних мов програмування для web-розробки і використовуватися в широкому спектрі web-додатків та сервісів. Але є недоліки, які не можна не назвати:

1. Хоча Java не складна у вивченні, вона призводить до створення складного коду, який є особливо багатослівним.
2. Хоча Java є мовою з високою портативністю та масштабованістю, вона може мати меншу швидкодію порівняно з іншими мовами

3. Java відома своєю великою витратою пам'яті, особливо для масштабних додатків (рис.2.3.1).

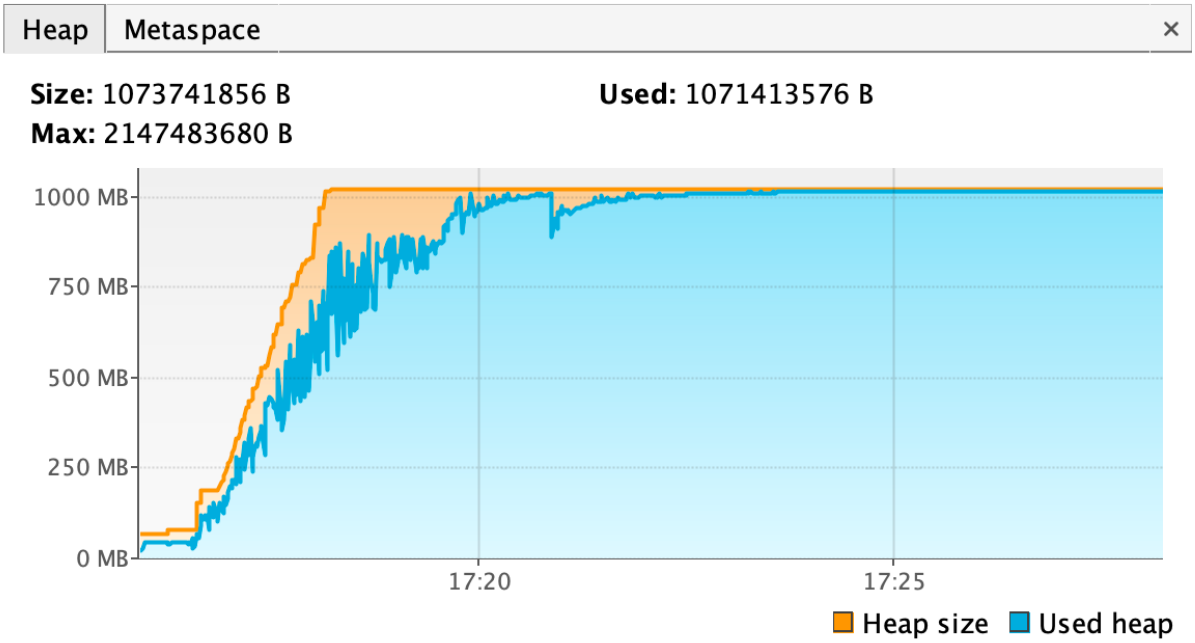


Рисунок 2.3.1 – Витік пам'яті в Java

2.4 Go

У сфері web-розробки мова програмування Go, яку часто називають Golang, стала важливим гравцем [7]. Ця компільована мова програмування, відома своєю простотою та високою продуктивністю, пропонує унікальний підхід до створення надійних web-додатків. Однією з головних переваг мови програмування Go для web-розробки є її висока продуктивність. Розроблена зі статичною системою типів, Go забезпечує поєднання швидкості та надійності, яке важко порівняти з іншими мовами. Ефективний механізм збору сміття та можливість компіляції в один двійковий файл ще більше підвищують продуктивність мови. Але, не дивлячись на це, Go має свої недоліки:

1. У порівнянні з іншими популярними мовами, Golang має відносно невелику стандартну бібліотеку (рис. 2.4.1).
2. Golang є простою мовою, але вона має деякі унікальні особливості, на вивчення та розуміння яких розробникам може знадобитися певний час.

3. Хоча Golang існує вже деякий час, її екосистема все ще відносно незріла порівняно з іншими мовами.

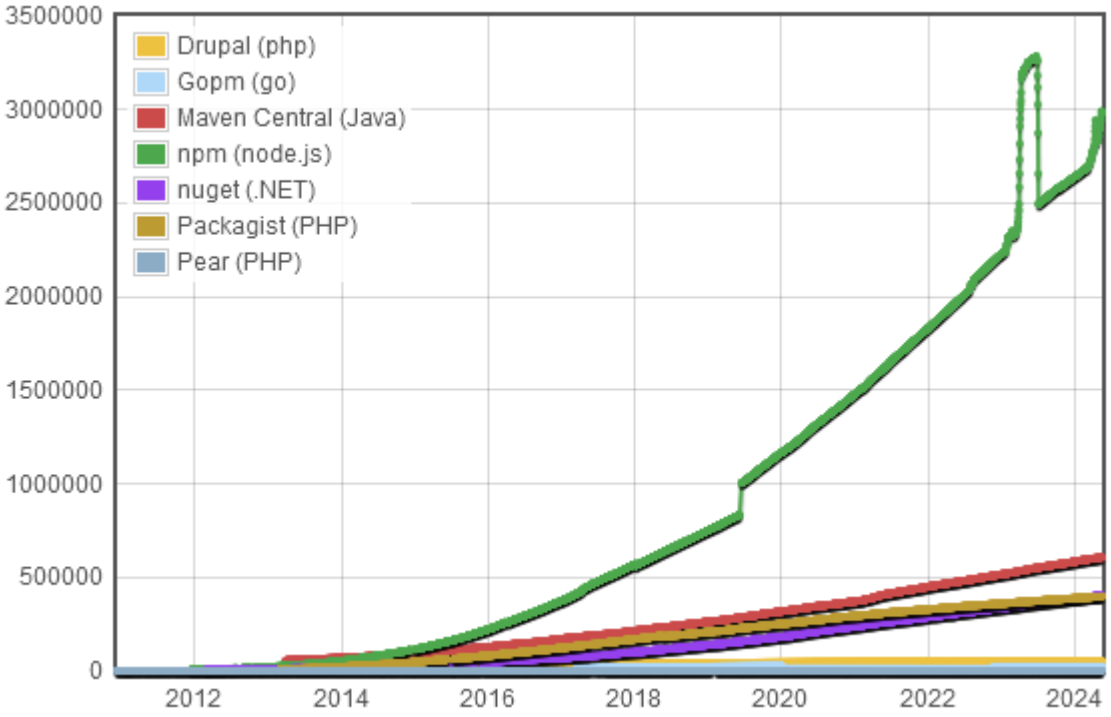


Рисунок 2.4.1 – Графік кількості доступних бібліотек за технологіями

2.5 JavaScript

Після ретельного аналізу та порівняння різних технологій, визначили, що JavaScript є оптимальним вибором. Причинами, які вплинули на це рішення є [8]:

1. Популярність. За даними Stackoverflow.com, JavaScript є найпопулярнішою мовою програмування, що використовується професійними розробниками сьогодні. Результатом цієї популярності є її спільнота, що постійно вдосконалює платформу та її екосистему, створюючи нові бібліотеки, інструменти та фреймворки. Також це означає, що в Інтернеті доступно багато ресурсів, таких як навчальні посібники, документація та форуми.

2. Велика кількість фреймворків та бібліотек. JavaScript відомий своєю безліччю фреймворків та бібліотек, що дозволяють робити на JavaScript майже все, що можна тільки уявити. Наприклад, такі фреймворки, як React та Angular, забезпечують швидкий та ефективний процес створення інтерактивних інтерфейсів, Node.js дозволяє розробляти серверну частину додатків, а TypeScript,

як і у традиційних об'єктно-орієнтованих мовах, має підтримку повноцінних класів та статичну типізацію.

3. Швидкість розробки та впровадження змін. JavaScript володіє простим синтаксисом та великою кількістю готових рішень у вигляді бібліотек та фреймворків. Це дозволяє розробникам швидко прототипувати та розробляти функціонал, а також легко впроваджувати зміни в процесі розробки. Багато інструментів, таких як live reload і hot module replacement, також допомагають прискорити цей процес, що дозволяє розробникам більш ефективно працювати над своїми проєктами.

4. Підтримка великих корпорацій та стабільність. JavaScript є вибором багатьох великих технологічних компаній, таких як Google, Facebook, Microsoft, Netflix і багато інших. Це означає, що мова має стабільну підтримку, активний розвиток та надійність, оскільки вона постійно використовується для створення великих та складних додатків з найвищими вимогами до продуктивності та надійності.

Спочатку JavaScript був відомий як LiveScript і був створений в компанії Netscape Communications Бренданом Айхом у 1995 році як мова сценаріїв для використання з браузером компанії – Netscape Navigator [9]. Проте Netscape змінили назву на JavaScript, щоб позиціонувати її як супутник мови Java, продукту свого партнера, компанії Sun Microsystems. Однак, окрім деякої поверхневої синтаксичної схожості, JavaScript жодним чином не пов'язаний з мовою програмування Java.

З його появою все більше браузерів додавали підтримку JavaScript, але протягом більшої частини своєї історії він не розглядався як серйозна мова програмування. Проте при потребі до запуску додатку в браузері, не дивлячись на проблеми з продуктивністю і безпекою в перших версіях, розробникам доводилося його використовувати, бо вони не мали альтернатив.

У 2008 році данським відділенням Google було розроблено V8 – високопродуктивний JavaScript рушій з відкритим кодом для використання в своїх браузерах компанії.

Незабаром після цього Райан Даль випустив крос-платформне середовище з відкритим вихідним кодом під назвою Node.js, що звільнило JavaScript від обмежень браузера і призвело до його нинішньої популярності. А все через те, що з Node.js з’явилася можливість запускати JavaScript код за межами браузера.

Зрештою, результатом цього стала велика популярність JavaScript за рахунок того, що тепер його можна використовувати не тільки для клієнтської частини браузерних, а й для всіх видів додатків, включаючи серверні, мобільні та десктопні.

У даному дослідженні JavaScript використовувався для написання як клієнтської, так і серверної частини застосунку.

2.6 Node.js

З моменту першого випуску Node.js відбулося багато цікавих подій [10]. Однією з найпомітніших був випуск Node Package Manager (NPM) у 2010 році. NPM - це інтерфейс командного рядка, який дозволяє легко встановлювати та керувати пакунками (тобто фрагментами коду, які можна використовувати повторно) для додатків на Node.js і на вересень 2022 року, згідно офіційному сайту Node.js, NPM мав більш ніж 2.1 мільйон пакунків. Це значно спростило розробникам обмін та повторне використання коду, а також сприяло зростанню екосистеми Node.js.

Ще однією важливою віхою в історії Node.js стало створення Node.js Foundation у 2015 році. Node.js Foundation - це некомерційна організація, яка сприяє розвитку та впровадженню Node.js, і її підтримують такі великі компанії, як Microsoft, IBM, PayPal та Intel. Це дало Node.js величезний поштовх до зростання довіри та допомогло зміцнити його позиції як ключової технології для Інтернету.

У 2019 році проєкт Node.js перейшов на нову каденцію випусків: релізи довгострокової підтримки (LTS) виходять кожні шість місяців, а нові функції - кожні чотири місяці. Це допомогло зберегти Node.js свіжим і сучасним, забезпечуючи при цьому стабільність роботи корпоративних додатків.

Зрештою, це зробило Node.js досить поширеною бібліотекою, популярність якої не перестає зростати з самого її випуску.

Node.js був використаний для реалізації серверної частини додатку, що дозволило скористатися його перевагами в продуктивності та масштабованості.

2.7 React

Історія React починається з 2010 року, коли Джордан Вок, інженер-програміст у Facebook, створив ранній прототип React під назвою "FaxJS" [11]. Він хотів створити подібну бібліотеку для JavaScript, яка б полегшила створення та підтримку складних користувацьких інтерфейсів.

У 2011 році React був вперше розгорнутий у стрічці новин Facebook. Пізніше, у 2012 році, він був розгорнутий в Instagram. У 2013 році React став загальнодоступним.

З того часу React став одним з найпопулярніших фреймворків для розробки інтерфейсів у світі. Його використовують компанії будь-якого розміру для створення високопродуктивних та масштабованих web-додатків.

Деякими з причин його популярності є:

1. Компонентна архітектура: React використовує компонентну архітектуру, що означає, що користувацькі інтерфейси будуються з менших компонентів, які можна використовувати повторно.

2. Декларативний синтаксис: React використовує декларативний синтаксис, що означає, що розробники описують, як має виглядати інтерфейс, а React піклується про його рендеринг. Це робить код React більш читабельним і легшим для налагодження.

3. Велика спільнота та екосистема: React має велику та активну спільноту розробників. Це означає, що розробникам React доступна велика кількість ресурсів, включаючи бібліотеки, інструменти та навчальні посібники.

React - це фреймворк, який постійно розвивається, а команда розробників постійно додає нові функції та покращення.

React був використаний для реалізації клієнтської частини додатку, що дозволило створити динамічний та інтерактивний користувацький інтерфейс.

2.8 MongoDB

MongoDB була створена Дуайтом Меррiманом та Еліотом Горовіцем, які працювали розробниками в компанії DoubleClick, що пізніше була придбана Google [12]. Працюючи над проєктом з управління великими обсягами даних у кількох центрах обробки даних, вони виявили, що традиційні реляційні бази даних не забезпечують необхідної їм продуктивності та масштабованості. Вони вирішили створити власну систему баз даних, яка згодом стала MongoDB.

MongoDB швидко завоювала популярність серед розробників завдяки своїй гнучкості, масштабованості та простоті використання. Вона також має широкий спектр можливостей, включаючи підтримку індексування, реплікації, шардингу та агрегації.

Сьогодні MongoDB використовують деякі з найбільших світових компаній, включаючи Adobe, eBay, Forbes та The New York Times, для управління та обробки своїх даних. Компанія MongoDB Inc. також розробила ряд супутніх продуктів і послуг, включаючи хмарний хостинг, аналітику та консалтингові послуги. Деякими перевагами MongoDB є:

1. Висока швидкість/продуктивність. Документно-орієнтована природа MongoDB дозволяє їй працювати швидше, ніж традиційним реляційним базам даних.

2. Гнучкість. MongoDB - це система баз даних NoSQL, що робить її придатною як для структурованих, так і для неструктурованих даних.

MongoDB була використана в якості бази даних для цього застосунку, забезпечуючи надійне та ефективне управління даними.

2.9 Visual Studio Code

Visual Studio Code - це легкий, але потужний і повністю безкоштовний редактор вихідного коду, який працює на ПК і доступний для Windows, macOS та Linux. Він має вбудовану підтримку JavaScript, TypeScript і Node.js та багату екосистему розширень для інших мов і середовищ виконання (таких як C++, C#, Java, Python, PHP, Go, .NET). Ключовими можливостями Visual Studio Code є [13]:

1. Редагування коду. VS Code забезпечує підсвічування синтаксису, автозавершення та форматування коду для різних мов програмування, що полегшує розробникам написання та редагування коду.

2. Інтегрований термінал. VS Code постачається з інтегрованим терміналом, що дозволяє розробникам виконувати команди та запускати скрипти, не виходячи з редактора.

3. Інтеграція з Git. VS Code має вбудовану підтримку Git, що дозволяє розробникам легко керувати змінами вихідного коду та співпрацювати з іншими розробниками.

4. Розширення. VS Code має багату екосистему розширень, які забезпечують додаткову функціональність і підтримку конкретних мов програмування, фреймворків та інструментів.

5. IntelliSense. VS Code надає інтелектуальне завершення коду, пропозиції щодо коду та підказки щодо параметрів для широкого спектру мов та фреймворків, що полегшує розробникам швидке та точне написання коду.

6. Налаштовуваний користувацький інтерфейс. VS Code має настроюваний користувацький інтерфейс, який дозволяє розробникам персоналізувати своє середовище кодування за допомогою тем, іконок та інших налаштувань.

Visual Studio Code був використаний як основний інструмент для розробки, забезпечуючи ефективне та зручне середовище для написання коду.

2.10 Postman

Postman - це інструмент розробки API (application programming interface), який допомагає створювати, тестувати та модифікувати API. Практично будь-яка функціональність, яка може знадобитися будь-якому розробнику, інкапсульована в цьому інструменті. Його використовують понад 5 мільйонів розробників щомісяця, щоб зробити розробку API легкою та простою. Ось ключові аспекти, які роблять Postman потужним інструментом у сфері розробки API:

1. Універсальні методи запитів. Postman підтримує безліч методів HTTP-запитів, включаючи GET, POST, PUT, DELETE і PATCH. Ця універсальність дозволяє розробникам всебічно взаємодіяти з API.

2. Гнучкі формати тіла запиту. Розробники отримують вигоду від гнучкості обробки різних форматів тіла запиту, включаючи дані форми, URL-кодовані дані, необроблені дані та двійкові дані. Ця адаптивність задовольняє різноманітні вимоги різних API.

3. Організоване тестування API. Колекції в Postman слугують потужним організаційним інструментом, що дозволяє розробникам ефективно класифікувати та керувати запитами до API.

Postman був використаний для тестування та налагодження API, що забезпечило якісну і надійну інтеграцію серверної частини з клієнтським інтерфейсом.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ WEB-ЗАСТОСУНКУ ДЛЯ ВИКЛАДЕННЯ, ПОШУКУ ТА ОЦІНКИ РЕЦЕПТІВ

3.1 Серверна частина

Серверна частина додатку — це складова програмного забезпечення, яка відповідає за обробку запитів від клієнтської сторони, взаємодію з базою даних та забезпечення потрібного функціоналу. Це окремий компонент, який виконується на сервері та забезпечує взаємодію між клієнтами (зазвичай web-браузерами) та даними, що знаходяться на сервері.

Основні аспекти серверної частини додатку включають:

1. Обробка запитів. Серверна частина відповідає за обробку запитів, які надходять від клієнтів. Це може бути HTTP-запити, які надсилаються web-браузером або іншими клієнтськими програмами.

2. Взаємодія з базою даних. Багато додатків потребують доступу до даних, які зберігаються в базі даних. Серверна частина відповідає за взаємодію з базою даних, включаючи читання, запис, оновлення та видалення даних.

3. Бізнес-логіка. В середині серверної частини зазвичай міститься бізнес-логіка додатку, яка визначає, як саме повинні оброблятися запити, які дії повинні виконуватися та які дані повинні бути повернуті клієнту.

4. Відправлення відповідей клієнтам. Нарешті, серверна частина відправляє відповіді клієнтам на їхні запити. Це може бути HTML-код для відображення web-сторінок, JSON-дані для використання в клієнтському JavaScript або інші формати даних.

Усі ці компоненти спільно працюють для забезпечення ефективного та безпечного функціонування додатку. Серверна частина може бути написана на різних мовах програмування та використовувати різні технології в залежності від вимог та потреб проєкту.

3.1.1 Створення моделей

Модель у контексті web-застосунків — це абстракція, яка представляє структуру даних і взаємодію з ними у базі даних. Вона визначає, які поля та типи даних будуть використовуватись, як ці дані будуть зберігатись та оброблятись. Модель використовується для створення, читання, оновлення та видалення (CRUD) даних у базі даних. Моделі були розроблені для MongoDB за допомогою Mongoose. MongoDB — це документно-орієнтована NoSQL база даних, яка зберігає дані у вигляді документів у форматі JSON. Mongoose — це об'єктно-документний моделювальник (ODM) для Node.js, який надає зручний спосіб роботи з MongoDB, забезпечуючи структуру та валідацію даних через схеми та моделі.

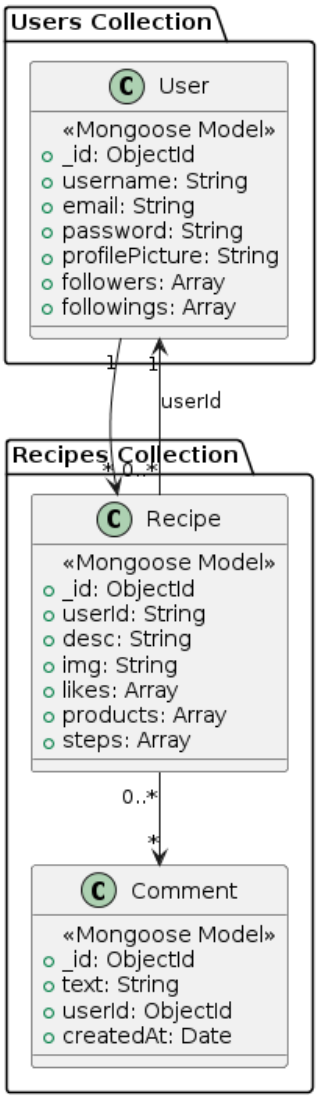


Рисунок 3.1.1.1 – Схема бази даних

Модель користувача має такі поля:

1. username - ім'я користувача, яке повинно бути унікальним, мінімальна довжина 7 символів, максимальна — 20 символів. Використовується для збереження імені користувача.
2. email - електронна пошта користувача, яка також повинна бути унікальною і має максимальну довжину 50 символів. Використовується для входу.
3. password - пароль користувача, мінімальна довжина 6 символів. Використовується для аутентифікації користувача.
4. profilePicture – зображення профілю користувача. Використовується для відображення аватару користувача.
5. followers - масив користувачів, які стежать за цим користувачем.
6. followings - масив користувачів, за якими стежить цей користувач.

Рисунок 3.1.1.2 – Приклад об’єкту моделі User

Модель рецепту має такі поля:

1. userId - ідентифікатор користувача, який створив рецепт.
2. desc - опис рецепту, який може містити до 500 символів.
3. img - зображення рецепту.
4. likes - масив користувачів, які вподобали цей рецепт.
5. products - масив продуктів, які потрібні для приготування рецепту, де кожен продукт має поле "name".

6. steps - масив кроків приготування рецепту. Кожен крок має поле "description", яке є обов'язковим, і може містити зображення кроку "stepImg".
7. comments - масив коментарів до рецепту, які представлені об'єктами з полями "text", "userId" та "createdAt".

Рисунок 3.1.1.3 – Приклад об’єкту моделі Recipe

3.1.2 Створення маршрутизації

Маршрутизація у web-застосунках — це процес визначення, які дії повинні виконатися при надходженні конкретного HTTP-запиту на певний URL-адрес. Вона відповідає за направлення запитів користувачів до відповідних частин вашого web-застосунку або API.

Маршрутизація використовується для:

1. Організація логіки застосунку: маршрутизація допомагає розділити web-застосунок на логічні частини, такі як реєстрація користувача, управління контентом або взаємодія з базою даних. Кожен маршрут може вказувати на конкретний контролер або обробник, який відповідає за виконання певних дій.

2. Визначення дій для кожного URL-шляху: маршрутизація дозволяє вказати, які дії повинні бути виконані при доступі до конкретного URL-шляху. Наприклад, застосунок може мати окремий маршрут для отримання списку користувачів, інший - для їх реєстрації та так далі.

3. Обробка HTTP-запитів: кожен HTTP-метод (GET, POST, PUT, DELETE) може мати свій власний маршрут, що визначає, які дії виконати при цьому типі запиту. Це дозволяє вам взаємодіяти з вашими даними та клієнтами відповідно до стандартів HTTP.

Маршрутизація допомагає вашому web-застосунку бути організованим, легко зрозумілим та підтримуваним, що дозволяє ефективно працювати над його розвитком та розширенням.

При розробці додатку, для маршрутизації було використано фреймворк Express, що є одним з найпопулярніших фреймворків для створення web-застосунків на платформі Node.js.

Express.js — це мінімалістичний та гнучкий web-фреймворк для Node.js, який дозволяє легко створювати web-застосунки та API. Він надає ряд корисних функцій, таких як маршрутизація, обробка запитів та відповідей, шаблонізація, обробка помилок та інше.

При розробці застосунку даного типу важливо забезпечити користувача можливістю реєстрації. Реєстрація є одним із перших кроків, які виконує користувач при взаємодії з web-застосунком і відіграє ключову роль у публікації рецептів. Для цього можна використати маршрут реєстрації користувача (див. рисунок 3.1.2.1).

```
router.post('/register', async (req, res) => {
    try {
        const salt = await bcrypt.genSalt(10);
        const hashedPassword = await bcrypt.hash(req.body.password, salt);
        const newUser = new User({
            username: req.body.username,
            email: req.body.email,
            password: hashedPassword,
        });
        const user = await newUser.save();
        console.log(`user from backend:` + user);
        res.status(200).json(user);
    } catch (error) {
        res.status(500).json(error);
        console.log(error);
    }
});
```

Рисунок 3.1.2.1 – Маршрут реєстрації користувача

Також, після реєстрації, треба забезпечити користувача можливістю авторизації. Це можна зробити за допомогою маршруту авторизації (див. рисунок 3.1.2.2).

```
router.post('/login', async (req, res) => {
    try {
        const user = await User.findOne({ email: req.body.email });
        if (!user) {
            res.status(404).send('User not found');
        }
        const validPassword = await bcrypt.compare(
            req.body.password,
            user.password
        );
        if (!validPassword) {
            res.status(400).json('Wrong password');
        }
        res.status(200).json(user);
    } catch (error) {
        res.status(500).json(error);
    }
});
```

Рисунок 3.1.2.2 – Маршрут авторизації користувача

Після реєстрації і авторизації важливо, щоб користувач мав можливість створювати свої рецепти, адже це є однією з ключових функцій додатку. Для цього використовується маршрут створення рецепту (див. рисунок 3.1.2.3).

```
router.post('/', async (req, res) => {
    const newRecipe = new Recipe(req.body);
    try {
        const savedRecipe = await newRecipe.save();
        res.status(200).json(savedRecipe);
    } catch (error) {
        res.status(500).json(error);
    }
});
```

Рисунок 3.1.2.3 – Маршрут створення рецепту

Після того, як користувач створив свій рецепт, важливо забезпечити йому можливість видаляти його за потреби. Це дозволить користувачам керувати своїм контентом, видаляючи рецепти. Для цього використовується маршрут видалення рецепту (див. рисунок 3.1.2.4).

```
router.delete('/:id', async (req, res) => {
    try {
        const recipe = await Recipe.findById(req.params.id);

        await recipe.deleteOne();
        res.status(200).json('The recipe has been deleted');
    } catch (error) {
        res.status(500).json(error);
    }
});
```

Рисунок 3.1.2.4 – Маршрут видалення рецепту

Під час розробки web-застосунку, важливо також забезпечити користувача можливістю отримувати рецепти. Для цього було розроблено декілька маршрутів.

Наприклад, маршрут перегляду рецептів конкретного користувача (див. рисунок 3.1.2.5) використовується на сторінці його профілю.

```
router.get('/profile/:username', async (req, res) => {
    try {
        const user = await User.findOne({ username: req.params.username });
        const recipes = await Recipe.find({ userId: user._id });
        res.status(200).json(recipes);
    } catch (error) {
        res.status(500).json(error);
    }
});
```

Рисунок 3.1.2.5 – Маршрут перегляду рецептів користувача

Маршрут перегляду рецептів користувача та користувачів, на яких він підписаний, (див. рисунок 3.1.2.6) використовується на головній сторінці.

```
router.get('/timeline/:userId', async (req, res) => {
    try {
        const currentUser = await User.findById(req.params.userId);
        const userRecipes = await Recipe.find({ userId: currentUser.id });
        const friendRecipes = await Promise.all(
            currentUser.followings.map((friendId) => {
                return Recipe.find({ userId: friendId });
            })
        );
        res.status(200).json(userRecipes.concat(...friendRecipes));
    } catch (error) {
        res.status(500).json(error);
    }
});
```

Рисунок 3.1.2.6 - Маршрут перегляду рецептів користувача та користувачів, на яких він підписаний

Маршрут пошуку рецептів (див. рисунок 3.1.2.7) дозволяє користувачам знаходити потрібні рецепти за назвою.

```
router.get('/search/:desc', async (req, res) => {
    try {
        const desc = req.params.desc;
        if (!desc || typeof desc !== 'string') {
            return res
                .status(400)
                .json(
                    'Description parameter is required and must be a string for search'
                );
        }

        const recipes = await Recipe.find({
            desc: { $regex: new RegExp(desc, 'i') },
        });
        if (recipes.length === 0) {
            return res.status(404).json('Recipes not found');
        }

        res.status(200).json(recipes);
    } catch (error) {
        res.status(500).json(error);
    }
});
```

Рисунок 3.1.2.7 - Маршрут пошуку рецептів

Маршрут отримання найпопулярніших рецептів за лайками (див. рисунок 3.1.2.8) надає користувачам можливість швидко знайти та переглянути найбільш популярні страви, які отримали найбільшу кількість оцінок від інших користувачів.

```
router.get('/top-liked', async (req, res) => {
    try {
        const topRecipes = await Recipe.find().sort({ likes: -1 }).limit(10);
        res.json(topRecipes);
    } catch (err) {
        res.status(500).json({ message: err.message });
    }
});
```

Рисунок 3.1.2.8 - Маршрут отримання найпопулярніших рецептів за лайками

Маршрут отримання найпопулярніших рецептів за лайками за поточний місяць (див. рисунок 3.1.2.9) надає користувачам можливість відкрити доступ до найбільш улюблених та актуальних страв, які отримали найбільшу кількість лайків протягом поточного місяця.

```
router.get('/top-liked-this-month', async (req, res) => {
    try {
        const currentDate = new Date();

        const firstDayOfMonth = new Date(
            currentDate.getFullYear(),
            currentDate.getMonth(),
            1
        );

        const lastDayOfMonth = new Date(
            currentDate.getFullYear(),
            currentDate.getMonth() + 1,
            0
        );

        const topRecipesCurrentMonth = await Recipe.find({
            createdAt: { $gte: firstDayOfMonth, $lte: lastDayOfMonth },
        })
            .sort({ likes: -1 })
            .limit(10);

        res.json(topRecipesCurrentMonth);
    } catch (err) {
        res.status(500).json({ message: err.message });
    }
});
```

Рисунок 3.1.2.9 - Маршрут отримання найпопулярніших рецептів за лайками за поточний місяць

Маршрут отримання випадкового рецепту (див. рисунок 3.1.2.10) дозволяє користувачам отримати доступ до страви, яка вибирається випадковим чином зі списку, створюючи непередбачуваний та цікавий досвід для користувачів.

```
router.get('/random', async (req, res) => {
    try {
        const count = await Recipe.countDocuments();
        const random = Math.floor(Math.random() * count);
        const randomRecipe = await Recipe.findOne().skip(random);

        if (!randomRecipe) {
            return res.status(404).json({ message: 'Recipe not found' });
        }
        res.status(200).json(randomRecipe);
    } catch (error) {
        res.status(500).json(error);
    }
});
```

Рисунок 3.1.2.10 - Маршрут отримання випадкового рецепту

Також важливо забезпечити користувачу можливість реагувати на рецепти. Одним з них є можливість залишати лайки під рецептами, реалізована за допомогою маршруту лайку рецепту (див. рисунок 3.1.2.11).

```
router.put('/:id/like', async (req, res) => {
    try {
        const recipe = await Recipe.findById(req.params.id);
        if (!recipe.likes.includes(req.body.userId)) {
            await recipe.updateOne({ $push: { likes: req.body.userId } });
            res.status(200).json('The post has been liked');
        } else {
            await recipe.updateOne({ $pull: { likes: req.body.userId } });
            res.status(200).json('The post has been disliked');
        }
    } catch (err) {
        res.status(500).json(err);
    }
});
```

Рисунок 3.1.2.11 - Маршрут лайку рецепту

Іншим способом є можливість залишити коментар під рецептом, яка була реалізована за допомогою маршруту створення коментаря (див. рисунок 3.1.2.12).

```
router.post('/:recipeId/comments', async (req, res) => {
    try {
        const { text, userId } = req.body;
        const recipe = await Recipe.findById(req.params.recipeId);
        recipe.comments.push({ text, userId });
        await recipe.save();
        res.status(201).json({ message: 'Comment created successfully' });
    } catch (error) {
        res.status(500).json(error);
    }
});
```

Рисунок 3.1.2.12 - Маршруту створення коментаря

Для того, щоб відображати коментарі, є свій окремий маршрут, який зображений на рисунку 3.1.2.13.

```
router.get('/:recipeId/comments/', async (req, res) => {
    try {
        const recipe = await Recipe.findById(req.params.recipeId);
        res.status(200).json(recipe.comments);
    } catch (error) {
        res.status(404).json(error);
    }
});
```

Рисунок 3.1.2.13 – Маршрут відображення коментарів

Маршрут підписки на іншого користувача (див. рисунок 3.1.2.14) надає можливість користувачам переглядати рецепти та активність інших користувачів, на яких вони підписані, прямо на головній сторінці web-застосунку. Це дозволяє користувачам зручно відстежувати та взаємодіяти з контентом своїх улюблених авторів.

```
router.put('/:id/follow', async (req, res) => {
    try {
        const user = await User.findById(req.params.id);
        const currentUser = await User.findById(req.body.userId);
        if (!user.followers.includes(req.body.userId)) {
            await user.updateOne({ $push: { followers: req.body.userId } });
            await currentUser.updateOne({
                $push: { followings: req.params.id },
            });
            res.status(200).json('User has been followed');
        } else {
            res.status(403).json('You already follow this user');
        }
    } catch (error) {}
});
```

Рисунок 3.1.2.14 - Маршрут підписки на іншого користувача

Також маршрут відписки від користувача (див. рисунок 3.1.2.15) дозволяє користувачам припинити стеження за активністю та видалити рецепти та оновлення від конкретного користувача з їхньої головної сторінки.

```
router.put('/:id/unfollow', async (req, res) => {
    try {
        const user = await User.findById(req.params.id);
        const currentUser = await User.findById(req.body.userId);
        if (user.followers.includes(req.body.userId)) {
            await user.updateOne({ $pull: { followers: req.body.userId } });
            await currentUser.updateOne({
                $pull: { followings: req.params.id },
            });
            res.status(200).json('User has been unfollowed');
        } else {
            res.status(403).json('You dont follow this user');
        }
    } catch (error) {}
});
```

Рисунок 3.1.2.15 – Маршрут відписки від користувача

Також, для забезпечення користувача можливостю публікувати фотографії, було створено маршрут публікації фото, який зображений на рисунку 3.1.2.16.

```
const storage = multer.diskStorage({
    destination: (req, file, cb) => {
        cb(null, 'public/images');
    },
    filename: (req, file, cb) => {
        cb(null, req.body.name);
    },
});

const upload = multer({ storage: storage });
app.post('/api/upload', upload.single('file'), (req, res) => {
    try {
        return res.status(200).json('File uploaded successfully');
    } catch (error) {
        console.error(error);
        return res.status(500).json('Error uploading file');
    }
});
```

Рисунок 3.1.2.16 – Маршрут публікації фото

Усі наведені маршрути, розроблені для забезпечення функціональності web-застосунку, грають важливу роль у наданні користувачам можливості повноцінної взаємодії з платформою. Маршрути для реєстрації, авторизації, створення, видалення та пошуку рецептів, а також для керування підписками і взаємодії з контентом, як-от лайки та коментарі, були ретельно інтегровані у клієнтську частину додатку.

3.2 Клієнтська частина

Клієнтська частина додатку — це інтерфейс, з яким взаємодіють користувачі. Вона представляє собою фронтенд, тобто частину додатку, яка відображається в браузері або іншому клієнтському програмному забезпеченні та забезпечує інтерактивність із користувачем.

Основні аспекти клієнтської частини додатку включають:

1. Відображення інтерфейсу користувача: Клієнтська частина відповідає за відображення графічного інтерфейсу користувача (GUI). Це може включати web-сторінки, мобільні додатки, десктопні програми та інші інтерфейси.

2. Інтерактивність: Клієнтська частина забезпечує можливість взаємодії з користувачем. Це може бути введення даних, вибір опцій, натискання кнопок та інші дії, які виконує користувач.

3. Відправлення запитів на сервер: Клієнтська частина може надсилати запити на серверну частину додатку для отримання даних, збереження змін або виконання певних операцій.

4. Обробка відповідей від сервера: Після надсилання запитів клієнтська частина отримує відповіді від сервера, які можуть бути використані для оновлення інтерфейсу користувача або виконання інших дій.

Клієнтська частина додатку може бути реалізована за допомогою різних технологій та підходів, а одним з найпопулярніших фреймворків для розробки клієнтської частини є React.

React — це бібліотека для створення інтерфейсів користувача, яка дозволяє розробникам будувати динамічні та інтерактивні web-сторінки та додатки. Вона базується на компонентах, які дозволяють розділити інтерфейс на невеликі, повторно використовувані частини. React використовує власну віртуальну DOM для ефективного оновлення сторінки при зміні даних, що робить додатки, розроблені з його використанням, швидкими та ефективними.

3.2.1 Сторінка реєстрації

Користувач починає свою взаємодію з сайтом саме зі сторінки реєстрації (див. рисунок 3.2.1.1). Це перша точка контакту між ним і додатком. Сторінка реєстрації надає користувачу можливість створення облікового запису, що дозволяє йому отримати доступ до функціоналу та сервісів додатку.

На цій сторінці користувач вводить основні дані для створення свого облікового запису, такі як ім'я користувача, електронна пошта та пароль.

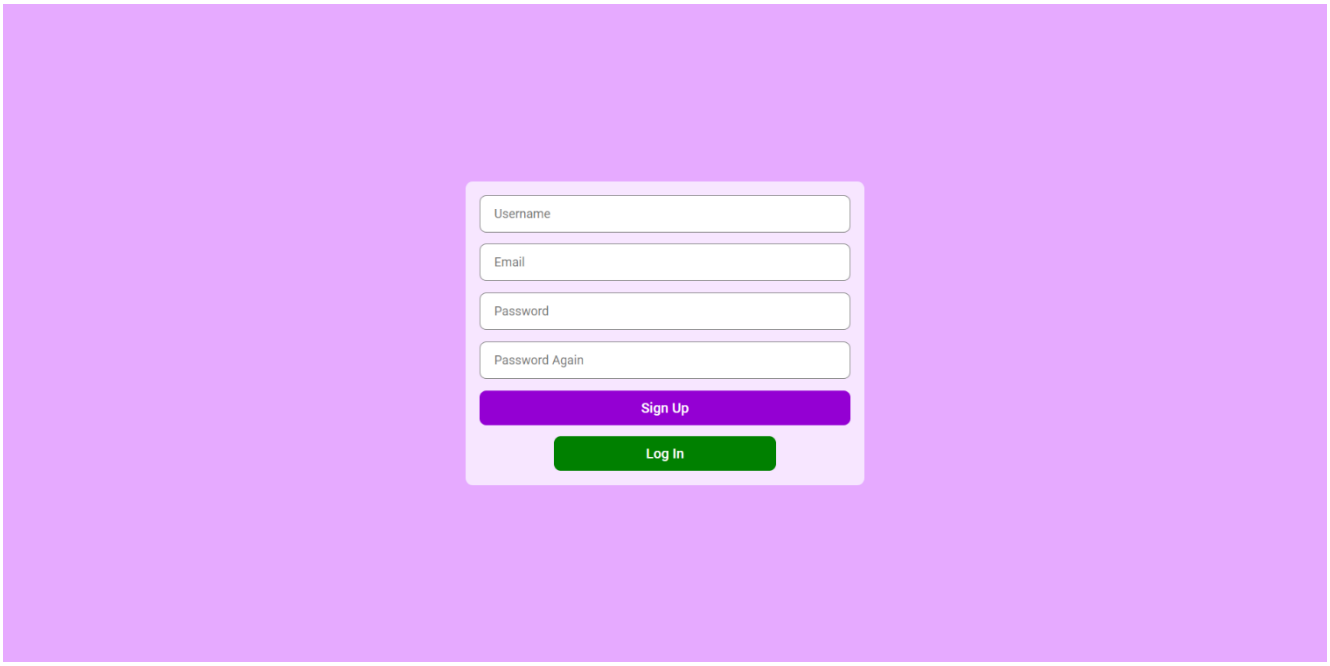


Рисунок 3.2.1.1 Сторінка реєстрації

Також на цій сторінці була створена валідація:

1. Користувач має ввести дані у всі поля.
2. Поле електронної пошти має містити знак «@» та символи після.
3. Пароль має бути не менше 6 символів.
4. У двох полях для введення паролю вони повинні бути однаковими.

Після введення цих даних, користувач натискає кнопку реєстрації, що передає дані до маршрутизатора реєстрації за допомогою окремої функції (див. рисунок 3.2.1.2).

```
const handleClick = async (e) => {
    e.preventDefault();
    if (passwordAgain.current.value !== password.current.value) {
        passwordAgain.current.setCustomValidity('Passwords dont match');
    } else {
        const user = {
            username: username.current.value,
            email: email.current.value,
            password: password.current.value,
        };
        try {
            await axios.post(
                'http://localhost:5000/api/auth/register',
                user
            );
            setIsRegistered(true);
        } catch (error) {
            console.log(error);
        }
    }
};
```

Рисунок 3.2.1.2 – Функція реєстрації

Також користувач має можливість перейти на сторінку авторизації кнопкою входу, якщо у нього вже є створений обліковий запис.

3.2.2 Сторінка авторизації

Якщо користувач вже здійснив реєстрацію, він має можливість увійти в свій обліковий запис. Це можна зробити на сторінці авторизації (див. рисунок 3.2.2.1). Для цього користувач має ввести свої електронну пошту та пароль. На цій сторінці використана така ж валідація, як і на сторінці реєстрації.

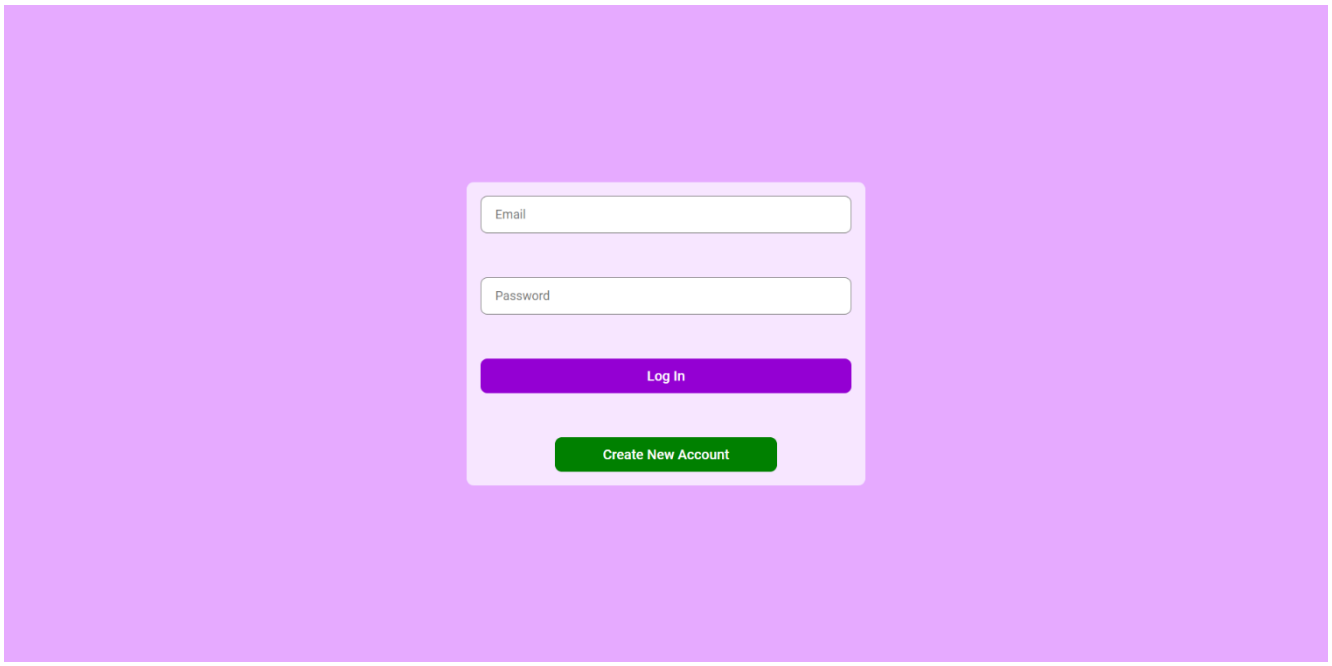


Рисунок 3.2.2.1 – Сторінка авторизації користувача

Після вводу даних, користувач натискає кнопку входу, що передає дані до маршрутизатора авторизації за допомогою окремої функції (див рисунок 3.2.2.2).

```
const handleSubmit = async (e) => {
    e.preventDefault();
    try {
        const response = await axios.post(
            'http://localhost:5000/api/auth/login',
            {
                email: email.current.value,
                password: password.current.value,
            }
        );
        dispatch({ type: 'LOGIN_SUCCESS', payload: response.data });
        setLoggedIn(true);
    } catch (error) {
        console.log(error.response.data);
    }
};
```

Рисунок 3.2.2.2 – Функція авторизації

Також користувач має можливість перейти на сторінку реєстрації кнопкою «Створити новий аккаунт», якщо він ще не має облікового запису.

Для зберігання авторизації було створено AuthContext (див. рисунок 3.2.2.3) разом із AuthReducer. AuthContext використовується для управління станом авторизації користувачів та забезпечення доступу до цього стану у всьому додатку. Функції LoginStart, LoginSuccess, LoginFailure, Follow, UnFollow і UpdateProfilePicture визначають дії, які можуть змінювати стан авторизації, а AuthReducer використовується для виконання цих дій та зміни стану AuthContext відповідно до виконаних дій користувача.

```
const INITIAL_STATE = {
    user: JSON.parse(localStorage.getItem('user')) || null,
    isFetching: false,
    error: false,
};

export const AuthContext = createContext(INITIAL_STATE);

export const AuthContextProvider = ({ children }) => {
    const [state, dispatch] = useReducer(AuthReducer, INITIAL_STATE);

    useEffect(() => {
        const user = JSON.parse(localStorage.getItem('user'));
        if (user) {
            dispatch({ type: 'LOGIN_SUCCESS', payload: user });
        }
    }, []);

    return (
        <AuthContext.Provider
            value={{
                user: state.user,
                isFetching: state.isFetching,
                error: state.error,
                dispatch,
            }}
        >
            {children}
        </AuthContext.Provider>
    );
};
```

Рисунок 3.2.2.3 – Контекст авторизації

3.2.3 Головна сторінка

Після того, як користувач успішно пройшов процес реєстрації або авторизації, його перенаправляють на головну сторінку додатку. Головна сторінка є центральним місцем, з якого користувач може отримати доступ до основного функціоналу та ресурсів додатку.

Головна сторінка (див. рисунок 3.2.3.1) складається з двох компонентів: верхньої панелі та стрічки.

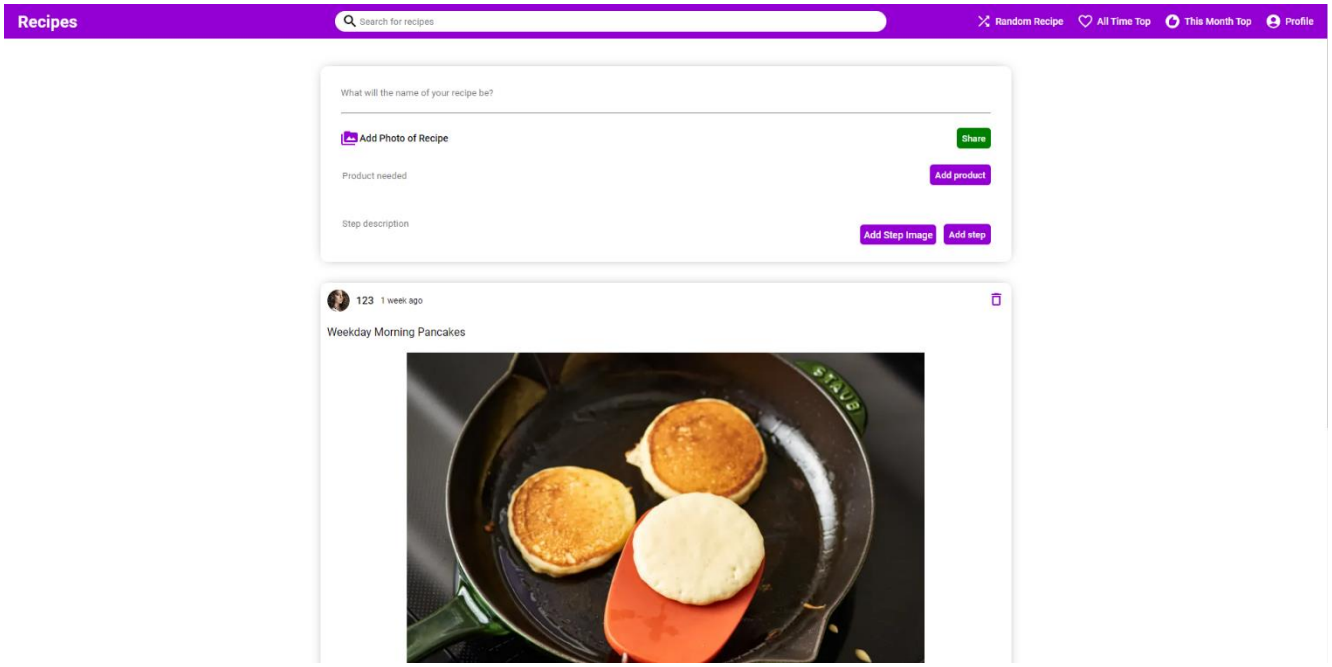


Рисунок 3.2.3.1 – Головна сторінка

Верхня панель має наступні елементи:

1. Логотип: кнопка, розташована у верхньому лівому куті, призначена для переходу на головну сторінку.
2. Поле пошуку рецептів: це текстове поле, де користувач може вводити пошуковий запит для знаходження певних рецептів.
3. Кнопка «Випадковий рецепт»: ця кнопка дозволяє користувачу отримати випадковий рецепт.

4. Кнопка «Найкращі рецепти за весь час»: ця кнопка призначена для перегляду 10 найкращих рецептів, які були опубліковані.
5. Кнопка «Найкращі рецепти за поточний місяць»: ця кнопка дозволяє переглянути 10 найкращих рецептів за поточний місяць.
6. Кнопка переходу до профілю користувача: ця кнопка дозволяє користувачу перейти на свій профіль.

Ці елементи забезпечують користувачам доступ до основних функцій і можливостей додатку.

Стрічка рецептів - це основна частина додатку, де користувач може бачити рецепти, які були створені ним самим, а також рецепти інших користувачів, на яких він підписаний. Вона також складається з двох компонентів: форми публікації рецепту та самих рецептів.

Форма рецептів дозволяє користувачу заповнити інформацію рецепту, таку як:

1. Назву рецепту.
2. Головну фотографію рецепту.
3. Продукти, необхідні для приготування.
4. Кроки приготування та окремі фотографії до них.

Після введення даних до форми, користувач має можливість публікації цього рецепту натиснувши кнопку «Поділитися», яка за допомогою окремої функції (див. рисунок 3.2.3.2) відправляє дані до свого маршрутизатору і цей рецепт публікується.

```
const submitHandler = async (e) => {
    e.preventDefault();
    const newPost = {
        userId: user._id,
        desc: desc.current.value,
        img: '',
        products: productInputs.map((product) => ({ name: product.value })),
    };

    try {
        if (file) {
            const data = new FormData();
            const fileName = Date.now() + file.name;
            data.append('name', fileName);
            data.append('file', file);
            newPost.img = fileName;
            await axios.post('http://localhost:5000/api/upload', data);
        }
        const postedSteps = await Promise.all(
            stepInputs.map(async (step) => {
                if (step.stepImage) {
                    const stepImageData = new FormData();
                    const stepImageName = Date.now() + step.stepImage.name;
                    stepImageData.append('name', stepImageName);
                    stepImageData.append('file', step.stepImage);
                    await axios.post(
                        'http://localhost:5000/api/upload',
                        stepImageData
                    );
                    return {
                        description: step.description,
                        stepImg: stepImageName,
                    };
                } else {
                    return { description: step.description, stepImg: '' };
                }
            })
        );
        newPost.steps = postedSteps;
        await axios.post('http://localhost:5000/api/recipe', newPost);
        window.location.reload();
    } catch (err) {
        console.error(err);
    }
};
```

Рисунок 3.2.3.2 – Функція публікації рецепту

Компонент рецепту – це компонент, який відповідає за відображення інформації рецепту та взаємодію з ним. На рисунку 3.2.3.3 можна побачити, як виглядає цей компонент. Він відображає всю необхідну інформацію, таку як зображення рецепту, назва, опис, список інгредієнтів та кроки приготування. Крім

того, компонент включає можливість залишати коментарі, лайки та видалити рецепт, якщо ви є його автором.

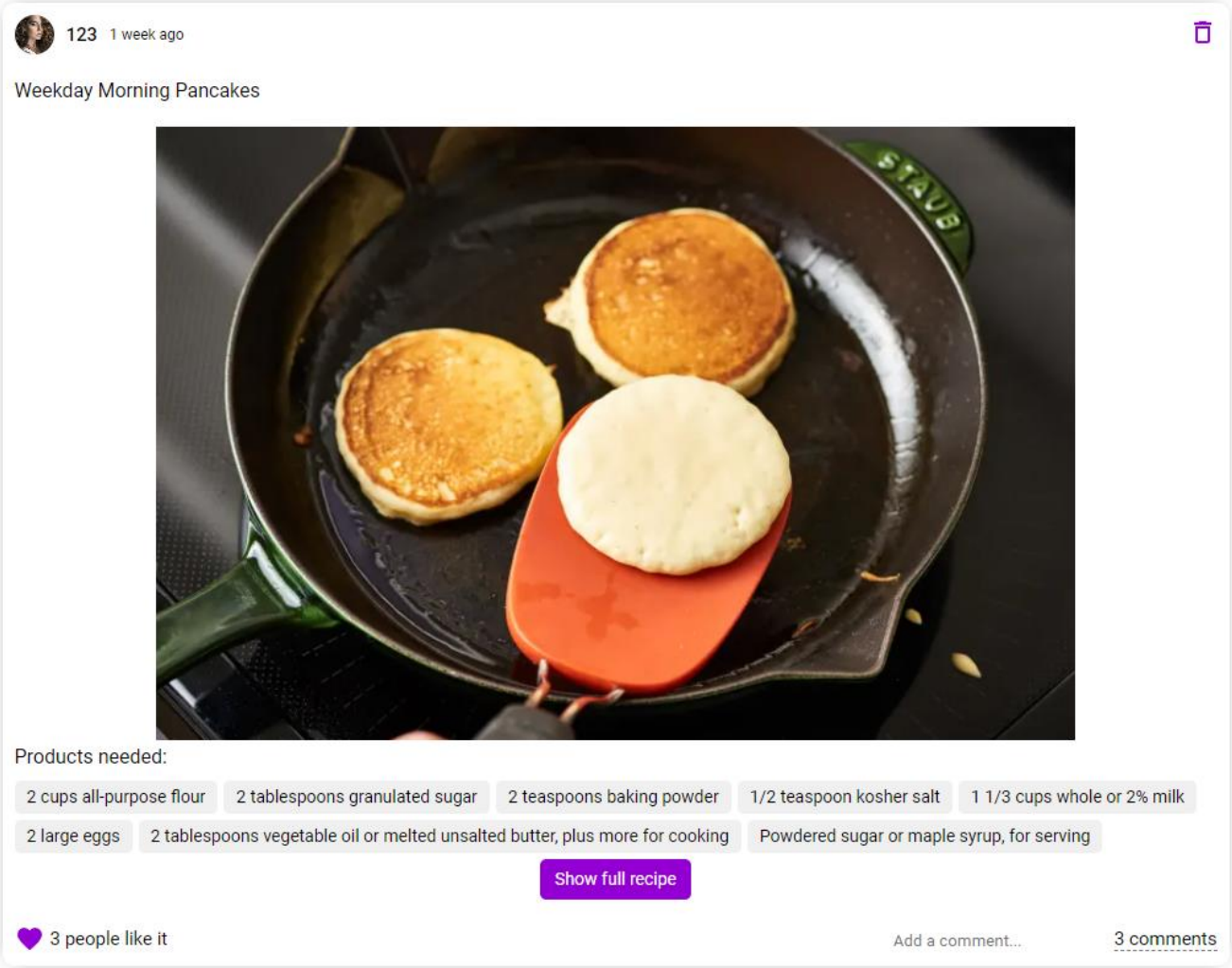


Рисунок 3.2.3.3 – Приклад рецепту

Для того, щоб дати користувачу можливість виразити своє вподобання, було створено функцію (див. рисунок 3.2.3.4), яка дозволяє залишити або прибрати лайк, викликаючи викликає свій маршрутизатор.

```
const likeHandler = () => {
    try {
        axios.put(`http://localhost:5000/api/recipe/${recipe._id}/like`, {
            userId: currentUser._id,
        });
    } catch (error) {
        console.error(error);
    }
    setLike(isLiked ? like - 1 : like + 1);
    setIsLiked(!isLiked);
};
```

Рисунок 3.2.3.4 – Функція для кнопки лайку

Аналогічно, функція для публікації коментарів, створена для того, щоб користувачі могли залишати свої думки, спостереження та рецептурні поради щодо конкретного рецепту. Функцію продемонстровано на рисунку 3.2.3.5.

```
const handleKeyPress = async (e) => {
    if (e.key === 'Enter') {
        try {
            const res = await axios.post(
                `http://localhost:5000/api/recipe/${recipe._id}/comments`,
                { userId: currentUser._id, text: inputText }
            );
            const newComment = {
                _id: res.data._id,
                userId: currentUser._id,
                text: inputText,
                createdAt: Date.now(),
            };
            setInputText('');
            setComments([...comments, newComment]);
            console.log(newComment);
            fetchComments();
            setCommentCount(commentCount + 1);
        } catch (error) {
            console.error(error);
        }
    }
};
```

Рисунок 3.2.3.5 – Функція публікації коментарів

Для того, щоб користувач мав можливість видалити рецепт, було створено функцію (див. рисунок 3.2.3.6), яка викликає відповідний маршрут для видалення рецепту.

```
const deleteRecipeHandler = async () => {
    try {
        await axios.delete(
            `http://localhost:5000/api/recipe/${recipe._id}`
        );
        window.location.reload();
    } catch (error) {
        console.log(error);
    }
};
```

Рисунок 3.2.3.6 – Функція для видалення рецепту

3.2.4 Сторінка профілю

Після успішної авторизації користувач також може перейти на сторінку свого профілю, яка зображена на рисунку 3.2.4.1. Сторінка профілю є особистим простором користувача, де він може переглядати свої рецепти та створювати нові. Також користувач має доступ до інших користувачів, на яких він підписаний.

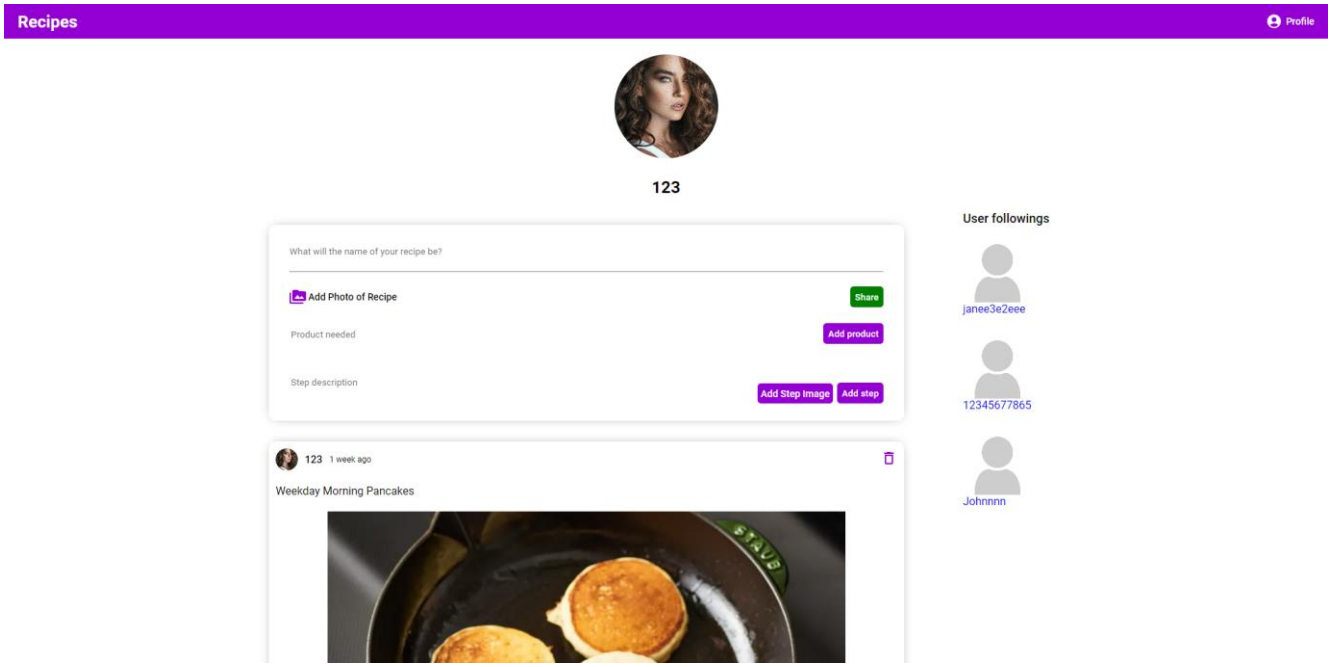


Рисунок 3.2.4.1 – Профіль користувача

Сторінка також складається з верхньої панелі, стрічки, форми публікації рецепту та правої панелі.

Права панель створена для відображення користувачів, на яких підписаний поточний користувач. З неї також можна перейти на сторінку потрібного користувача, яка зображена на рисунку 3.2.4.2 та там на нього підписатися.

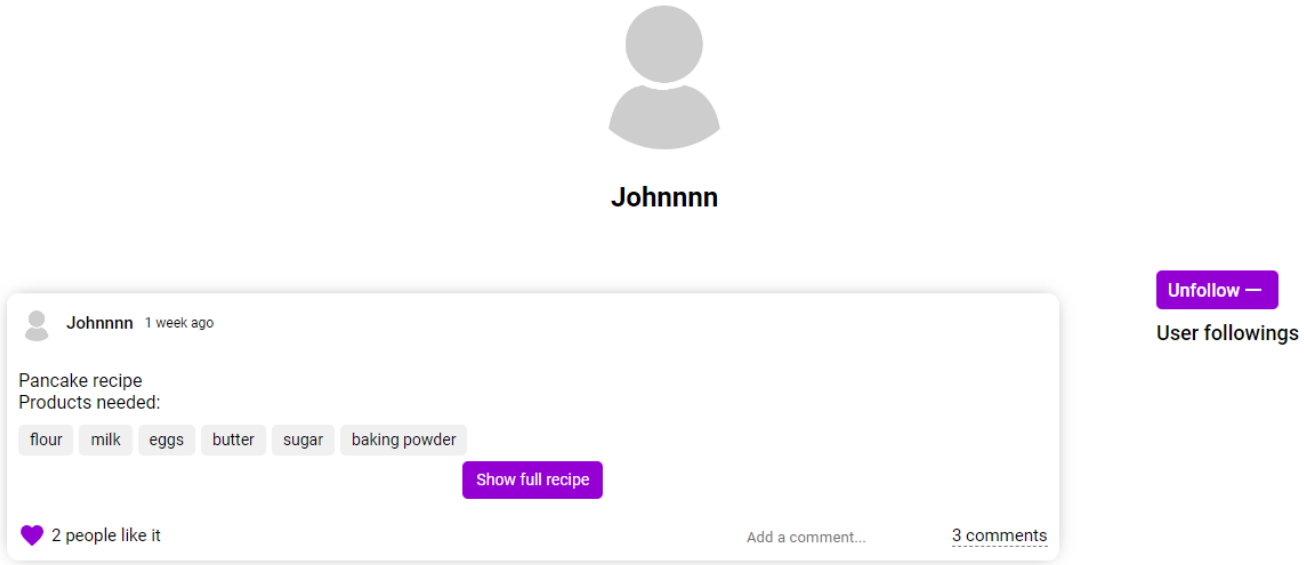


Рисунок 3.2.4.2 – Сторінка профілю іншого користувача

Крім того, на своїй сторінці користувач може змінити своє фото профілю. Для цього йому потрібно навести курсор на своє фото, клікнути та вибрати потрібну фотографію. Це реалізовано за допомогою функції, зображеної на рисунку 3.2.4.3.

```
const handleFileChange = async (e) => {
    const selectedFile = e.target.files[0];
    if (selectedFile) {
        setFile(selectedFile);
        const data = new FormData();
        const fileName = Date.now() + selectedFile.name;
        data.append('name', fileName);
        data.append('file', selectedFile);

        try {
            await axios.post('http://localhost:5000/api/upload', data);
            const updatedUser = { ...user, profilePicture: fileName };
            await axios.put(`http://localhost:5000/api/users/${user._id}`, {
                profilePicture: fileName,
            });
            setUser(updatedUser);
        } catch (err) {
            console.error(err);
        }
    }
};
```

Рисунок 3.2.4.3 – Функція заміни фотографії профілю

3.3 Тестування

Тестування програмного забезпечення - це метод оцінки його функціональності. Цей процес перевіряє, чи відповідає програмне забезпечення очікуваним вимогам і чи не містить помилок. Основна мета тестування - виявлення помилок, несправностей або невідповідностей вимогам. Тестування зосереджене на вимірюванні відповідності специфікаціям, функціональності та продуктивності програмного забезпечення або додатку.

Таблиця 3.3.1

Тестування

	Назва	Кроки	Очікуваний результат	Реальний результат
1	Перевірка роботи застосунку	1. Відкриття застосунку	Застосунок відкривається	Застосунок відкривається
2	Перевірка сторінки реєстрації	1. Відкриття застосунку 2. Введення даних для реєстрації 3. Натискання кнопки "Зареєструватися"	Користувача реєструє і переводить на головну сторінку	Користувача реєструє і переводить на головну сторінку
3	Перевірка сторінки реєстрації	1. Відкриття застосунку 2. Натискання кнопки входу	Користувача переводить на сторінку авторизації	Користувача переводить на сторінку авторизації
4	Перевірка сторінки авторизації	1. Відкриття застосунку 2. Введення даних для авторизації 3. Натискання кнопки "Ввійти"	Користувача авторизує і переводить на головну сторінку	Користувача авторизує і переводить на головну сторінку

Продовження таблиці 3.3.1

	Назва	Кроки	Очікуваний результат	Реальний результат
5	Перевірка верхньої панелі	1. Відкриття застосунку 2. Авторизація 3. Ввод назви до поля пошуку	Відображення рецептів за назвою	Відображення рецептів за назвою
6	Перевірка верхньої панелі	1. Відкриття застосунку 2. Авторизація 3. Натискання на логотип	Користувача переносить на головну сторінку	Користувача переносить на головну сторінку
7	Перевірка верхньої панелі	1. Відкриття застосунку 2. Авторизація 3. Натискання на кнопку випадкового рецепту	У стрічці відображається випадковий рецепт	У стрічці відображається випадковий рецепт
8	Перевірка верхньої панелі	1. Відкриття застосунку 2. Авторизація 3. Натискання на кнопку кращих рецептів	У стрічці відображаються 10 рецептів з найбільшою кількістю лайків	У стрічці відображаються 10 рецептів з найбільшою кількістю лайків
9	Перевірка верхньої панелі	1. Відкриття застосунку 2. Авторизація 3. Натискання на кнопку кращих рецептів за поточний місяць	У стрічці відображаються 10 рецептів з найбільшою кількістю лайків за поточний місяць	У стрічці відображаються 10 рецептів з найбільшою кількістю лайків за поточний місяць

Продовження таблиці 3.3.1

	Назва	Кроки	Очікуваний результат	Реальний результат
10	Перевірка верхньої панелі	1. Відкриття застосунку 2. Авторизація 3. Натискання на кнопку профілю	Користувача переносить на сторінку профілю	Користувача переносить на сторінку профілю
11	Перевірка викладення рецепту	1. Відкриття застосунку 2. Авторизація 3. Ввод інформації про рецепт 4. Натискання кнопки публікації	Рецепт створюється	Рецепт створюється
12	Перевірка рецепту	1. Відкриття застосунку 2. Авторизація 3. Натискання кнопки лайку	Кількість лайків зростає або спадає в залежності від того, чи користувач вже поставив лайк	Кількість лайків зростає або спадає в залежності від того, чи користувач вже поставив лайк
13	Перевірка рецепту	1. Відкриття застосунку 2. Авторизація 3. Ввод коменратя 4. Натискання клавіші Enter	Коментар додається	Коментар додається

Продовження таблиці 3.3.1

	Назва	Кроки	Очікуваний результат	Реальний результат
14	Перевірка сторінки профілю	1. Відкриття застосунку 2. Авторизація 3. Перехід на сторінку профілю 4. Натискання на фотографію профілю 5. Вибір фотографії	Фотографія профілю змінюється на обрану	Фотографія профілю змінюється на обрану
15	Перевірка сторінки профілю	1. Відкриття застосунку 2. Авторизація 3. Перехід на сторінку профілю іншого користувача 4.Натискання кнопки підписки	Кількість користувачів, на яких підписаний поточний користувач, зростає або спадає в залежності від того, чи користувач вже підписався	Кількість користувачів, на яких підписаний поточний користувач, зростає або спадає в залежності від того, чи користувач вже підписався

ВИСНОВКИ

Дане дослідження було спрямовано на проєктування та реалізацію web-застосунку для викладення, пошуку та оцінки рецептів з використанням Node.js та React. Усі поставлені завдання були виконані у повному обсязі.

1. Аргументовано актуальність розробленого web-застосунку для викладення, пошуку та оцінки рецептів на основі аналізу наукової та практичної літератури з розробки програмних продуктів.

2. Шляхом аналізу близьких за функціоналом продуктів (Cookpad, Kitchen Stories, FoodYub) досліджено та встановлено переваги та недоліки трьох аналогів, на основі чого сформовано та описано функціональні та нефункціональні вимоги до розробленого продукту.

3. Обґрунтовано вибір засобів розробки web-застосунку для викладення, пошуку та оцінки рецептів: використано бібліотеку Express для створення серверної частини, Bcrypt для забезпечення безпеки паролів, Mongoose для взаємодії з базою даних MongoDB, MUI для розробки користувацького інтерфейсу в середовищі React, Axios для виконання HTTP-запитів

4. Спроєктовано та розроблено web-застосунок для викладення, пошуку та оцінки рецептів за допомогою Node.js та React, особливістю якого є можливість отримати випадковий рецепт та пропозицію популярних рецептів на основі кількості лайків.

5. Здійснено тестування web-застосунку для викладення, пошуку та оцінки рецептів за допомогою Node.js та React., отримані задовільні результати.

Розроблений застосунок створений з метою надати можливість користувачам отримувати натхнення та ділитися своїми кулінарними витворами з іншими. Його використання відкрито для будь-якої зацікавленої особи, хто має бажання експериментувати у кулінарній галузі або просто поділитися своїми улюбленими рецептами.

6. Робота пройшла апробацію:

Тимошенко М.П, Шевченко С.М. Розробка Web-застосунку для викладення, пошуку та оцінки рецептів з використанням Node.js та React: Матеріали Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях». 24 квітня 2024р.; Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2024. С. 73.

ПЕРЕЛІК ПОСИЛАНЬ

1. Cookpad [Електронний ресурс] – Режим доступу до ресурсу: https://cookpad.com.
2. Food Yub [Електронний ресурс] – Режим доступу до ресурсу: https://foodyub.com/.
3. Kitchen Stories [Електронний ресурс] – Режим доступу до ресурсу: https://www.kitchenstories.com.
4. Python Web Development Tutorials [Електронний ресурс] – Режим доступу до ресурсу: https://realpython.com/tutorials/web-dev/.
5. Advantages and Disadvantages of PHP [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://www.geeksforgeeks.org/advantages-and-disadvantages-of-php/.
6. 12 pros and cons of Java for your project [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: https://anywhere.epam.com/en/blog/pros-and-cons-java.
7. Go Programming Language for Web Development: An In-Depth Analysis [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://medium.com/@purwantorenaldi/go-programming-language-for-web-development-an-in-depth-analysis-5882fe5c7fdb.
8. Top 10 Reasons to Learn JavaScript [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: https://www.simplilearn.com/reasons-to-learn-javascript-article.
9. Introduction [Електронний ресурс] – Режим доступу до ресурсу: https://launchschool.com/books/javascript/read/introduction.
10. BRIEF HISTORY OF NODEJS [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://medium.com/@ogbuuzoma413/brief-history-of-nodejs-de0cac0af448.
11. The History of React.js: A Story of Innovation and Community [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://www.linkedin.com/pulse/history-reactjs-story-innovation-community-l-anderson.
12. A Brief History of MongoDB [Електронний ресурс] – Режим доступу до ресурсу: https://www.tutorjoes.in/mongodb_tutorial/history_of_mongodb.
13. Visual Studio Code Features [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://www.linkedin.com/pulse/visual-studio-code-features-pramuditha-madura.
14. What is Software Testing? [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: https://www.geeksforgeeks.org/software-testing-basics/.

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

РОЗРОБКА WEB-ЗАСТОСУНКУ ДЛЯ ВИКЛАДЕННЯ, ПОШУКУ ТА ОЦІНКИ РЕЦЕПТІВ З ВИКОРИСТАННЯМ NODE.JS ТА REACT

Виконав студент 4 курсу
групи ПД-42
Тимошенко Максим Петрович
Керівникроботи
К.п.н., доц., доцент кафедри ІПЗ Шевченко Світлана Миколаївна

Київ – 2024

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи** – спрощення процесу викладення, пошуку та оцінки рецептів за допомогою технологій Node.js та React.
- **Об’єкт дослідження** – процес викладення, пошуку та оцінки рецептів.
- **Предмет дослідження** – розробка web-застосунку для викладення, пошуку та оцінки рецептів з використанням технологій Node.js та React.

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Проаналізувати існуючі додатки для викладення, пошуку та оцінки рецептів, виявити їх переваги та недоліки.
2. Розробити функціональні й нефункціональні вимоги на основі проаналізованих даних для додатку для викладення, пошуку та оцінки рецептів.
3. Дослідити інструменти розробки додатку для викладення, пошуку та оцінки рецептів.
4. Спроектувати інтерфейс та архітектуру додатку.
5. Розробити web-додаток, враховуючи спроектовані інтерфейс та архітектуру.
6. Провести тестування додатку.

3

АНАЛІЗ АНАЛОГІВ

Назва	Kitchen Stories	FoodYub	Cookpad	Додаток
Платформа	Web, Android, iOs	Web	Web, Android, iOs	Web
Можливість створення рецепту власноруч	+	+	+	+
Спрощений доступ до створення рецепту (без переходу на інші сторінки)	-	-	-	+
Випадковий рецепт	-	-	-	+
Формат оцінки рецептів	Лайки	Лайки	Емодзі	Лайки
Можливість підписки на інших користувачів	-	+	+	+
Можливість коментування рецептів	+	-	+	+
Можливість пошуку рецепту за назвою	+	+	+	+
Рейтинг найкращих рецептів на основі кількості лайків	-	-	-	+

4

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Функціональні

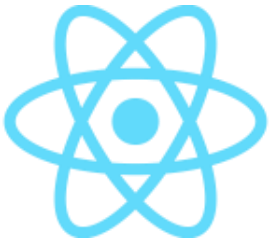
1. Можливість створення та збереження рецептів користувачами в особистому кабінеті.
2. Можливість коментування рецептів, яка сприяє взаємодії між користувачами та вдосконаленню страв
3. Можливість додавати фотографії до готових страв або процесу приготування рецептів.
4. Можливість користувачів оцінювати рецепти за допомогою лайків.
5. Можливість підписатися на інших користувачів.
6. Можливість замінити фотографію профілю.
7. Можливість перегляду рецептів.
8. Сортування рецептів за датою створення.
9. Сортування рецептів за кількістю лайків.

Нефункціональні

1. Шифрування паролів.
2. Доступ до створення рецептів з будь якої сторінки сайту.
3. Кросбраузерність

5

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ



React

MUI

6

ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ

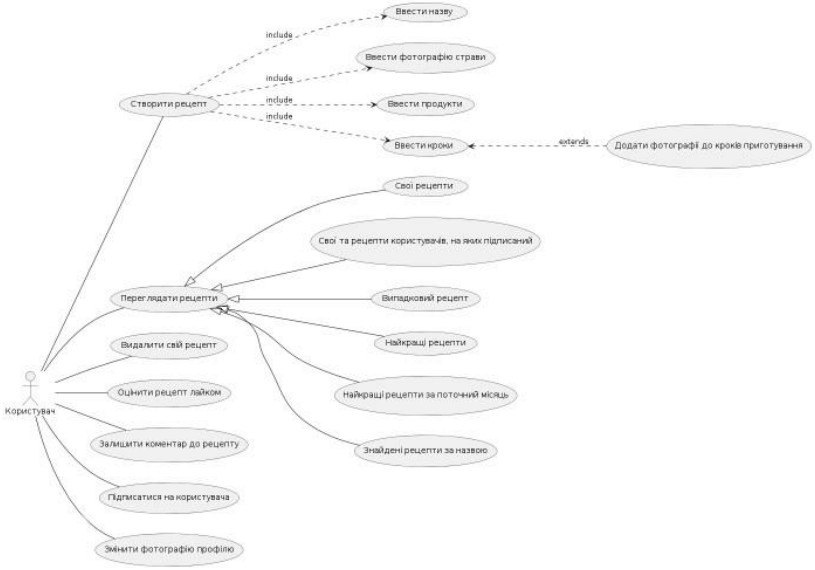
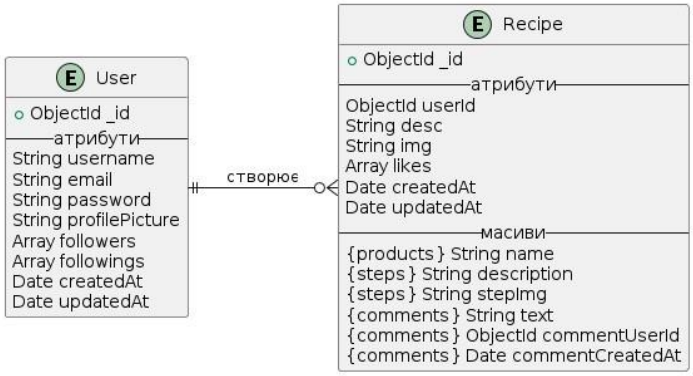
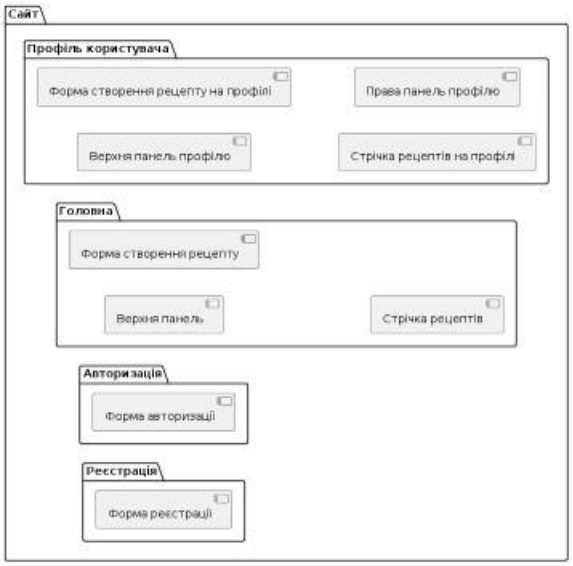


СХЕМА БАЗИ ДАНИХ

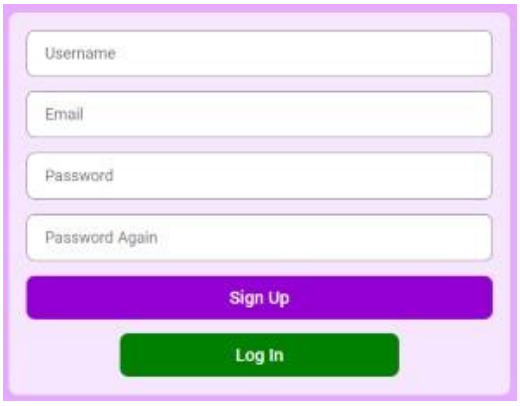


ДІАГРАМА ПАКЕТІВ

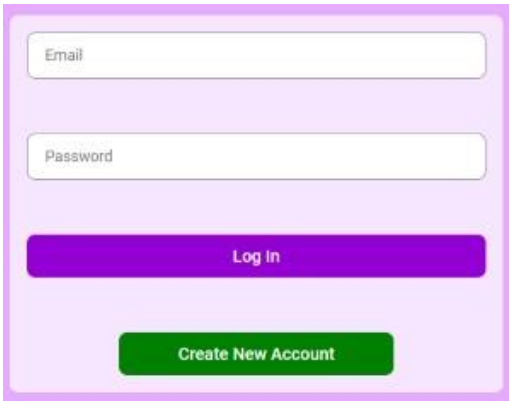


9

ЕКРАННІ ФОРМИ



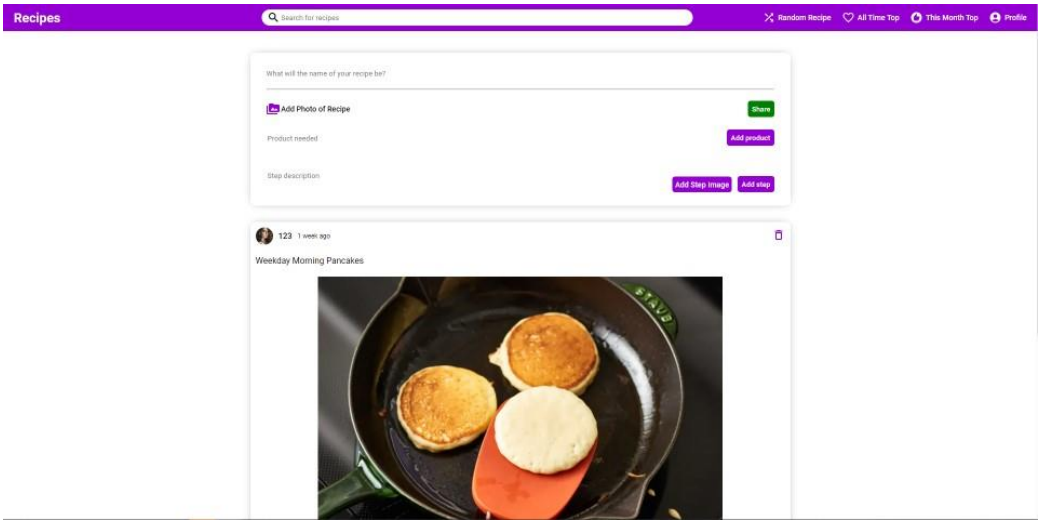
Реєстрація



Авторизація

10

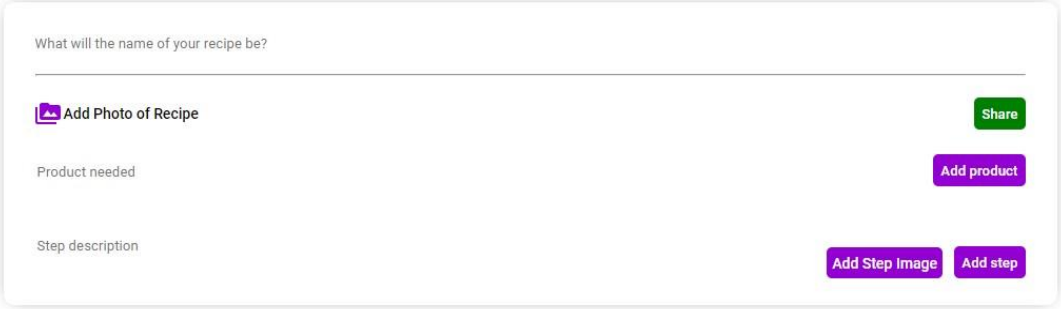
ЕКРАННА ФОРМА



Головна сторінка

11

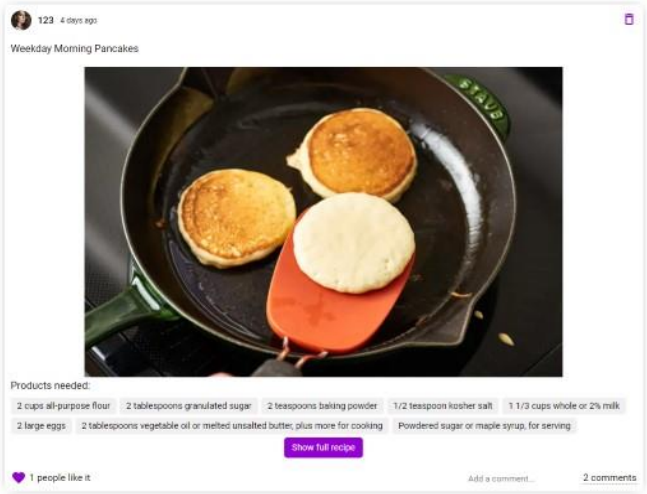
ЕКРАННА ФОРМА



Створення рецепту

12

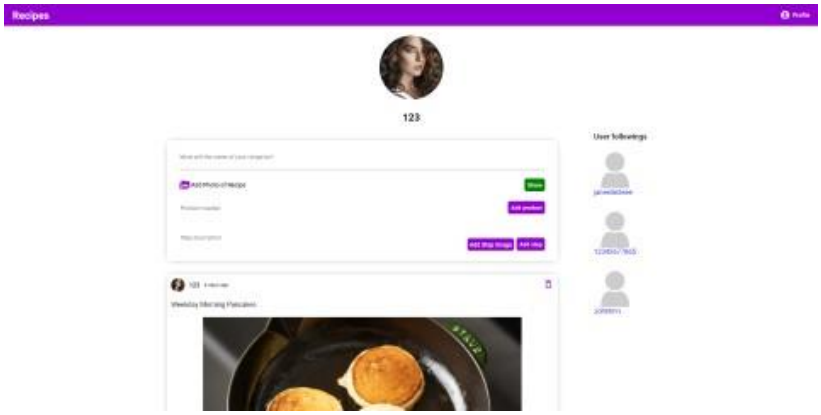
ЕКРАННА ФОРМА



Рецепт

13

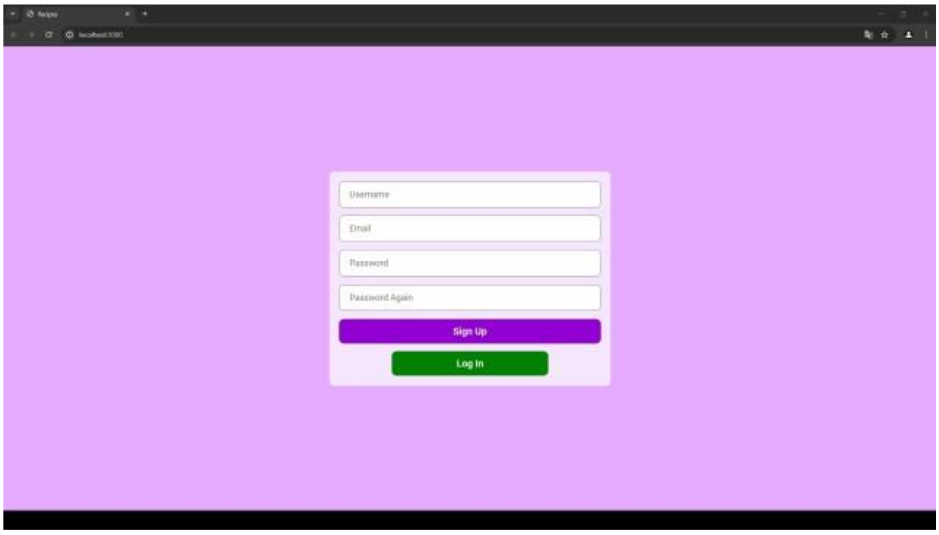
ЕКРАННА ФОРМА



Профіль користувача

14

ВІДЕО ПРЕДСТАВЛЕННЯ



15

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Тимошенко М.П, Шевченко С.М. Розробка Web-застосунку для викладення, пошуку та оцінки рецептів з використанням Node.js та React: Матеріали всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях». 24 квітня 2024 р.; Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2024. С. 73.

16