МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД «УНІВЕРСИТЕТ «КРОК»
Фаховий коледж Університету «КРОК»

ДИПЛОМНА РОБОТА
за темою
«Розробка та створення сайту бібліотеки з рецептами та способами їх приготування»

Студент 4 курсу групи ІПЗ 20к-1

Керівник дипломної роботи
_____Викладач_____
(посада керівника)

___Талалайко Кіріл Романович__ (прізвище, ім'я та по-батькові студента)

_____Кириченко В.В _____ (прізвище, ім'я та по-батькові керівника)

_____До _захисту_____
(резолюція «До захисту»)

_____ (підпис студента)

_____11.06.2024_____ (дата)

_____ (підпис викладача)

Зміст

ВСТУП

Проект веб-застосунку "Шукач рецептів – 'Healthy'", розпочався з ідеї для допомоги зайнятим людям, із різноманітними харчовими потребами, знайти та спланувати свій план харчування з корисних рецептів.

Цей проект націлений на перетворення планування дієти на індивідуально створений, доступний та приємний досвід надаючи користувачам зрозумілу платформу для пошуку та впорядкування їхніх улюблених рецептів.

Проект був розроблений за допомогою методології P3-express, Agile та системи пріоритезації. З технологій було використано стек "MERN" (MongoDB – база даних, Express.js – система для використання застосункових інтерфейсів API, React.js – фреймворк для JavaScript для створення інноваційного та адаптивного дизайну застосунку, Node.js – менеджер пакетів застосунку.), цей стек забезпечив зручну, розділену та масштабо-здатну роботу застосунку.

"Healthy" розроблений спеціально для користувачів, які піклуються про своє здоров'я. Це спрощує пошук рецептів, які відповідають певним дієтам, включаючи веганську, вегетаріанську та середземноморську. Ця веб-програма зосереджена на популярних дієтах, підкреслюючи зручність і економію часу для користувачів, які прагнуть досліджувати нові ідеї їжі, адаптовані до їхніх харчових потреб. З великим наголосом на зручному дизайні застосунок спрощує планування, враховуючи різноманітні харчові вимоги та перетворюючи це на приємний і легкий досвід.

Серед досягнень проекту були розробка та успішне впровадження основних функціональних можливостей, таких як безпечна реєстрація та система входу, дієтичні та харчові фільтри, збереження рецептів та клієнто-орієнтовне відображення інформації про рецепти відповідно до системи

пріоритезації (див. Таблиця 1.2.1.). Крім того, додаток має адаптивний дизайн і зрозумілий інтерфейс, що забезпечує бездоганний досвід для користувача.

Незважаючи на зіткнення з проблемами, пов'язаними з часовими та технологічними обмеженнями і доступністю ресурсів, я продемонстрував значні навички вирішення проблем і здатність до адаптації. Завдяки бажанню покращуватись я подолав ці перешкоди, отримавши цінний урок створення проекту, технічного розвитку та новітніх технологій. Висновком є висока важливість клієнто-орієнтованого дизайну, гнучкого підходу до проблем та повного використання технологій для перетворення ідеї в реальність. Після успішного завершення я прагну до постійного самовдосконалення та нових викликів.

РОЗДІЛ 1. АНАЛІЗ ІСНУЮЧОЮ ІНФОРМАЦІЇ.

1.1. Порівняльний аналіз існуючою інформації та рішень

У процесі розробки дипломної роботи "Розробка та створення сайту бібліотеки з рецептами та способами їх приготування" було проведено порівняльний аналіз існуючих інформаційних рішень та систем. Основна мета аналізу полягала у виявленні сильних та слабких сторін поточних рішень, щоб створити інноваційний та ефективний веб-додаток для користувачів.

Існуючі платформи для пошуку рецептів, такі як Allrecipes, Epicurious та Yummly, пропонують різноманітні можливості для користувачів, включаючи великий вибір рецептів, можливість фільтрації за інгредієнтами, дієтичними вимогами та типами страв. Однак, виявлені недоліки включають складність навігації, перевантаженість інтерфейсу рекламою та обмежені можливості персоналізації.

Веб-додаток "Healthy", орієнтований на усунення цих недоліків шляхом надання користувачам чистого, зручного інтерфейсу з розширеними функціями фільтрації та персоналізації. Використання технологічного стека MERN (MongoDB, Express.js, React.js, Node.js) дозволяє забезпечити високу швидкість взаємодії, масштабованість та безпеку даних користувачів.

Основна мета програми — зробити планування їжі більш доступним, персоналізованим і приємним для користувачів. Він задовольняє їхні унікальні кулінарні потреби та вподобання, орієнтуючись на широку аудиторію, яка

включає зайнятих професіоналів, ентузіастів кулінарії та людей, які усвідомлюють своє здоров'я та харчові звички.

Спрямований на задоволення різноманітних кулінарних потреб людей, які піклуються про своє здоров'я, зайнятих професіоналів і ентузіастів кулінарії. Веб-програма розроблена для спрощення процесу пошуку та впорядкування рецептів. Він прагне надати користувачам інструменти для вивчення нових страв, поживного харчування та бездоганної інтеграції планування їжі у свій розпорядок дня.

Обсяг проекту було чітко визначено, щоб включити надійний набір функцій: безпечну реєстрацію та вхід користувача, розширені дієтичні фільтри та фільтри страв, повне відображення рецептів та інтерактивні елементи для збереження та керування улюбленими стравами.

1.2. Постановка завдання на проектування

Основне завдання проектування веб-додатку "Healthy" полягало у створенні платформи, яка б не тільки відповідала, але й перевершувала очікування користувачів у контексті зручності використання та функціональності. Цілі проекту включали:

1. Створення безпечної системи реєстрації та входу. Забезпечення надійного зберігання даних користувачів з використанням Auth0 та MongoDB. Це дозволило створити систему, яка гарантує безпеку персональних даних, захищаючи їх від несанкціонованого доступу. Використання Auth0 забезпечує надійну автентифікацію, а MongoDB дозволяє безпечно зберігати інформацію, забезпечуючи її доступність та захист.

2. Розробка розширених дієтичних фільтрів. Включення можливостей фільтрації за веганськими, вегетаріанськими, середземноморськими дієтами та

іншими харчовими уподобаннями, що полегшує користувачам пошук відповідних рецептів, адаптованих до їхніх специфічних потреб і уподобань. Фільтри дозволяють швидко знайти рецепти, що відповідають певним дієтичним вимогам, зекономивши час та зусилля користувачів.

3. Інтеграція функції збереження рецептів. Користувачі отримали можливість зберігати свої улюблені рецепти для подальшого доступу та управління у своєму профілі. Це створює персоналізований досвід використання, дозволяючи користувачам легко повертатися до своїх улюблених рецептів, організовувати їх та ділитися ними з іншими.

4. Відображення детальної інформації про рецепти. Застосунок надає користувачам повну інформацію для приготування страв, включаючи інгредієнти, етапи приготування та харчові дані для кожного рецепту. Це допомагає користувачам точно дотримуватися рецептів, забезпечуючи приготування страв відповідно до заданих стандартів.

5. Забезпечення адаптивного дизайну. Використання React-bootstrap дозволило створити інтерфейс, що адаптується до різних пристроїв та розмірів екранів. Це забезпечує бездоганний досвід користувачів на всіх платформах, незалежно від того, чи користуються вони додатком на комп'ютері, планшеті чи смартфоні. Адаптивний дизайн гарантує зручність та доступність функцій застосунку для всіх користувачів.

Процес проектування включав ретельне планування та управління ризиками, включаючи технічні виклики та часові обмеження. Регулярні оцінки прогресу та гнучкий підхід до управління проектами дозволили вчасно реагувати на зміни та адаптуватися до нових умов, що сприяло успішному завершенню проекту.

За переглядом таблиці пріоритезації детально описано узгодження поставленого продукту з його цілями, підкреслюючи завершення всіх функцій «Обов'язково». Крім того, опис статусу «Бажано» та «Можливо» та «Проти».

1.2.1. Таблиця Пріоритезації

Пріоритетність	Опис цілі
Обов'язково	Система реєстрації та входу ✅
Обов'язково	Впровадження дієтичного фільтра ✅
Обов'язково	Впровадження фільтра їжі ✅
Обов'язково	Відображення результатів ✅
Обов'язково	Відображення інформації про рецепт ✅
Обов'язково	Функція збереження страв ✅
Обов'язково	Керування профілями користувачів ✅
Обов'язково	Відображення збережених страв ✅
Бажано	Налаштування профілю користувача
Бажано	Інтеграція соціальних мереж
Бажано	Пошук за ключовими словами
Можливо	Система рейтингу рецептів
Можливо	Підтримка контенту користувача
Можливо	Додаткові дієтичні фільтри
Проти	Система доєставки їжі
Проти	Відео-уроки з приготування
Проти	Список друзів

Імплементовані функції

Усі обов'язкові функції, які описані в таблиці Пріоритезації виконані. А саме:

- Система реєстрації та входу: Безпечна система, яка дозволяє користувачам реєструватися та входити в систему, надаючи персоналізований доступ до функцій програми. Використання Auth0 та MongoDB забезпечує надійну автентифікацію та зберігання даних, що гарантує безпеку персональної інформації користувачів.
- Впровадження дієтичного фільтра: Користувачі можуть фільтрувати рецепти на основі дієтичних потреб, таких як вегетаріанський, веганський, середземноморський, полегшуючи пошук персоналізованих рецептів. Це дозволяє користувачам швидко знаходити рецепти, що відповідають їхнім конкретним дієтичним вимогам, зекономлюючи час і зусилля.
- Впровадження фільтра їжі: Реалізовані фільтри дозволяють користувачам сортувати рецепти за типом страви, як-от закуска, сніданок, обід або вечеря, спрощуючи процес планування їжі. Це дає можливість користувачам легко організувати своє харчування відповідно до часу доби чи події.
- Відображення результатів: Програма динамічно відображає результати пошуку, пропонуючи користувачам варіанти рецептів, які відповідають вибраним фільтрам. Інтерфейс забезпечує швидкий доступ до необхідної інформації, що підвищує загальне задоволення від використання програми.
- Відображення інформації про рецепт: Користувачам доступна детальна інформація для кожного рецепту, включаючи інгредієнти, етапи приготування та харчові дані. Це допомагає користувачам точно слідувати рецептам та забезпечує інформованість про харчову цінність страв.

- Функція збереження страв: Користувачі можуть зберігати свої улюблені рецепти для легкого доступу та організації у своєму профілі. Це забезпечує персоналізований досвід використання, дозволяючи зберігати та організовувати улюблені рецепти для майбутнього використання.
- Упорядковане відображення збережених страв: Збережені рецепти впорядковано зручним для користувача способом, що полегшує користувачам навігацію та повторний перегляд улюблених страв. Це забезпечує легкий доступ до улюблених рецептів і допомагає в плануванні харчування.

Функції не або частково реалізовано

Незважаючи на те, що успішно запроваджено всі «обов'язкові» функції, забезпечивши надійну та функціональну основу для веб-програми «Healthy», деякі «бажані» та «можливі» функції було відкладено до майбутніх оновлень.

Це рішення було прийнято відповідно до стратегічного пріоритету, спрямованого на зосередження на якості та основних функціях для початкового запуску. Нижче наведено підхід і опис щодо цих функцій.

- Налаштування профілю користувача: Вдосконалення для більш глибокого налаштування профілю, включно з дієтичними обмеженнями та перевагами, заплановано на майбутні оновлення. Це дозволить користувачам більш точно налаштовувати свій профіль відповідно до індивідуальних потреб та вподобань.
- Інтеграція в соціальні мережі: Ця функція, спрямована на надання користувачам можливості ділитися рецептами на платформах соціальних мереж, буде інтегрована в наступну версію програми. Це дозволить користувачам ділитися улюбленими рецептами з друзями та сім'єю, збільшуючи взаємодію та залученість.

- Функція пошуку за ключовими словами: Можливості розширеного пошуку для пошуку рецептів на основі конкретних інгредієнтів або ключових слів не були включені в застосунок, але є пріоритетними для наступних оновлень. Це дозволить користувачам більш точно знаходити рецепти, що відповідають їхнім потребам, полегшуючи процес пошуку.

Функції зі статусом «Проти»

- Система доставки їжі: Підтверджено, що виходить за межі які початково визначені для проекту. Це рішення було прийнято, оскільки основна мета програми полягає у пошуку та збереженні рецептів, а не в організації доставки їжі.
- Відео-уроки з приготування: Також підтверджено, що вони виходять за межі, зосереджені на основній функції відкриття рецептів і планування їжі. Впровадження відео-уроків потребує значних ресурсів і не є першочерговим завданням для даного проекту.
- Список друзів: Підтверджено, що виходить за межі через низький рівень потенційного використання, оскільки застосунок для пошуку рецептів не є соціальною мережею. Основна мета програми полягає в наданні користувачам інструментів для планування харчування, а не в побудові соціальних зв'язків.

Випуск веб-застосунку «Шукач рецептів» із усіма «обов'язковими» функціями вказує на значне досягнення у забезпеченні всеосяжної та зручної платформи для користувачів, які стежать за дієтою. Майбутні оновлення будуть зосереджені на інтеграції цих функцій, реагуванні на відгуки користувачів і постійному покращенні взаємодії з користувачем.

РОЗДІЛ 2. ПРОЕКТНІ І ТЕХНІЧНІ РІШЕННЯ. ВИДИ ЗАБЕЗПЕЧЕННЯ

Проект веб-додатку "Шукачу Рецептів" розпочався з чіткого бачення: розробити планувальник їжі та шукач рецептів для цифрово-обізнаної аудиторії. "Healthy" — це інноваційний веб-додаток, призначений для зміни старомодного способу планування дієти та пошуку рецептів.

Ця програма, створена з урахуванням різноманітних дієтичних уподобань, спрощує процес пошуку відповідних рецептів, які відповідають індивідуальним дієтичним планам.

2.1. Інформаційне забезпечення

Інформаційне забезпечення в проекті веб-додатку "Healthy" включає організацію та структурування даних, необхідних для ефективної роботи додатку. Основними компонентами інформаційного забезпечення є:

1. База даних MongoDB. Вона використовується для зберігання інформації про користувачів, рецепти, дієтичні уподобання та інші необхідні дані. MongoDB забезпечує гнучкість та ефективність управління даними, що дозволяє легко адаптуватися до потреб користувачів та збільшувати обсяги збережених даних.

2. Функціональність фільтрації. Реалізована за допомогою REST API, яка отримує дані з бази даних на основі обраних користувачем фільтрів (категорії дієти та їжі) та повертає відповідні результати у форматі JSON. Це дозволяє користувачам швидко знаходити рецепти, що відповідають їхнім уподобанням.

3. Система збереження та відображення рецептів. Користувачі можуть зберігати свої улюблені рецепти, які будуть організовані у їх профілях для зручного доступу в майбутньому. Ця система забезпечує інтерактивний та персоналізований досвід користування додатком.

Дизайн та структура застосунку

- Дизайн, орієнтований на користувача: Програма була розроблена з акцентом на взаємодію з користувачем, кульмінацією якої став інтуїтивно зрозумілий і доступний інтерфейс. Основна увага приділялася створенню простого та логічного інтерфейсу, який дозволяє користувачам легко орієнтуватися у програмі та швидко знаходити необхідну інформацію. Для досягнення цієї мети використовувалися принципи UX/UI дизайну, включаючи зрозумілі навігаційні елементи, зручні форми та зрозумілі візуальні підказки. Такий підхід до проектування допоміг забезпечити відповідність програми високим стандартам зручності використання, що робить її привабливою для широкого кола користувачів.
- Комплексна функціональність: Успішна реалізація всіх визначених «обов'язкових» функцій, зокрема дієтичних і харчових фільтрів, відображення інформації про рецепти та керування профілями користувачів, заклала міцну основу для програми. Включення таких функцій як збереження рецептів, динамічне відображення результатів пошуку та детальна інформація про кожний рецепт забезпечують користувачам повний набір інструментів для планування харчування. У сукупності ці функції забезпечують комплексне рішення для планування їжі та пошуку рецептів, що відповідає основним цілям проекту.
- Спрощення планування їжі: Додаток ефективно оптимізував процес планування їжі. Це спрощення проявляється в зменшенні складності та

зусиль, необхідних для планування дієти, що безпосередньо узгоджується з метою проекту зробити планування їжі більш керованим і приємним досвідом. Завдяки інтуїтивно зрозумілому інтерфейсу та розширеним фільтрам користувачі можуть легко знайти потрібні рецепти, адаптувати їх під свої потреби та зберігати для подальшого використання.

- Покращений пошук рецептів: Програма може похвалитися розширеною базою даних рецептів у поєднанні зі складними параметрами фільтрації. Ця функція дозволяє користувачам легко шукати та знаходити рецепти, які відповідають їхнім дієтичним уподобанням, значно збагачуючи досвід пошуку рецептів. Використання ефективних алгоритмів пошуку та фільтрації гарантує, що користувачі отримують найрелевантніші результати за лічені секунди, що робить процес пошуку швидким та ефективним.

Такі підходи до дизайну та функціональності програми «Healthy» забезпечують високу зручність та задоволення користувачів, дозволяючи їм легко планувати своє харчування та знаходити нові, відповідні їхнім потребам рецепти.

2.2. Математичне забезпечення

Математичне забезпечення проекту полягає у використанні алгоритмів для обробки та фільтрації даних, а також для забезпечення ефективного функціонування додатку. Основні аспекти включають:

1. Алгоритми пошуку та фільтрації:
    - Обробка запитів користувачів: Використання ефективних алгоритмів для обробки запитів користувачів та фільтрації рецептів на основі

дієтичних уподобань та інших критеріїв. Це дозволяє швидко та точно знаходити потрібну інформацію серед великої кількості даних.

- Ранжування результатів: Алгоритми ранжування допомагають визначити найрелевантніші результати для користувача, враховуючи такі фактори, як популярність рецептів, рейтинги, та відповідність дієтичним потребам.
- Побудова рекомендацій: Використання методів машинного навчання для побудови рекомендацій на основі попередніх виборів та уподобань користувачів. Це допомагає запропонувати нові рецепти, які можуть бути цікавими для користувачів, забезпечуючи індивідуальний підхід до кожного.

2. Оптимізація продуктивності:
   - Індексація бази даних: Застосування методів індексації для швидкого доступу до даних у базі. Індексація дозволяє значно зменшити час пошуку необхідної інформації, підвищуючи ефективність роботи додатку.
   - Оптимізація коду: Аналіз та оптимізація коду для підвищення швидкості його виконання. Це включає рефакторинг коду для видалення зайвих операцій, зменшення використання пам'яті та покращення загальної продуктивності.
   - Налаштування серверних конфігурацій: Оптимізація серверних конфігурацій для забезпечення швидкої та надійної роботи додатку. Це може включати балансування навантаження, налаштування кешування, та використання асинхронних операцій для обробки запитів.
   - Аналіз та моніторинг продуктивності: Впровадження систем моніторингу для відстеження продуктивності додатку в реальному часі. Це дозволяє швидко виявляти та усувати проблеми, забезпечуючи стабільну роботу системи.

- Стиснення та оптимізація даних: Використання методів стиснення даних для зменшення обсягу переданих даних між клієнтом і сервером, що дозволяє зменшити час завантаження та підвищити швидкість роботи додатку.

3. Обробка великих даних:
   - Паралельна обробка: Використання паралельних обчислень для обробки великих обсягів даних. Це дозволяє значно підвищити продуктивність системи, розподіляючи навантаження на кілька процесорів або серверів.
   - Аналіз даних: Використання алгоритмів аналізу даних для виявлення тенденцій та патернів у поведінці користувачів. Це допомагає краще розуміти потреби користувачів та адаптувати функціонал додатку відповідно до їх вимог.
   - Прогнозування: Використання методів прогнозування для передбачення майбутніх дій користувачів на основі аналізу історичних даних. Це дозволяє покращити користувацький досвід та забезпечити більш персоналізовані рекомендації.

Таким чином, математичне забезпечення проекту "Шукач рецептів – 'Healthy'" є ключовим компонентом, що дозволяє ефективно обробляти дані, забезпечувати високу продуктивність та надавати користувачам індивідуальний підхід до пошуку та збереження рецептів.

2.3. Програмне забезпечення

Проект веб-додатку було створено з використанням стеку технологій MERN. Цей набір технологій допоміг досягти чітко виставлених цілей щодо створення масштабованої, ефективної та зручної програми.

Стек MERN, що складається з MongoDB, Express.js, React.js і Node.js, був обраний завдяки надійному поєднанню технологій, які разом створюють

повноцінне середовище розробки. Розширений технологічний стек веб-додатку "Healthy" включає:

1. MongoDB. Документо-орієнтована база даних яка пропонує гнучкість, необхідну для керування різноманітним і динамічним вмістом програми. Незалежно від того, чи йдеться про широкий спектр дієтичних уподобань, детальні інструкції щодо рецептів чи профілі користувачів, MongoDB адаптується до потреб продукту. Це забезпечує ефективне та безперебійне керування даними.

2. Express.js. Веб-фреймворк для Node.js, який використовується для створення серверних API та забезпечення взаємодії між фронтендом та бекендом додатку. Він надає потужні інструменти для створення веб-додатків і API. Його можна описати як "невидимий" шар між сервером Node.js та кінцевими додатками, який спрощує розробку за допомогою численних мідлвар (middleware), що дозволяє легко управляти маршрутами, обробляти запити і відправляти відповіді. Express надає функціональність, яка не входить в базовий пакет Node.js, таку як більш зручну обробку запитів і кращу інтеграцію з іншими бібліотеками і фреймворками. Це значно спрощує створення складних веб-додатків і мікросервісів, забезпечуючи розробникам швидкість і гнучкість у розгортанні серверного коду.

3. React.js. Фреймворк для створення користувацьких інтерфейсів, що забезпечує динамічне та інтерактивне відображення даних, а також адаптивний дизайн для різних пристроїв та екранів. Це серце дизайну інтерфейсу користувача. Його компонентна архітектура забезпечує широкі можливості налаштування, гарантуючи, що додаток не тільки відповідає, але й перевершує різноманітні очікування своєї бази користувачів. React.js сприяє швидкому оновленню та плавній взаємодії з користувачем, що робить планування їжі чудовим досвідом.

4. Node.js. Середовище виконання JavaScript на сервері, яке забезпечує виконання серверного коду та обробку запитів від клієнтів. Це платформа, яка допомагає у виконанні I/O-інтенсивних операцій, таких як читання файлів, доступ до мереж і баз даних, що забезпечує асинхронне програмування за допомогою подій, тобто взаємодії клієнта або користувача з додатком. Node.js заснований на JavaScript-двигуні від Google, який перекладає JavaScript-код у нативний машинний код, дозволяючи програмам працювати швидко і ефективно. Це робить Node.js ідеальним вибором для реалізації швидких і масштабованих мережевих додатків, які можуть обслуговувати велику кількість одночасних з'єднань без втрати продуктивності.

Комбінуючи Express.js і Node.js, розробники можуть створювати ефективні і гнучкі серверні рішення, які оптимізують взаємодію між фронтендом і бекендом, підтримуючи високий рівень абстракції та контроль над потоком обробки даних. Це забезпечує швидке розгортання веб-додатків і їх безперервну інтеграцію і доставку.

Крім основних компонентів стека MERN, використовуються також додаткові залежності та інструменти, такі як Auth0 для автентифікації користувачів, React-bootstrap для створення адаптивного інтерфейсу та інші бібліотеки для забезпечення функціональності та безпеки додатку

Переваги

1. Масштабо-спроможність: Веб-додаток "Healthy" має високу масштабованість, що дозволяє легко адаптуватися до зростання кількості користувачів, нових рецептів та додаткових функцій. Це забезпечується використанням стеку технологій MERN (MongoDB, Express.js, React.js, Node.js), який дозволяє динамічно керувати даними та ефективно обробляти великі обсяги

інформації. MongoDB, як документо-орієнтована база даних, забезпечує гнучкість і ефективність у керуванні даними користувачів та рецептів, що дозволяє легко адаптуватися до зростаючих потреб користувачів та обсягів збережених даних.

2. Ефективність: Застосунок оптимізований для швидкої та ефективної взаємодії з користувачами, завдяки використанню сучасних технологій та алгоритмів. Відповідь системи на запити користувачів є миттєвою, що не лише покращує загальне враження від користування, але й забезпечує високу задоволеність та залученість користувачів.

3. Безпека: Надійні заходи безпеки, які впроваджені у нашому застосунку, забезпечують захист персональних та кулінарних даних користувачів. Використання передових методів шифрування та автентифікації допомагає зберегти довіру користувачів і захистити їх інформацію від несанкціонованого доступу.

4. Інновації: Платформа постійно вдосконалюється з врахуванням відгуків користувачів та новітніх технологічних трендів. Ми прагнемо не лише задовольнити поточні потреби наших користувачів, а й обробляти майбутні запити, регулярно впроваджуючи інноваційні функції, що роблять наш застосунок ще більш привабливим та зручним у використанні.

Технологічний стек веб-застосунку "Healthy" вирізняється чотирма основними перевагами, які роблять його унікальним і зручним для користувачів. Масштабованість системи забезпечує безперебійну роботу при зростанні кількості користувачів і розширенні функціональних можливостей. Висока ефективність досягається завдяки швидкій та миттєвій взаємодії, що підвищує задоволеність користувачів. Надійні заходи безпеки гарантують захист персональних даних, що сприяє довірі користувачів до платформи. Постійні

інновації дозволяють нам впроваджувати нові функції, відповідати на зростаючі потреби та підвищувати якість сервісу. Ці переваги разом створюють динамічну, масштабовану та безпечну платформу, яка задає нові стандарти в області кулінарних веб-застосунків.

ВИСНОВКИ

У процесі виконання дипломної роботи на тему "Розробка та створення сайту бібліотеки з рецептами та способами їх приготування" було досягнуто основних цілей проекту, що включали створення безпечної та зручної платформи для пошуку та збереження рецептів. Проект було реалізовано з використанням технологічного стека MERN (MongoDB, Express.js, React.js, Node.js), що забезпечив високу продуктивність та масштабованість додатку.

Основні досягнення проекту:

1. Створення безпечної системи реєстрації та входу: Використання Auth0 та MongoDB забезпечило надійне зберігання даних користувачів та захист від несанкціонованого доступу, що підвищило довіру користувачів до платформи.

2. Розробка розширених дієтичних фільтрів: Функціонал фільтрації за вегетаріанськими, веганськими та середземноморськими дієтами дозволив користувачам швидко знаходити рецепти, що відповідають їхнім харчовим уподобанням, що значно полегшило процес планування їжі.

3. Інтеграція функції збереження рецептів: Користувачі отримали можливість зберігати улюблені рецепти для подальшого використання та управління в своєму профілі, що створює персоналізований досвід використання платформи.

4. Відображення детальної інформації про рецепти: Додаток надає користувачам повну інформацію про інгредієнти, етапи приготування та харчові дані для кожного рецепту, що допомагає точно слідувати рецептам та забезпечує інформованість про харчову цінність страв.

5. Забезпечення адаптивного дизайну: Використання React-bootstrap дозволило створити інтерфейс, що адаптується до різних пристроїв та розмірів екранів. Це забезпечило бездоганний досвід користувачів на всіх платформах, незалежно від того, чи користуються вони додатком на комп'ютері, планшеті чи смартфоні. Адаптивний дизайн гарантує зручність та доступність функцій застосунку для всіх користувачів.

6. Інтеграція функції збереження рецептів: Користувачі отримали можливість зберігати свої улюблені рецепти для подальшого доступу та управління у своєму профілі. Це створює персоналізований досвід використання, дозволяючи користувачам легко повертатися до своїх улюблених рецептів, організовувати їх та ділитися ними з іншими.

7. Відображення детальної інформації про рецепти: Застосунок надає користувачам повну інформацію для приготування страв, включаючи інгредієнти, етапи приготування та харчові дані для кожного рецепту. Це допомагає користувачам точно дотримуватися рецептів, забезпечуючи приготування страв відповідно до заданих стандартів.

Проблеми та виклики:

Під час розробки проекту виникли деякі технічні та організаційні виклики, зокрема пов'язані з оптимізацією продуктивності, інтеграцією зі сторонніми API та забезпеченням безпеки даних. Всі ці проблеми були успішно вирішені завдяки ретельному плануванню, тестуванню та впровадженню відповідних технологічних рішень.

Одним із найбільших викликів для мене було створення фронт-енду. Це завдання вимагало від мене значних зусиль і часу, оскільки я створював всі об'єкти самостійно. Робота з React.js, забезпечення адаптивності інтерфейсу та

впровадження користувацьких функцій потребували ретельного вивчення і застосування найкращих практик фронт-енд розробки. Незважаючи на складнощі, мені вдалося успішно завершити всі необхідні елементи, що стало важливим кроком у реалізації проекту.

Подальший розвиток:

1. Незважаючи на успіхи, проект залишається відкритим для подальшого розвитку та вдосконалення. Серед перспективних напрямків розширення функціональності можна виділити:

2. Налаштування профілю користувача: Вдосконалення для більш глибокого налаштування профілю, включаючи дієтичні обмеження та переваги, заплановано на майбутні оновлення. Це дозволить користувачам більш точно налаштовувати свій профіль відповідно до індивідуальних потреб та вподобань.

3. Інтеграція в соціальні мережі: Ця функція спрямована на надання користувачам можливості ділитися рецептами на платформах соціальних мереж буде інтегрована в наступну версію програми. Це дозволить користувачам ділитися улюбленими рецептами з друзями та сім'єю, збільшуючи взаємодію та залученість.

4. Функція пошуку за ключовими словами: Можливості розширеного пошуку для пошуку рецептів на основі конкретних інгредієнтів або ключових слів не були включені в застосунок, але є пріоритетними для наступних оновлень. Це дозволить користувачам більш точно знаходити рецепти, що відповідають їхнім потребам, полегшуючи процес пошуку.

ДОДАТОК

Персональні висновки

Робота над проектом веб-застосунку "Шукач рецептів – 'Healthy'" продемонструвала мою здатність до розв'язання складних технічних задач, ефективного використання сучасних технологій та вміння працювати в умовах обмежених ресурсів. Отримані результати свідчать про успішне досягнення поставлених цілей і створення корисного продукту, який має значний потенціал для подальшого розвитку.

Якщо б я мав змогу виконати цю роботу наново, я б використав отриманий досвід і дотримувався б наступного плану, щоб забезпечити ще кращий результат, наприклад:

1. Попереднє планування та аналіз:
    - Детальний аналіз вимог: Перед початком розробки важливо ретельно проаналізувати вимоги до проекту. Це включає вивчення потреб цільової аудиторії, визначення основних функцій, які повинні бути реалізовані, та встановлення пріоритетів для цих функцій.
    - Планування етапів проекту: Розробка чіткого плану проекту з розбиттям на етапи. Це допоможе уникнути затримок і забезпечить поступовий прогрес. На кожному етапі слід встановити конкретні цілі та завдання.
2. Вибір технологій:
    - Аналіз технологічного стеку: Вибір відповідного технологічного стеку є критично важливим. Я б знову обрав MERN (MongoDB, Express.js, React.js, Node.js) через його гнучкість і масштабованість, але детальніше вивчив би можливі альтернативи та їхні переваги.

- Попереднє вивчення технологій: Перед початком роботи з новими технологіями варто витратити час на їх детальне вивчення. Це допоможе уникнути проблем у процесі розробки та забезпечить більш ефективне використання інструментів.

3. Розробка фронт-енду:
   - Прототипування інтерфейсу: Перед початком коду варто створити прототип інтерфейсу користувача. Це дозволить отримати зворотний зв'язок від потенційних користувачів і внести необхідні корективи ще до початку розробки.
   - Використання готових компонентів: Замість створення всіх об'єктів самостійно, я б використав готові бібліотеки компонентів, такі як Material-UI або Bootstrap цілком. Це значно зекономить час і забезпечить високу якість дизайну.
   - Тестування на різних пристроях: Регулярне тестування адаптивності інтерфейсу на різних пристроях і розмірах екранів допоможе забезпечити зручність користування для всіх категорій користувачів.
4. Розробка бек-енду:
   - Тестування API: Регулярне тестування API для забезпечення його стабільної роботи. Використання інструментів для автоматичного тестування аби виявляти проблеми на ранніх етапах.
   - Документація: Ретельна документація всіх розроблених API. Це спростить інтеграцію та підтримку додатку в майбутньому.
5. Оптимізація продуктивності:
   - Індексація та оптимізація запитів: Оптимізація запитів до бази даних та використання індексів для швидкого доступу до даних.
   - Кешування: Впровадження механізмів кешування для зменшення навантаження на сервер і прискорення часу відповіді.
   - Моніторинг продуктивності: Використання інструментів для моніторингу продуктивності додатку в реальному часі. Це

дозволить оперативно реагувати на проблеми та підвищувати стабільність роботи системи.

6. Тестування та забезпечення якості:
    - Автоматизація тестування: Впровадження автоматизованих тестів для перевірки функціональності додатку. Включаючи модульне, інтеграційне та функціональне тестування.
    - Залучення користувачів: Проведення тестування з реальними користувачами для отримання зворотного зв'язку та виявлення можливих проблем з використанням.
    - Регулярні перевірки безпеки: Періодичне проведення перевірок безпеки для виявлення та усунення вразливостей в додатку.
7. Управління проектом:
    - Агілеві методології: Використання методологій Agile, таких як Scrum або Kanban, для гнучкого управління проектом. Для швидкого адаптування до змін і забезпечення постійного прогресу.
    - Управління ризиками: Ідентифікація потенційних ризиків на ранніх етапах та розробка стратегій їх пом'якшення.

Враховуючи ці висновки та підходи, я вважаю, що виконання подібного проекту буде більш ефективним та продуктивним. Завдяки чіткішому плануванню, глибшому вивченню технологій та більшій увазі до тестування та оптимізації, можна досягти ще вищої якості кінцевого продукту та задовольнити всі потреби користувачів.

Технічні проблеми та рішення: адаптивний інтерфейс

Проблема, що виникла: Основна технічна проблема, з якою я зіткнувся, полягала в досягненні оптимальної чутливості інтерфейсу. Ця проблема була

особливо помітною на різних пристроях і розмірах екрана, що призвело до неузгодженості взаємодії з користувачем.

Реалізоване рішення: щоб вирішити цю проблему, я використав React-bootstrap, фреймворк, який відповідає за комплексну структуру застосунку, відому своєю швидкістю реагування та адаптивністю. Ця інтеграція дозволила створити більш плавний і адаптивний дизайн, забезпечивши бездоганну адаптацію інтерфейсу програми на різних пристроях.

Управління ризиками

Управління ризиками було критично важливим компонентом проекту. З самого початку потрібно було завчасно виявляти потенційні ризики, оцінювати їхній можливий вплив і впроваджувати стратегії їх пом'якшення. Цей процес був основоположним для успіху проекту, оскільки треба було бути впевненим що проект можливо створити.

- Технічні ризики: виявлені потенційні технічні проблеми, зокрема проблеми інтеграції зі сторонніми API. Провівши ретельний технічний огляд, можна точно оцінити ймовірність і вплив цих ризиків.
- Ризики дедлайну: Регулярні оцінки прогресу та методи гнучкого управління проектами допомогли мені не відставати від плану та за потреби адаптувати його.

Це дало ще один цінний урок, який можна використати для майбутніх проектів; Початок процесу управління ризиками на ранніх стадіях життєвого циклу проекту дозволив виявити потенційні проблеми до того, як вони стануть критичними. Крім того, здатність адаптуватися до мінливих обставин навчила важливості гнучкості в управлінні проектами та необхідності бути готовим. І останнє, але не менш важливе, цілісна оцінка щодо потенційних ризиків і

поточних викликів гарантує дійсну обізнаність і сприяє розумінню можливості виконання проекту або задачі.

Тестування

У цьому розділі наведено огляд тестування та заходів із забезпечення якості, реалізованих протягом розробки веб-програми «Healthy». Застосовані стратегії:

- **Модульне тестування**: проводиться, щоб переконатися, що окремі компоненти програми правильно функціонують ізольовано. Це включало тестування певних функцій і функцій на коректність, надійність і продуктивність.

- **Тестування інтеграції**: зосереджено на оцінці взаємодії між різними частинами програми, включаючи інтеграцію зовнішніх компонентів із серверними службами та API сторонніх розробників. Це забезпечило безперебійний потік даних і функціональність у всій програмі.

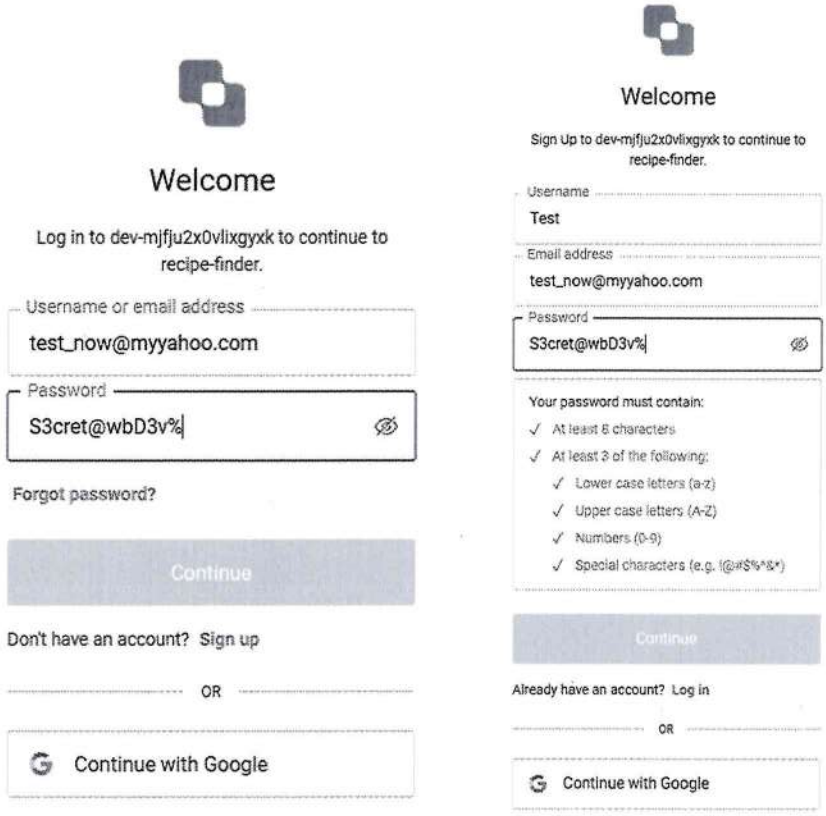
- **Тестування прийнятності користувачами**: залучення реальних користувачів до тестування програми в реальному світі, надання цінних відгуків про інтерфейс користувача, досвід і загальну зручність використання. Це допомогло виявити будь-які проблеми, характерні для користувача, які могли бути неочевидними на ранніх етапах тестування.

- **Тестування продуктивності**: оцінено швидкість реакції, стабільність і масштабованість програми за різних умов навантаження. Це гарантувало, що програма може працювати з багатьма паралельними користувачами без впливу на продуктивність.

Суворе тестування та процеси забезпечення якості, проведені для застосунку, підкреслюють тверде прагнення до якості та досконалості (див. Тестування).

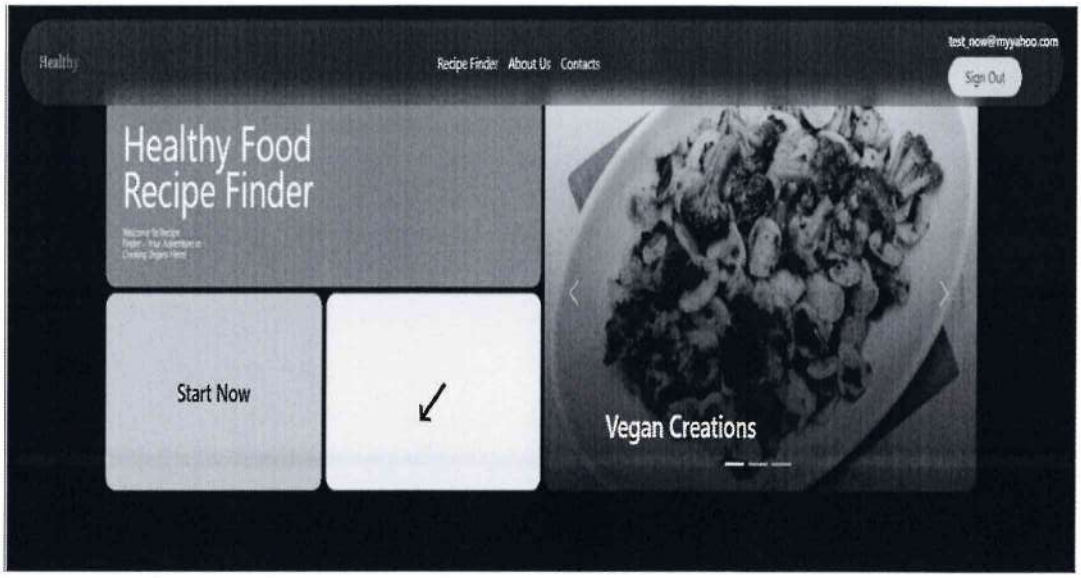
Реєстрація користувача

Тестовий приклад 1 - перевірте, чи можуть користувачі зареєструватися за допомогою дійсних облікових даних (електронна адреса, пароль).



Тест 1 – Результат.

Після реєстрації користувач переходить до свого облікового запису

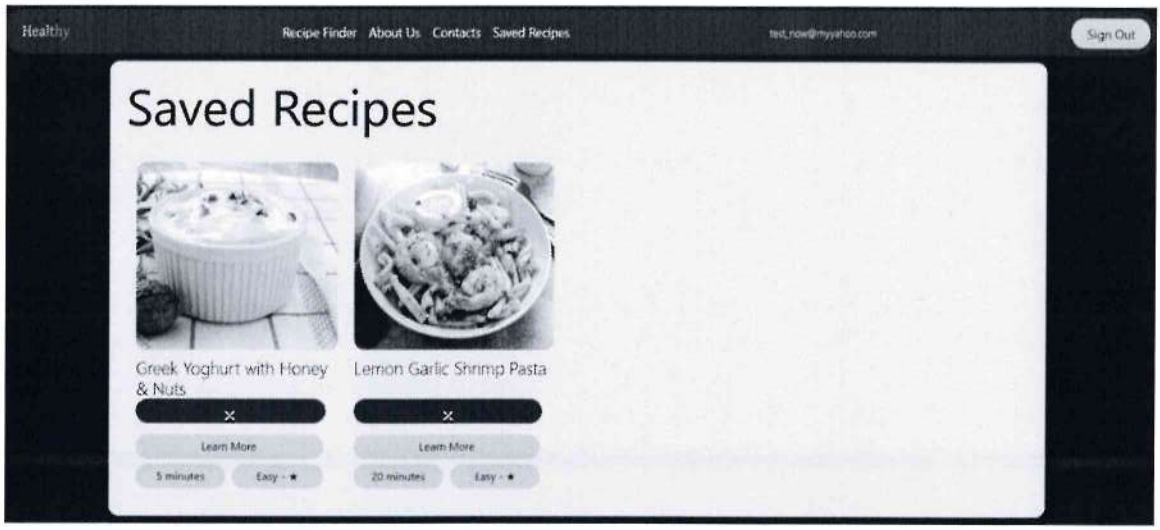


Тестовий приклад 2 - Перевірка, чи не можуть користувачі зареєструватися з недійсними обліковими даними.

Тест 2 – Результат

Тестовий приклад 3 – функція фільтрування рецептів

Тестовий приклад 4 – функція збереження рецепту



Тестовий приклад 5 – зображення індивідуального повного рецепту

Застосункові інтерфейси доступу (APIs)

Тест 1 – Доступ до Головної Сторінки:

Запит

Метод: GET

URL: https://recipe-finder-8bfu.onrender.com/

Очікуване

Код Статусу: 200 OK

Зміст: HTML-відповідь, яка рендерить головну сторінку додатку.

Фактичне

Код Статусу: 200 OK

Час Відповіді: 450 мс

Тести

Код статусу 200: Пройдено.

Час відповіді менше 300 мс: Не пройдено (Час відповіді 450 мс)

Відповідь містить заголовок 'Content-Type' з 'text/html': Пройдено.

Аналіз

Сторінка завантажується успішно з правильним кодом статусу та типом вмісту. Проте, час відповіді перевищує ціль у 300 мс, що вказує на необхідність оптимізації продуктивності. Фактичний час відповіді свідчить, що сервер може ефективно обробляти запит, але існує можливість поліпшення у завантаженні або обслуговуванні ресурсів.

Наступні Кроки

- Оптимізувати конфігурацію сервера: Переглянути конфігурацію веб-сервера на предмет потенційних оптимізацій, які могли б знизити час відповіді.
- Переглянути код додатку: Проаналізувати код, який обслуговує головну сторінку, на предмет неефективності або вузьких місць.
- Оптимізувати фронтенд-ресурси: Переконатися, що всі зображення, скрипти та CSS-файли оптимізовані для веб.

Тест 2 – Доступ до всіх рецептів:

Запит

Метод: GET

URL: https://recipe-finder-server-f153.onrender.com/api/recipes

Очікуване

Код Статусу: 200 OK

Зміст: JSON-відповідь з масивом рецептів.

Фактичне

Код Статусу: 502 Bad Gateway

Час Відповіді: 621 мс

Тести

Код статусу 502: Пройдено
Час відповіді менше 300 мс: Не пройдено (Час відповіді 621 мс)
Відповідь містить заголовок 'Content-Type' з 'application/json': Пройдено
Аналіз

Спроба отримати всі рецепти завершилася помилкою 502 Bad Gateway, що вказує на проблему з сервером або мережевим шляхом між клієнтом та сервером API. Тест на тип вмісту відповіді пройшов, що свідчить, що сервер налаштований реагувати типом вмісту 'application/json' навіть у сценаріях помилок.

Наступні Кроки

- Розслідувати конфігурацію сервера та мережі: Перевірити логи сервера, щоб з'ясувати, чому сервер повернув помилку 502 Bad Gateway.
- Забезпечити стабільність API сервера: Перевірити, що сервер API працює коректно і немає внутрішніх помилок або неправильних налаштувань, що спричиняють помилки 502.
- Переглянути налаштування проксі та шлюзу: Якщо в шляху запиту задіяний зворотній проксі або шлюз, переглянути їх налаштування, щоб переконатися, що вони правильно маршрутизують запити до сервера API.

Тест 3 – Видалення рецепту:

Запит

Метод: DELETE

URL: https://recipe-finder-server-xgd5.onrender.com/api/recipes/

Очікуване

Код Статусу: 400 Bad Request

Зміст: JSON-відповідь, що містить поле з повідомленням про помилку, що пояснює, чому запит був неправильним.

Фактичне

Код Статусу: 400 Bad Request

Час Відповіді: 61,344 мс

Тести

Код статусу 400: Пройдено

Відповідь має необхідні поля: Пройдено

Повідомлення про помилку є непорожнім рядком: Пройдено

Час відповіді менше 500 мс: Не пройдено (Час відповіді 61,344 мс; очікувалося нижче 500 мс)

Аналіз

Тест підтверджує, що кінцева точка правильно ідентифікує неправильний запит, повертаючи код статусу 400 Bad Request разом із повідомленням про помилку, що відповідає критеріям функціональної правильності. Проте, дуже великий час відповіді значно перевищує прийнятний поріг продуктивності 500 мс. Ця надзвичайна затримка у часі відповіді є дуже незвичною для простого запиту DELETE, навіть такого, що призводить до помилки 400, що вказує на значні основні проблеми.

Наступні Кроки

- Розслідувати продуктивність сервера: Необхідно негайно діагностувати причину крайньої затримки.
- Переглянути логіку обробки помилок: Вивчити логіку обробки помилок при отриманні конкретних рецептів за ID.
- Оптимізувати запити до бази даних: Якщо відповідь помилки включає запити до бази даних (наприклад, для перевірки ID рецепту), переконатися, що ці запити оптимізовані і індексовані правильно.

Відгуки користувачів

Відгук від користувачів

- Категорія користувача: більшість є постійними користувачами, що може свідчити про високий рівень утримання користувачів або відповідність їхнім потребам. Однак є також потенційні користувачі, які ще не користувалися додатком, що передбачає можливості для розвитку.
- Частота використання: значна частина користувачів користуватиметься додатком кілька разів на тиждень, що вказує на те, що додаток, швидше за все, вважається корисним інструментом, а не щоденною необхідністю.
- Простота використання: користувачі вважають програму дуже простою у використанні, що є чудовим показником ефективного дизайну інтерфейсу та досвіду користувача.
- Привабливість дизайну: усі відповідачі вважають дизайн візуально привабливим, що може бути вагомим фактором для задоволення користувачів і подальшого використання.
- Навігація: Навігація програмою одностайно оцінюється як дуже проста, вказуючи на добре організований та інтуїтивно зрозумілий інтерфейс.
- Особливості та функціональність: функції «Дієтичний фільтр» і «Збереження страв» вважаються найбільш корисними, що свідчить про те, що користувачі прагнуть персоналізувати свій досвід і піклуються про своє здоров'я.
- Задоволення вмістом: користувачі поділяються між дуже задоволеними та нейтральними щодо різноманітності рецептів, що може вказувати на бажання ширшого вибору або спеціалізованого вмісту.
- Виконання дієтичних уподобань: більшість вважає, що рецепти дуже добре відповідають їхнім дієтичним уподобанням, що підсилює цінність функції дієтичного фільтра.

- Загальна задоволеність: користувачі загалом задоволені програмою, причому більшість із них дуже задоволені.
- Рекомендації: більшість користувачів рекомендуватимуть додаток іншим, що свідчить про позитивне сприйняття та потенційний маркетинг із вуст в уста.
- Додатковий відгук: пропозиції включають:
    - Уніфікованість колірної схеми в усьому додатку.
    - Пропозиція більше працювати над реалізацією дизайну.
    - Спостереження, що показ на мобільному телефоні трохи тісний.

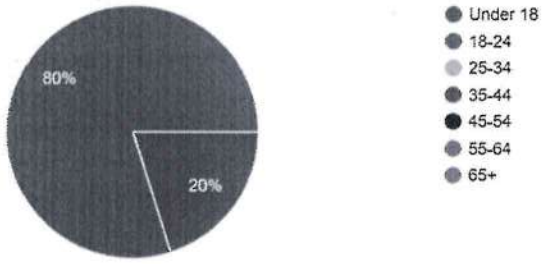
Загалом відгуки є переважно позитивними з високою задоволеністю від простоти використання, навігацією та дизайном. Користувачі високо оцінюють такі параметри персоналізації, як Дієтичний фільтр і можливість зберігати страви. Є місце для вдосконалення з точки зору одноманітності дизайну та потенційного розширення різноманітності рецептів для підвищення задоволеності контентом. Примітка про те, що показ з мобільного пристрою обмежений, може вказувати на необхідність коригування адаптивного дизайну.

Demographic Questions

Copy

Age Group:

5 responses



Question: Please indicate which category best describes you: (Select one)

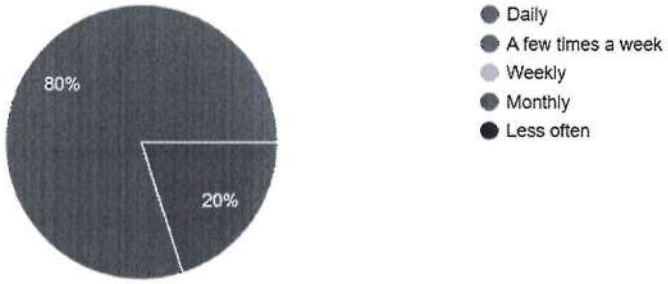
Copy

5 responses

How often would you use the Recipe Finder App?

Copy

5 responses

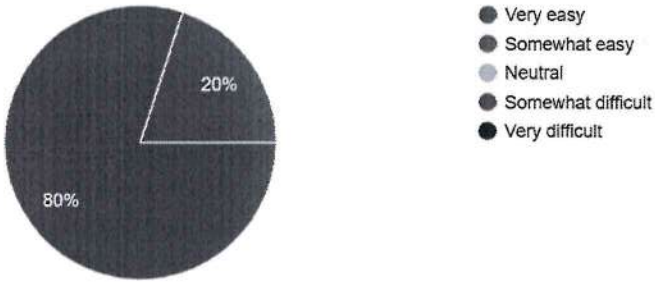


Copy

Usability and Design:

How would you rate the ease of use of the Recipe Finder App?

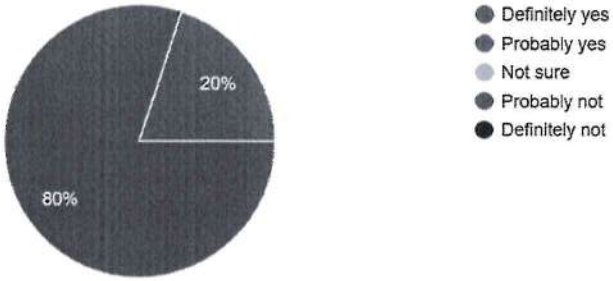
5 responses



Would you recommend the Recipe Finder App to others?

Copy

5 responses



Additional Feedback:

Please share any other comments or suggestions you have to improve the Recipe Finder App

5 responses

make the colours identical throughout the app

I'm absolutely satisfied with this app.

make an app in the future

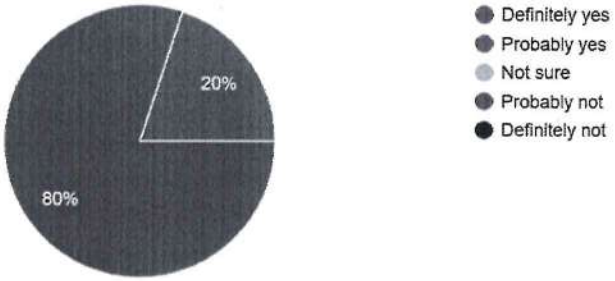
Would work a bit more with design implementation to the final variant and

The layout on mobile is a bit cramped

Would you recommend the Recipe Finder App to others?

Copy

5 responses



Additional Feedback:

Please share any other comments or suggestions you have to improve the Recipe Finder App

5 responses

make the colours identical throughout the app

I'm absolutely satisfied with this app.

make an app in the future

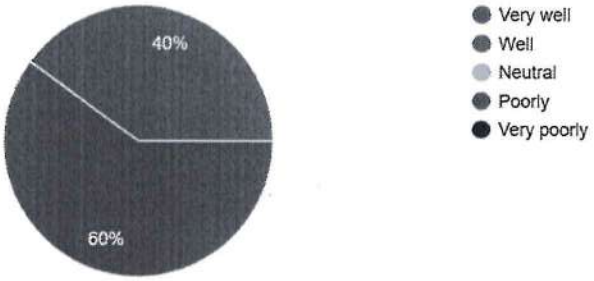
Would work a bit more with design implementation to the final variant and

The layout on mobile is a bit cramped

How well do the recipes meet your dietary preferences?

Copy

5 responses

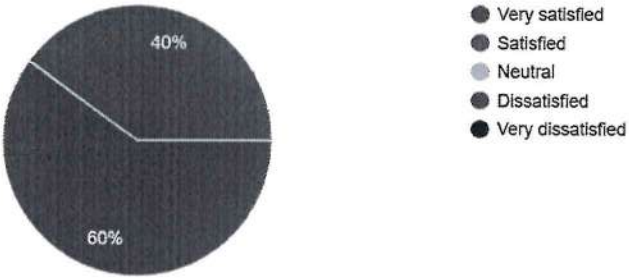


Copy

Overall Satisfaction:

Overall, how satisfied are you with the Recipe Finder App?

5 responses

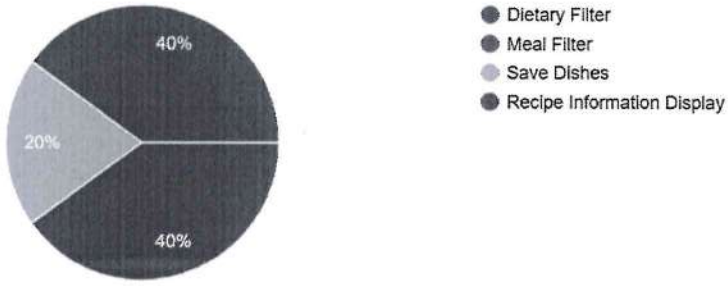


Features and Functionality

Copy

Which feature(s) do you find most useful in the app? (Check all that apply)

5 responses

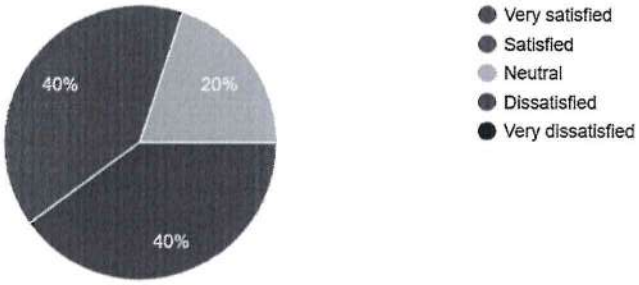


Copy

Content:

How satisfied are you with the variety of recipes available?

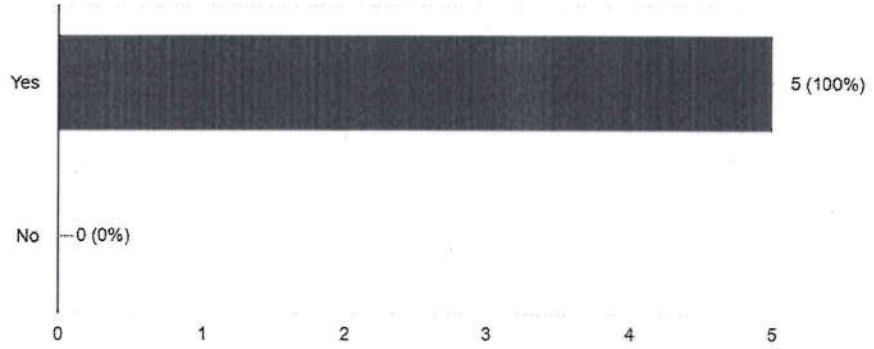
5 responses



Is the app's design (layout, color scheme and fonts) visually appealing?

Copy

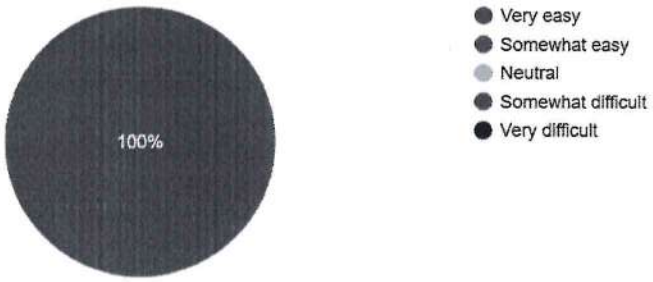
5 responses



How easy is it to navigate through the app to find what you need?

Copy

5 responses



MERN (MongoDB, Express.js, React.js, Node.js)

Внутрішні залежності: CORS, dotenv, express.js, mongoose.

Інтерфейсні залежності: auth0(з'єднувач), bootstrap, http-proxy-middleware, react, react-dom, react-router-dom, web-vitals.

Сторонні залежності: Auth0, Render.com, Netlify, Mongo Atlas.

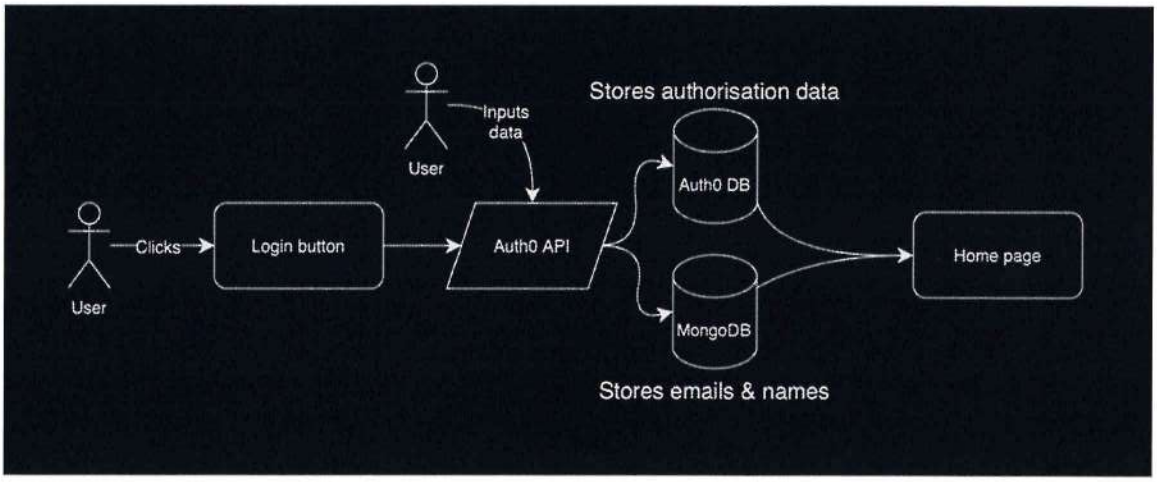
Система аутентифікації

Аля автентифікації, а саме: реєстрація, вхід, вихід із системи та зберігання даних користувача програма використовує систему автентифікації на основі token Auth0 і MongoDB, яка дозволяє входити через обліковий запис Google. Для уточнення, натискання кнопки входу запускає низку функцій, спочатку це передає користувача до форми входу/реєстрації, наданої Auth0, але налаштованої під застосунок. Після цього користувач вводить свої дані у форму (наприклад, ім'я, електронну адресу, пароль або вхід за допомогою облікового запису Google) після успішного входу/реєстрації облікові дані користувачів шифруються та безпечно зберігаються в базі даних Auth0, а адреса електронної пошти та ім'я користувачів зберігаються в MongoDB на майбутнє використовувати. Нарешті користувач перенаправляється на головну сторінку.

```
import React from "react";
import { useAuth0 } from "@auth0/auth0-react";

const LoginButton = () => {
  const { loginWithRedirect, isAuthenticated } = useAuth0();

  return (
    !isAuthenticated && <button className="login-button" onClick={() => loginWithRedirect()}>Sign In</button>
  )
}

export default LoginButton;
```

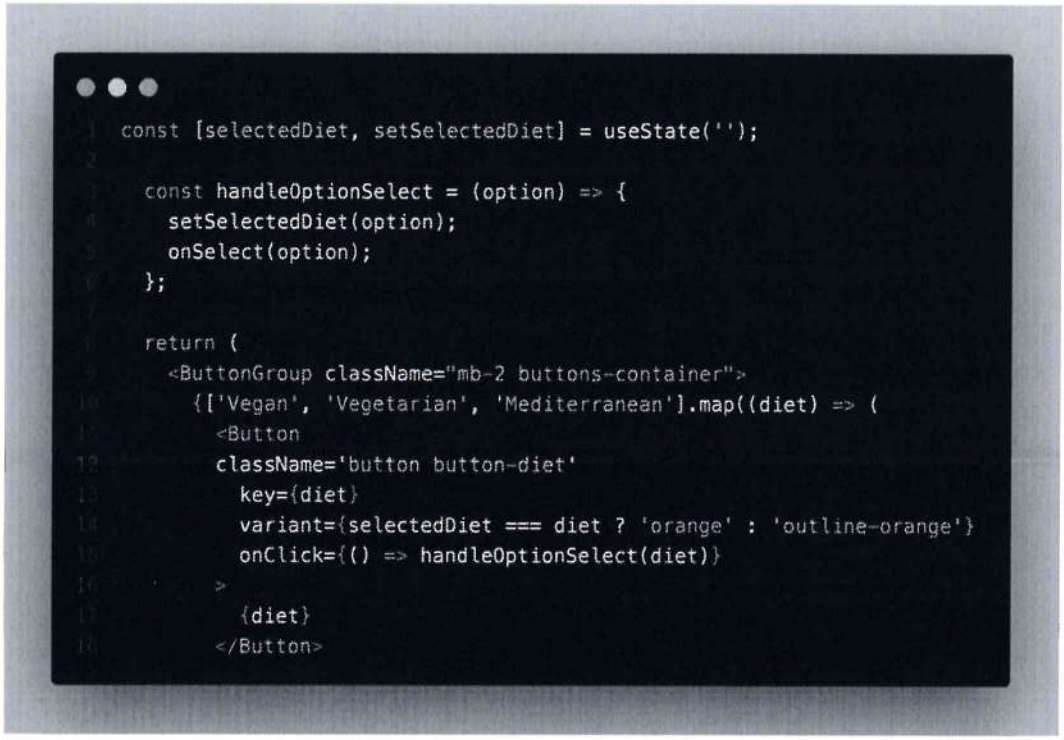
```
import React from "react";
import { useAuth0 } from "@auth0/auth0-react";

const LogoutButton = () => {
  const { logout, isAuthenticated } = useAuth0();

  return (
    isAuthenticated && <button className="logout-button" onClick={() => logout()}>Sign Out</button>
  )
}

export default LogoutButton
```

```jsx
const { user, isAuthenticated } = useAuth0()

useEffect(() => {
  // Call the function to send data to the backend when the component mounts
  if (isAuthenticated) {
    sendDataToBackend(user)
  }
}, [isAuthenticated, user])

const sendDataToBackend = async (userData) => {
  try {
    const response = await fetch(
      "https://recipe-finder-server-xgd5.onrender.com/save-user-data",
      {
        method: "POST",
        headers: {
          "Content-Type": "application/json",
        },
        body: JSON.stringify({
          name: userData.name,
          email: userData.email,
        }),
      }
    )

    if (!response.ok) {
      throw new Error("Failed to save user data")
    }

    const responseData = await response.json()
    console.log("User data saved successfully", responseData)
  } catch (error) {
    console.error("Error saving user data:", error)
  }
}
```



Система фільтрування

Система фільтрації розроблена з використанням 2 груп кнопок, які після вибору обох фільтрів надсилають запит на надсилання до REST API, її тіло містить фільтри категорії дієти та їжі, і в кожному рецепті ці категорії реалізовані, нарешті API збирає необхідні дані з бази даних і повертає відповідь JSON, яка в результаті відображається.



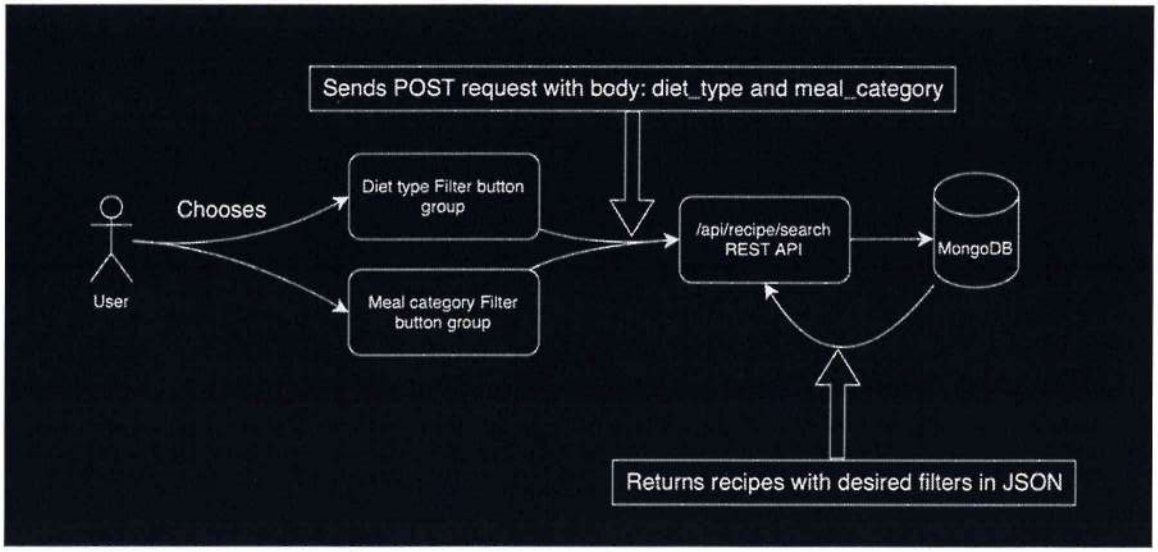
```
const [selectedDiet, setSelectedDiet] = useState('');

  const handleOptionSelect = (option) => {
    setSelectedDiet(option);
    onSelect(option);
  };

  return (
    <ButtonGroup className="mb-2 buttons-container">
      {['Vegan', 'Vegetarian', 'Mediterranean'].map((diet) => (
        <Button
        className='button button-diet'
          key={diet}
          variant={selectedDiet === diet ? 'orange' : 'outline-orange'}
          onClick={() => handleOptionSelect(diet)}
        >
          {diet}
        </Button>
```

```
const MealCategoryFilter = ({ type, onSelect }) => {
  const [selectedMeal, setSelectedMeal] = useState('');

  const handleOptionSelect = (option) => {
    setSelectedMeal(option);
    onSelect(type, option);
  };

  return (
    <ButtonGroup className="mb-2 buttons-container">
      {['Snack', 'Breakfast', 'Lunch', 'Dinner'].map((meal) => (
        <Button
        className='button button-category'
          key={meal}
          variant={selectedMeal === meal ? 'orange' : 'outline-orange'}
          onClick={() => handleOptionSelect(meal)}
        >
          {meal}
        </Button>
      ))}
```

```
const [mainFilter, setMainFilter] = useState(null)
const [subFilter, setSubFilter] = useState(null)
const [recipes, setRecipes] = useState(null)
const [isLoading, setIsLoading] = useState(false)

const handleMainFilterSelect = (filter) => {
  setMainFilter(filter)
}

const handleSubFilterSelect = (type, filter) => {
  setSubFilter(filter)
}

useEffect(() => {
  console.log(`Sub filter updated: ${subFilter}`)
}, [subFilter])

useEffect(() => {
  if (mainFilter !== null && subFilter !== null) {
    setIsLoading(true)
    sendRecipeRequest(mainFilter, subFilter)
  }
}, [mainFilter, subFilter])

const sendRecipeRequest = async (mainFilter, subFilter) => {
  try {
    const response = await fetch("https://recipe-finder-server-xgd5.onrender.com/api/recipes/search", {
      method: "POST",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify({
        dietTypeFilter: mainFilter,
        mealCategoryFilter: subFilter,
      }),
    })

    if (!response.ok) {
      throw new Error("Failed to fetch recipes")
    }
    const responseData = await response.json()
    console.log(responseData)
    setRecipes(responseData)
  } catch (error) {
    console.error("Error fetching recipes:", error)
  } finally {
    setIsLoading(false)
  }
}
```

```
// GET 1 category recipes
router.post("/search", getRecipes)
```

```
const getRecipes = async (req, res) => {
  const dietType = req.body.dietTypeFilter
  const mealCategory = req.body.mealCategoryFilter

  const recipes = await Recipe.find({
    dietType: dietType,
    mealCategory: mealCategory,
  })

  if (!recipes) {
    return res
      .status(404)
      .json({ error: "No recipes find for desired category" })
  }

  return res.status(200).json(recipes)
}
```



Дисплей результатів

Картки для шукача рецептів

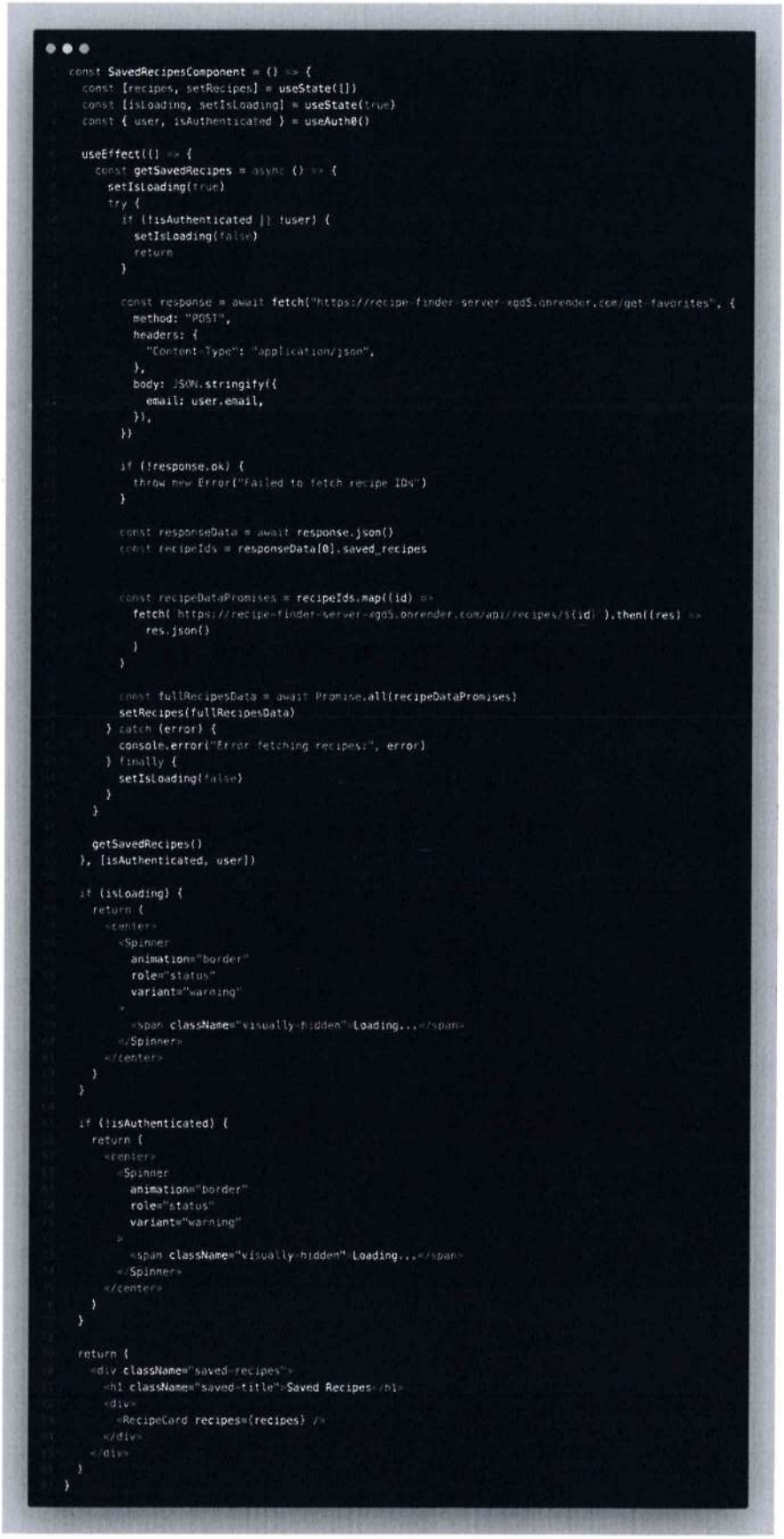
```
<div className="recipes-container">
      {recipes.map((recipe) => (
        <div className="recipe" key={recipe._id}>
          <div className="recipe-content">
            {recipe.image && (
              <img
                className="recipeImage"
                src={`${recipe.image.toString(
                  "base64"
                )}`}
                alt={"afs"}
              />
            )}
            <h2 className="recipe-name">{recipe.name}</h2>
            {path && (
              <p
                className="remove-button"
                onClick={() => removeRecipe(recipe._id)}
              >
                ×
              </p>
            )}
            <div className="info-bullets-container">
              <LearnMoreButton recipeID={recipe._id} />
              <p className="info-bullets">{recipe.preparationTime} minutes</p>
              <p className="info-bullets">
                {recipe.difficulty === 1 && (
                  <>
                    <span>Easy - </span>
                    {[...Array(recipe.difficulty)].map((_, index) => (
                      <span key={index}>★</span>
                    ))}
                  </>
                )}
                {recipe.difficulty === 2 && (
                  <>
                    <span>Medium - </span>
                    {[...Array(recipe.difficulty)].map((_, index) => (
                      <span key={index}>★</span>
                    ))}
                  </>
                )}
                {recipe.difficulty === 3 && (
                  <>
                    <span>Hard - </span>
                    {[...Array(recipe.difficulty)].map((_, index) => (
                      <span key={index}>★</span>
                    ))}
                  </>
                )}
              </p>
            </div>
          </div>
        </div>
```

Повна інформація про рецепт

```
<div className="single-recipe">
      <React.Fragment key={recipe._id}>
        <div className="recipe-container">
          <div className="recipe-left">
            {recipe.image && (
              <img
                className="single-recipe-image"
                src={`${recipe.image.toString(
                  "base64"
                )}`}
                alt={"afs"}
              />
            )}
          </div>
          <div className="recipe-right">
            <h1 className="recipe-title-text">{recipe.name}</h1>
            <div className="info-bullets-container-s">
              <p className="info-bullets">{recipe.preparationTime} minutes</p>
              <p className="info-bullets">
                {recipe.difficulty === 1 && (
                  <>
                    <span>Easy - </span>
                    {[...Array(recipe.difficulty)].map((_, index) => (
                      <span key={index}>★</span>
                    ))}
                  </>
                )}
                {recipe.difficulty === 2 && (
                  <>
                    <span>Medium - </span>
                    {[...Array(recipe.difficulty)].map((_, index) => (
                      <span key={index}>★</span>
                    ))}
                  </>
                )}
                {recipe.difficulty === 3 && (
                  <>
                    <span>Hard - </span>
                    {[...Array(recipe.difficulty)].map((_, index) => (
                      <span key={index}>★</span>
                    ))}
                  </>
                )}
              </p>
            </div>
            <br></br>
            <h2 className="recipe-text-underlined underlined">
              Ingredients :{" "}
            </h2>
            <ul className="main-recipe-text">
              {recipe.ingredients.map((ingredient, index) => (
                <li key={index}>{ingredient}</li>
              ))}
            </ul>
            <h2 className="recipe-text-underlined underlined">
              Instructions :{" "}
            </h2>
            <ul className="main-recipe-text">
              {recipe.preparation.map((preparationStep, index) => (
                <li key={index}>{preparationStep}</li>
              ))}
            </ul>
            <SaveRecipeButton recipeID={recipe._id} />
          </div>
        </div>
      </React.Fragment>
    </div>
```

```jsx
const Recipe = () => {
  const [recipe, setRecipe] = useState(null)
  const [isLoading, setIsLoading] = useState(false)
  const location = useLocation()
  const urlParams = new URLSearchParams(location.search)
  const ridParam = urlParams.get("rID")

  //fetch recipe
  useEffect(() => {
    const sendRecipeRequest = async () => {
      setIsLoading(true)

      try {
        const response = await fetch(
          "https://recipe-finder-server-xgd5.onrender.com/api/recipes/" + ridParam,
          {
            method: "GET",
            headers: {
              "Content-Type": "application/json",
            },
          }
        )

        if (!response.ok) {
          throw new Error("Failed to fetch recipes")
        }
        const responseData = await response.json()
        setRecipe(responseData)
        console.log(responseData)
      } catch (error) {
        console.error("Error fetching recipes:", error)
      } finally {
        setIsLoading(false)
      }
    }

    sendRecipeRequest()
  }, [ridParam])

  if (isLoading) {
    return (
      <center>
        <Spinner
          style={{ marginTop: "50px" }}
          animation="border"
          role="status"
          variant="warning"
        >
          <span className="visually-hidden">Loading...</span>
        </Spinner>
      </center>
    )
  }

  return (
    <div>
      <SingleRecipe recipe={recipe} />
    </div>
  )
```

Збережені рецепти



```
const SavedRecipesComponent = () => {
  const [recipes, setRecipes] = useState([])
  const [isLoading, setIsLoading] = useState(true)
  const { user, isAuthenticated } = useAuth0()

  useEffect(() => {
    const getSavedRecipes = async () => {
      setIsLoading(true)
      try {
        if (!isAuthenticated || !user) {
          setIsLoading(false)
          return
        }

        const response = await fetch("https://recipe-finder-server-xgd5.onrender.com/get-favorites", {
          method: "POST",
          headers: {
            "Content-Type": "application/json",
          },
          body: JSON.stringify({
            email: user.email,
          }),
        })

        if (!response.ok) {
          throw new Error("Failed to fetch recipe IDs")
        }

        const responseData = await response.json()
        const recipeIds = responseData[0].saved_recipes


        const recipeDataPromises = recipeIds.map((id) =>
          fetch( https://recipe-finder-server-xgd5.onrender.com/api/recipes/${id} ).then((res) =>
            res.json()
          )
        )

        const fullRecipesData = await Promise.all(recipeDataPromises)
        setRecipes(fullRecipesData)
      } catch (error) {
        console.error("Error fetching recipes:", error)
      } finally {
        setIsLoading(false)
      }
    }

    getSavedRecipes()
  }, [isAuthenticated, user])

  if (isLoading) {
    return (
      <center>
        <Spinner
          animation="border"
          role="status"
          variant="warning"
        >
          <span className="visually-hidden">Loading...</span>
        </Spinner>
      </center>
    )
  }

  if (!isAuthenticated) {
    return (
      <center>
        <Spinner
          animation="border"
          role="status"
          variant="warning"
        >
          <span className="visually-hidden">Loading...</span>
        </Spinner>
      </center>
    )
  }

  return (
    <div className="saved-recipes">
      <h1 className="saved-title">Saved Recipes</h1>
      <div>
        <RecipeCard recipes={recipes} />
      </div>
    </div>
  )
}
```

Функція зберігання рецепту

```jsx
const SaveRecipeButton = ({ recipeID }) => {
  const { user, isAuthenticated } = useAuth0()
  const [isSaved, setIsSaved] = useState(false)

  const handleClick = () => {
    if (!isSaved) {
      saveRecipe(user.email, recipeID)
    }
  }

  const saveRecipe = async (email, recipe_id) => {
    try {
      const response = await fetch("https://recipe-finder-server-xgd5.onrender.com/add-to-favorites", {
        method: "POST",
        headers: {
          "Content-Type": "application/json",
        },
        body: JSON.stringify({
          email: email,
          recipe: recipe_id,
        }),
      })

      if (!response.ok) {
        throw new Error("Failed to save recipe")
      }

      const responseData = await response.json()
      console.log("Recipe saved successfully", responseData)
      setIsSaved(true) // Update state to indicate recipe is saved
    } catch (error) {
      console.error("Error saving recipe:", error)
    }
  }

  return (
    <div>
      {isAuthenticated && (
        <p onClick={handleClick} className={`save-button ${isSaved ? 'saved' : ''}`}>
          {isSaved ? '✓' : 'Add to favorites'}
        </p>
      )}
    </div>
  )
```

Моделі бази даних

```
const mongoose = require("mongoose")

const Schema = mongoose.Schema

const recipeSchema = new Schema(
  {
    name: {
      type: String,
      required: true,
    },
    dietType: {
      type: String,
      required: true,
    },
    mealCategory: {
      type: String,
      required: true,
    },
    preparationTime: {
      type: Number,
      required: true,
    },
    difficulty: {
      type: Number,
      required: true,
    },
    ingredients: {
      type: Array,
      required: true,
    },
    image:{
      type: String,
      required: true,
    },
    preparation:{
      type: Array,
      required: true,
    },
    reviews: [
          {
        userEmail: {
          type: String,
          required: false,
        },
        comment: {
          type: String,
          required: false,
        },
        rating: {
          type: Number,
          required: false,
        }
      }
    ]
  },
  { timestamps: true }
)

module.exports = mongoose.model("Recipe", recipeSchema)
```