ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «МОБІЛЬНИЙ ДОДАТОК ДЛЯ ПІДБОРУ КУЛІНАРНИХ РЕЦЕПТІВ З ВИКОРИСТАННЯМ ШТУЧНОГО ІНТЕЛЕКТУ»

на здобуття освітнього ступеня бакалавра
зі спеціальності 122 Комп'ютерні науки
освітньо-професійної програми Штучний інтелект

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

_____ Владислав ЛОМАКА

Виконав:
здобувач вищої освіти
група ШІД-41
Владислав ЛОМАКА

Керівник: Тетяна КИСІЛЬ

Рецензент:
науковий ступінь,
вчене звання
_____
Ім'я, ПРІЗВИЩЕ

Київ – 2025

ЗАВДАННЯ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Штучного інтелекту

Ступінь вищої освіти Бакалавр

Спеціальність Комп'ютерні науки

Освітньо-професійна програма Штучний інтелект

ЗАТВЕРДЖУЮ
Завідувач кафедрою Штучного інтелекту

_____ Ольга ЗІНЧЕНКО
«_____» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Ломаці Владиславу Ігоровичу
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Мобільний додаток для підбору кулінарних рецептів з використанням штучного інтелекту

керівник кваліфікаційної роботи Тетяна КИСІЛЬ, старший викладач
(Ім'я, ПРІЗВИЩЕ науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» лютого 2025 р. № 56

2. Строк подання кваліфікаційної роботи «23» травня 2025 р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література, мобільна розробка за UX/UI дизайном, Android/iOS, доступні API, користувацькі сценарії.

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

1. Вибір моделі штучного інтелекту для створення кулінарних рецептів
2. Дослідження методів машинного навчання та глибинного навчання для генерації кулінарних рецептів

3. Розробка архітектури системи: клієнтська частина (мобільний додаток), серверна частина (API, мікросервіси), модуль AI

5. Перелік графічного матеріалу: *презентація*
1. Скріншоти інтерфейсу мобільного додатка
2. Діаграми архітектури системи (UML-діаграми, діаграми потоку даних)
3. Структурні схеми алгоритмів (алгоритм обробки запиту користувача, алгоритм генерації кулінарних рецептів AI-моделлю)

6. Дата видачі завдання «25» лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	25.02-28.02.25	виконано
2	Дослідження технологій, алгоритмів та методів реалізації мобільного додатка з інтеграцією штучного інтелекту	28.02-01.03.25	виконано
3	Аналіз особливостей генерації рецептів за допомогою штучного інтелекту	01.03-5.03.25	виконано
4	Застосування штучного інтелекту для генерації кулінарних рецептів	05.03-10.03.25	виконано
5	Розробка архітектури системи: клієнтська частина (мобільний додаток), серверна частина (API, мікросервіси), база даних, модуль AI	10.03-15.03.25	виконано
6	Проектування користувацького інтерфейсу (UX/UI) мобільного додатку	15.03-20.04.25	виконано
	Розробка тестових сценаріїв для перевірки функціоналу та користувацького досвіду	20.04-28.04.25	виконано
	Аналіз результатів тестування та виявлення недоліків	28.04-10.05.25	виконано
7	Оформлення роботи: вступ, висновки, реферат	10.05-12.05	виконано
8	Розробка демонстраційних матеріалів	17.05-23.05.25	виконано

Здобувач вищої освіти _____ Владислав ЛОМАКА
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Тетяна КИСІЛЬ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 72 стор., 3 табл., 12 рис., 20 джерел.

Мета роботи. Розробка мобільного додатка для підбору кулінарних рецептів з використанням штучного інтелекту.

Об'єкт дослідження. Генерація кулінарних рецептів на основі вказаних користувачем продуктів з використанням штучного інтелекту.

Предмет дослідження. Технології, алгоритми та методи реалізації мобільного додатка з інтеграцією штучного інтелекту, який генерує рецепт на основі вказаних користувачем продуктів.

Короткий зміст роботи. У роботі представлено мобільний додаток, що здатен генерувати кулінарні рецепти, надавати список необхідних продуктів з їхньою кількістю або вагою, покрокову інструкцію з приготування страви. Додаток створено на базі React Native Expo, що забезпечує його кросплатформенність та можливість запуску як на Android, так і на iOS платформах.

КЛЮЧОВІ СЛОВА: AI, КУЛІНАРІЯ, РЕЦЕПТИ, МОБІЛЬНИЙ ДОДАТОК, ШТУЧНИЙ ІНТЕЛЕКТ, ГЕНЕРАЦІЯ РЕЦЕПТІВ, REACT NATIVE, EXPO, КРОСПЛАТФОРМНІСТЬ, МАШИННЕ НАВЧАННЯ.

ABSTRACT

Text of the qualification paper for the Bachelor's degree: 72 pages, 3 table, 12 figures, 20 sources.

Purpose of the work. Development of a mobile application for culinary recipe selection using artificial intelligence.

Object of research. Generation of culinary recipes based on user-specified ingredients using artificial intelligence.

Subject of research. Technologies, algorithms, and methods for implementing a mobile application with artificial intelligence integration that generates a recipe based on user-specified ingredients.

Brief content of the work. The paper presents a mobile application capable of generating culinary recipes, providing a list of necessary ingredients with their quantity or weight, and step-by-step cooking instructions. The application is built using React Native Expo, which ensures its cross-platform compatibility and the ability to run on both Android and iOS platforms.

KEYWORDS: AI, CULINARY, RECIPES, MOBILE APPLICATION, ARTIFICIAL INTELLIGENCE, RECIPE GENERATION, REACT NATIVE, EXPO, CROSS-PLATFORM, MACHINE LEARNING.

ЗМІСТ

ВСТУП

У сучасному світі, де темп життя постійно зростає, питання ефективного планування харчування та доступу до якісних кулінарних рецептів стає дедалі актуальнішим. Люди все частіше шукають зручні та персоналізовані рішення для спрощення процесу приготування їжі, економії часу та забезпечення раціонального харчування. З розвитком мобільних технологій та стрімким прогресом у галузі штучного інтелекту, виникає можливість створювати інноваційні інструменти, здатні задовольнити ці потреби. Існуючі мобільні додатки для підбору рецептів, хоча й пропонують широкий функціонал, часто не використовують повною мірою потенціал персоналізації та інтелектуального аналізу даних, що може суттєво покращити користувацький досвід.

Актуальність теми дипломної роботи полягає у зростаючому попиті на інтелектуальні мобільні рішення, що автоматизують рутинні процеси та надають користувачам високоперсоналізований контент. Інтеграція штучного інтелекту в мобільний додаток для підбору кулінарних рецептів дозволяє не лише ефективно фільтрувати та сортувати рецепти за різними критеріями, а й пропонувати рекомендації на основі індивідуальних вподобань, дієтичних обмежень, наявності інгредієнтів та навіть настрою користувача. Вибір технології React Native для розробки мобільного додатку зумовлений її кросплатформністю, що забезпечує охоплення широкої аудиторії користувачів операційних систем iOS та Android з мінімальними витратами ресурсів.

Метою дипломної роботи є розробка та реалізація мобільного додатку для підбору кулінарних рецептів з використанням штучного інтелекту, що забезпечуватиме персоналізовані рекомендації та зручний інтерфейс для користувачів.

Для досягнення поставленої мети необхідно вирішити наступні *завдання*: провести аналіз предметної області, включаючи огляд існуючих мобільних додатків для підбору рецептів та методів застосування штучного інтелекту в кулінарії; сформулювати функціональні та нефункціональні вимоги до проектованого мобільного додатку, а також обґрунтувати вибір технологічного

стеку React Native; розробити архітектуру мобільного додатку клієнтської частини (React Native); спроектувати базу знань для ефективного зберігання та управління кулінарними рецептами, інгредієнтами та користувацькими даними; розробити та реалізувати інтуїтивно зрозумілий та естетичний користувацький інтерфейс мобільного додатку; реалізувати алгоритми штучного інтелекту для персоналізованого підбору та рекомендації рецептів; провести тестування розробленого мобільного додатку для перевірки його функціональності, стабільності та відповідності вимогам; оцінити ефективність розробленої системи та визначити перспективи її подальшого розвитку.

Об'єктом дослідження є процес підбору кулінарних рецептів, *предметом дослідження* – методи та засоби інтеграції штучного інтелекту у мобільні додатки для автоматизації та персоналізації цього процесу.

Структура дипломної роботи складається зі вступу, трьох основних розділів, висновків, списку використаних джерел та додатків. Кожен розділ детально висвітлює відповідні аспекти проектування, розробки та тестування мобільного додатку генерації кулінарних рецептів.

1. ТЕОРЕТИЧНІ ОСНОВИ МОБІЛЬНИХ ДОДАТКІВ ТА ШТУЧНОГО ІНТЕЛЕКТУ В КУЛІНАРІЇ

1.1 Огляд сучасних мобільних платформ та фреймворків

Сучасний ринок мобільних додатків характеризується надзвичайною динамічністю та конкуренцією. Розробка ефективного та зручного мобільного рішення для підбору кулінарних рецептів вимагає ретельного аналізу доступних технологій. Цей підрозділ присвячений огляду сучасних мобільних платформ, особливостям кросплатформної розробки та детальному аналізу фреймворку React Native, що є ключовою технологією для реалізації даного дипломного проекту.

Розробка мобільних додатків може здійснюватися двома основними підходами: нативна та кросплатформна розробка. Нативна розробка передбачає створення окремих додатків для кожної мобільної платформи (iOS та Android) з використанням їхніх нативних мов програмування та SDK, таких як Swift/Objective-C для iOS та Kotlin/Java для Android. Перевагами такого підходу є висока продуктивність, повний доступ до всіх функцій пристрою, оптимальна інтеграція з екосистемою платформи, найкращий користувацький досвід та високий рівень безпеки. Однак, нативна розробка має і недоліки, такі як висока вартість через необхідність підтримки двох окремих кодових баз, збільшення часу розробки та потреба у спеціалізованих навичках розробників для кожної платформи.

На противагу цьому, кросплатформна розробка дозволяє створити один додаток, який працює на кількох мобільних операційних системах (iOS, Android) з єдиною кодовою базою, використовуючи спеціалізовані фреймворки. Це забезпечує економію часу та ресурсів, оскільки одна кодова база значно прискорює розробку та знижує витрати на підтримку. Кросплатформний підхід також дозволяє охопити ширшу аудиторію, роблячи додаток доступним одразу на обох популярних платформах, і спрощує підтримку завдяки внесенню оновлень та виправлень в одну кодову базу. Крім того, багато кросплатформних фреймворків використовують веб-технології, такі як JavaScript, HTML та CSS, що робить їх доступними для веб-розробників.

Таблиця 1.1 Порівняння підходів до розробки мобільних додатків

Підхід до розробки	***Переваги***	***Недоліки***
Нативна	* Висока продуктивність	* Висока вартість розробки (потреба підтримки двох окремих кодових баз)
	* Повний доступ до всіх функцій пристрою	* Збільшення часу розробки
	* Оптимальна інтеграція з екосистемою платформи	* Потреба у спеціалізованих навичках розробників для кожної платформи
	* Найкращий користувацький досвід	
	* Високий рівень безпеки	
Кросплатформна	* Економія часу та ресурсів (одна кодова база)	* Можливі обмеження у доступі до нативних функцій (ускладнений доступ до специфічних API пристрою)
	* Широке охоплення аудиторії (доступність одразу на iOS та Android)	* Продуктивність (у деяких випадках може бути дещо нижчою, ніж у нативних додатків, особливо для ресурсомістких застосунків або складної графіки)
	* Зручність підтримки (оновлення та виправлення вносяться в одну кодову базу)	* Залежність від фреймворку (обмеженість можливостей або проблеми з підтримкою фреймворку)
	* Використання веб-технологій (доступно для веб-розробників, знання JavaScript, HTML, CSS)	* Нативний вигляд та відчуття (іноді кросплатформні додатки можуть виглядати або почуватися не повністю нативними)

Проте, кросплатформна розробка також має свої недоліки, включаючи можливі обмеження у доступі до нативних функцій пристрою, потенційно нижчу продуктивність у порівнянні з нативними додатками (особливо для ресурсомістких застосунків або складної графіки) та залежність від самого фреймворку. Іноді кросплатформні додатки можуть виглядати або почуватися не повністю нативними.

На ринку вже існують деякі проєкти, які частково вирішують описану проблему, однак більшість з них або не використовують штучний інтелект повною мірою, або мають обмежену функціональність.

Платформа **SuperCook** дозволяє вводити наявні інгредієнти та отримувати список можливих рецептів. Однак система не генерує унікальний рецепт, а просто фільтрує наявну базу даних. Подібний підхід застосовується і у **MyFridgeFood** — цей сайт аналізує вказані продукти, але не пропонує нічого нового, лише добірки з наперед заданих рецептів.

Yummly є потужною платформою з елементами персоналізації, але вона більше схиляється до рекомендацій на основі переваг користувача, не обов'язково наявних інгредієнтів. Крім того, її алгоритми орієнтовані на вподобання, а не на адаптацію рецептів до обмеженого набору продуктів.

ChatGPT, як представник широкого класу генеративних моделей, може створювати рецепти за описом продуктів, однак без належної адаптації або інтеграції в зручний користувацький інтерфейс, його використання не є інтуїтивним для пересічного користувача.

Наведемо порівняльний аналіз реалізованих мобільні додатки по кулінарії, які спрощують процеси автоматизації у приготуванні їжі та харчуванням, значно підвищуючи та збагачуючи кулінарний досвід користувачів (табл. 1.2).

Таблиця 1.2 Переваги та недоліки кулінарних додатків

Додаток	***Переваги***	***Недоліки***
SuperCook	Дозволяє працювати із залишками інгредієнтів.	Не підтримує генерацію рецептів (лише добірка), не має персоналізації, не використовує ШІ/МН.
MyFridgeFood	Має функціонал для роботи із залишками інгредієнтів.	Не підтримує генерацію рецептів, не персоналізує рекомендації, не використовує ШІ/МН.
Yummly	Пропонує персоналізацію рецептів.	Не має можливості генерації рецептів, не підтримує роботу із залишками, використання ШІ/МН у ньому "Обмежено".
Allrecipes	Таблиця не вказує конкретних переваг за представленими критеріями.	Не підтримує генерацію рецептів, персоналізацію, роботу із залишками, і не використовує ШІ/МН.
MyApp	Підтримує можливість генерації рецептів, роботу із залишками, а також використовує ШІ/МН.	Можливості персоналізації позначені як "Обмежено".

На основі даного огляду видно, що більшість існуючих рішень мають значні недоліки у сфері інтелектуальних функцій, а саме:

– *Відсутність генерації рецептів***.** Переважна більшість додатків (SuperCook, MyFridgeFood, Yummly, Allrecipes) не можуть генерувати нові рецепти, пропонуючи лише добірку з існуючої бази. Це обмежує креативність і гнучкість для користувача.

– *Обмежена або відсутня персоналізація*. Тільки Yummly (рис. 1.1) пропонує персоналізацію, але навіть там її використання є "Обмеженим". Це означає, що користувачі не отримують рекомендації, адаптовані до їхніх унікальних вподобань та історії.

– *Недостатнє використання AI/ML*. Більшість додатків не використовують або використовують ШІ/МН лише обмежено, що знижує їхню здатність до розумного пошуку, рекомендацій та адаптації.

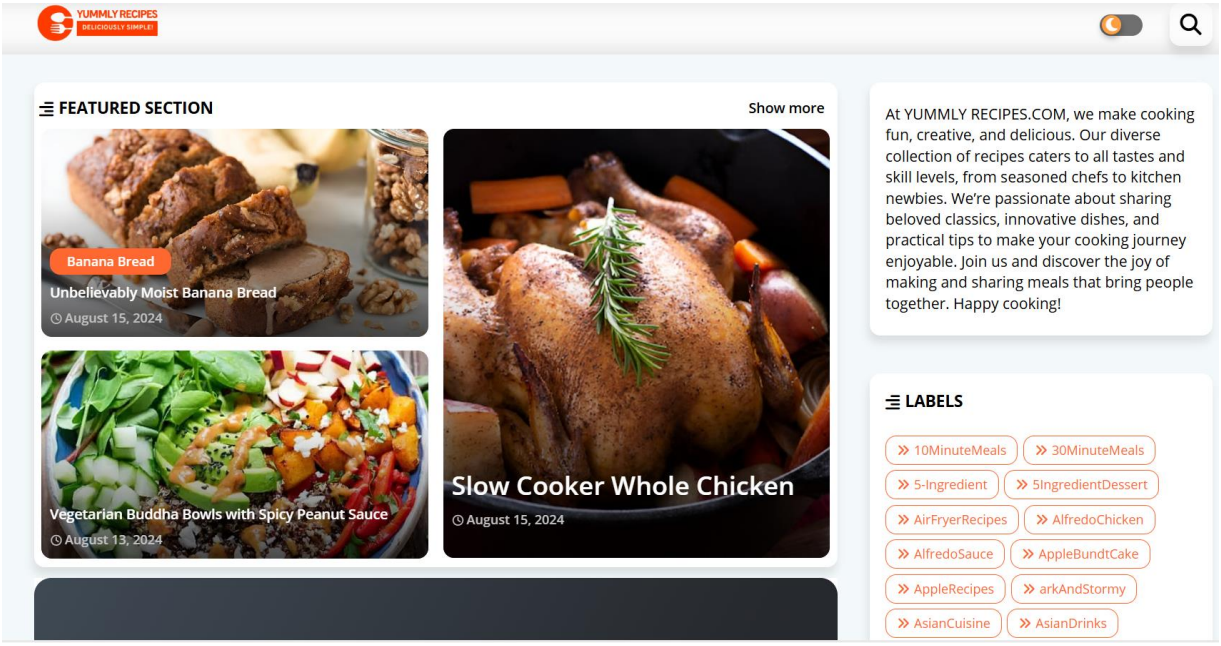


Рисунок 1.1 Кулінарна платформа Yummly з персоналізацією даних

Таким чином, існує потреба у спеціалізованій платформі, що поєднує генеративні можливості ШІ та зручність практичного застосування. Це дозволить вивести кулінарну допомогу на новий рівень. Як висновок, повноцінного інструменту, який би дозволяв саме створювати нові рецепти з врахуванням обраних продуктів, у відкритому доступі наразі немає. Більшість рішень базуються

на традиційному пошуку в базі даних. Натомість, використання генеративних моделей відкриває принципово нову парадигму — створення, а не підбір.

В межах даного дослідження пропонується створення інтелектуальної системи, яка генерує рецепти з обраних інгредієнтів на основі генеративної моделі. Такий підхід вирішує низку нагальних проблем:

- *Обмеженість часу і ресурсів* — користувач не витрачає час на пошук рецепту, що підходить під наявні продукти.
- *Індивідуальні вподобання* — можливість фільтрувати рецепти за дієтою, алергенами, калорійністю.
- *Боротьба з харчовими відходами* — використання залишків продуктів замість викидання.
- *Кулінарна освіта* — користувачі отримують нові ідеї, дізнаються про нові способи приготування звичних продуктів.

Таким чином, аналіз предметної області показав, що існує суттєва потреба у новому підході до створення кулінарних рецептів. Традиційні платформи не відповідають сучасним вимогам адаптивності та персоналізації. Використання штучного інтелекту для генерації рецептів з обраних продуктів — це логічне продовження цифрової трансформації у кулінарії. Такий підхід не лише підвищує ефективність побутових процесів, а й створює умови для раціонального споживання, зменшення відходів та розвитку кулінарної культури.

Для проекту, спрямованого на розробку мобільного додатку для підбору кулінарних рецептів, кросплатформний підхід є оптимальним. Він дозволяє швидко охопити дві основні мобільні платформи, мінімізувати витрати та використовувати існуючі знання веб-розробки, що є особливо важливим для стартапів або проектів з обмеженими ресурсами.

1.2 Аналіз React Native як технології для розробки мобільних додатків

React Native, відкритий кросплатформний фреймворк, розроблений Facebook, дає змогу створювати мобільні додатки за допомогою JavaScript та React.

Відмінною його рисою є компіляція коду в нативні компоненти інтерфейсу користувача, на відміну від гібридних додатків, що працюють у вбудованому веб-переглядачі (WebView).

Принцип роботи React Native ґрунтується на використанні "моста" (Bridge), який забезпечує зв'язок між JavaScript-кодом додатка та нативними модулями операційної системи. Це дозволяє розробникам писати код на JavaScript, а потім відображати його як справжні нативні компоненти інтерфейсу, наприклад, <View> трансформується у UIView в iOS або android.view.View в Android.

Основною перевагою React Native є його кросплатформність з єдиною кодовою базою, що значно заощаджує час та кошти на розробку. Завдяки використанню нативних компонентів інтерфейсу, додатки, розроблені на цьому фреймворку, виглядають і відчуваються як нативні, забезпечуючи високу продуктивність, що є критично важливим для плавного скролінгу списків рецептів або відображення зображень. Використання JavaScript та React дає змогу веб-розробникам швидко адаптуватися до мобільної розробки та ефективно використовувати потужну екосистему React з її компонентним підходом. Функції "Hot Reloading" та "Fast Refresh" суттєво прискорюють процес розробки, дозволяючи миттєво бачити зміни в коді без повної перекомпіляції. Крім того, велика та активна спільнота розробників React Native забезпечує доступ до численних бібліотек, компонентів та оперативної підтримки. Фреймворк також надає можливість інтеграції нативного коду, що дозволяє розробникам писати власні модулі на Java/Kotlin для Android або Swift/Objective-C для iOS у випадках, коли потрібен доступ до специфічних функцій пристрою.

Однак React Native має і певні недоліки. Розробникам з нативним досвідом може знадобитися час для адаптації до парадигми React, що створює певну криву навчання. Додатки, розроблені на React Native, можуть мати дещо більший розмір бінарного файлу порівняно з повністю нативними. Швидкі темпи оновлення фреймворку вимагають регулярної підтримки та адаптації проекту. Також, висока кількість взаємодій JavaScript з нативним кодом через "міст" може іноді негативно впливати на продуктивність.

Зважаючи на вимоги до розробки мобільного додатку для підбору кулінарних рецептів, де ключовими факторами є швидкість розробки, підтримка обох платформ та висока продуктивність інтерфейсу, React Native є оптимальним вибором.

Окрім React Native, у світі кросплатформної мобільної розробки існують інші популярні фреймворки, кожен з яких має свої відмінні риси (табл. 1.3).

Таблиця 1.3 Порівняння кросплатформних мобільних фрейворків

Критерій	React Native	Flutter	Xamarin
Мова програмування	JavaScript / TypeScript	Dart	C#
Принцип рендерингу UI	Використовує нативні UI-компоненти	Малює власний UI за допомогою Skia	Компілює в нативні додатки (Xamarin.Forms/Native)
Продуктивність	Висока, близька до нативної	Дуже висока, часто перевершує RN	Добра, але може бути нижчою для складних UI
Крива навчання	Легка для веб-розробників (React)	Вимагає вивчення Dart	Легка для .NET розробників
Розмір додатку	Середній	Більший (через власні рушії)	Середній
Доступ до нативних API	Через "міст" або нативні модулі	Через "платформні канали"	Повний доступ
Спільнота / Екосистема	Дуже велика, зріла, багато бібліотек	Швидко зростаюча, активна, багато віджетів	Середня
Використання у проекті	Оптимальний для даного проекту (баланс швидкості, продуктивності, кросплатформності та веб-технологій)	Добра альтернатива, якщо є досвід Dart/бажання контролю UI	Можливий, якщо є сильний .NET бекграунд

Одним з таких є **Flutter**, розроблений Google, який використовує мову програмування Dart. Його принцип роботи відрізняється від React Native тим, що

Flutter не використовує нативні компоненти інтерфейсу безпосередньо. Замість цього, він малює власний інтерфейс за допомогою потужного графічного рушія Skia. Це дозволяє досягти абсолютно ідентичного вигляду та відчуття (UI/UX) на всіх мобільних платформах, а також забезпечує дуже високу продуктивність, яка є порівнянною з нативною. Серед переваг Flutter слід виділити швидку розробку та багатий набір готових віджетів. Однак, до його недоліків належить необхідність вивчення нової мови програмування (Dart), більший розмір бінарного файлу готового додатка порівняно з React Native (що пояснюється вбудованим рендерингом інтерфейсу), а також менша зрілість екосистеми порівняно з широкою спільнотою React Native та веб-розробки.

Іншим значним кросплатформним фреймворком є **Xamarin** від Microsoft, що базується на мові програмування C#. Xamarin дозволяє розробникам писати код на C# та компілювати його у нативні додатки. Він пропонує два основні підходи: Xamarin.Forms, що дозволяє створювати єдиний інтерфейс користувача для всіх платформ, та Xamarin.Native, який надає можливість використовувати нативні UI-компоненти. Головними перевагами Xamarin є можливість використання C# (що робить його привабливим для розробників з досвідом роботи у .NET екосистемі), повний доступ до всіх нативних API пристрою та тісна інтеграція з екосистемою Microsoft. Водночас, Xamarin має і недоліки. Це, зокрема, складніша крива навчання для нових розробників, потенційно менша продуктивність (особливо для Xamarin.Forms) порівняно з чисто нативною розробкою та Flutter, а також менша спільнота та екосистема порівняно з такими гігантами, як React Native та Flutter.

З огляду на необхідність швидкої розробки, широкого охоплення користувачів iOS та Android, використання веб-технологій та прагнення до нативної продуктивності, React Native є найбільш оптимальним вибором для розробки мобільного додатку для підбору кулінарних рецептів.

1.3 Застосування штучного інтелекту в кулінарії

Застосування штучного інтелекту (ШІ) у сфері кулінарії відкриває нові можливості для персоналізації, автоматизації та оптимізації процесу приготування

їжі. Мобільний додаток для підбору кулінарних рецептів, який використовує ШІ, може значно покращити користувацький досвід, пропонуючи не просто пошук за ключовими словами, а й інтелектуальні рекомендації, адаптацію рецептів та взаємодію на основі природної мови. Цей підрозділ детально розглядає ключові задачі ШІ в кулінарній галузі, основні методи машинного навчання для підбору рецептів та роль нейронних мереж у обробці та генерації кулінарних даних.

1.3.1 Класифікація задач штучного інтелекту, що застосовуються в кулінарії

В кулінарії ШІ може вирішувати низку взаємопов'язаних задач, які можна класифікувати за такими основними напрямками:

1. *Рекомендаційні системи (Recommendation Systems)*. Один з найпоширеніших застосувань ШІ в кулінарних додатках. Головна мета таких систем — пропонувати користувачеві рецепти, які, ймовірно, йому сподобаються, виходячи з його попередніх вподобань, історії переглядів, оцінок, а також з урахуванням характеристик самих рецептів. У контексті мобільного додатку, це означає, що система може рекомендувати рецепти, які відповідають дієтичним обмеженням користувача (веган, без глютену), його улюбленим кухням (італійська, українська), або навіть пропонувати, що приготувати з наявних у холодильнику інгредієнтів. Як приклад, мобільному додатку користувач переглянув кілька рецептів з куркою та овочами. Система ШІ може запропонувати схожі рецепти або страви, які часто готуються разом з куркою, наприклад, "Куряче карі з кокосовим молоком" або "Овочевий салат з курячим філе гриль".
2. *Обробка природної мови (Natural Language Processing - NLP).* NLP є критично важливим для розуміння та взаємодії з кулінарними даними. Це включає:
   - *Аналіз рецептів***.** Вилучення ключової інформації з текстових описів рецептів (назви інгредієнтів, кількості, одиниці виміру, кроки приготування, час). Це дозволяє нормалізувати дані та зробити їх машинозчитуваними.
   - *Розуміння користувацьких запитів***.** Перетворення природної мови користувача ("Я хочу приготувати щось швидке без м'яса", "Дай рецепт борщу без картоплі") у структуровані запити, які може обробити система.

– *Генерація тексту***.** Автоматичне формування описів страв, адаптація рецептів під певні умови (наприклад, "як приготувати цей суп у мультиварці"), або ж створення діалогових відповідей для чат-бота всередині додатку. У мобільному додатку користувач вводить у пошук "рецепти з лососем для дієти". NLP-модуль розпізнає ключові слова, дієтичні обмеження та направляє запит до рекомендаційної системи.

3. *Комп'ютерний зір (Computer Vision - CV).* CV дозволяє ШІ "бачити" та інтерпретувати зображення. У кулінарії це може бути використано для:

– *Розпізнавання інгредієнтів***:** Визначення інгредієнтів на фотографії (наприклад, користувач фотографує вміст холодильника, а додаток визначає наявні продукти).

– *Аналіз зображень страв***:** Оцінка візуальної привабливості страви, категоризація за типом (десерт, основна страва), або навіть визначення основних інгредієнтів за зовнішнім виглядом.

– *Контроль якості/готовності***:** У більш складних системах (наприклад, з інтеграцією з "розумною" технікою) — оцінка готовності страви за її зовнішнім виглядом. **У мобільному додатку к**ористувач робить фото своєї їжі після приготування, і додаток може запропонувати схожі рецепти або дати оцінку презентації страви

1.3.2 Огляд методів машинного навчання для підбору рецептів

Для реалізації рекомендаційних систем та інтелектуального підбору рецептів у кулінарних додатках використовуються різні методи машинного навчання.

Одним із таких методів є **фільтрація на основі вмісту (Content-Based Filtering)**. Її принцип полягає в рекомендації елементів, які схожі на ті, що користувачеві сподобалися в минулому. Система створює "профіль користувача" на основі його попередніх взаємодій (оцінки, перегляди, збережені рецепти) та "профіль елементів", що містить ознаки рецептів, такі як інгредієнти, кухня, час приготування та калорійність. Потім знаходяться рецепти, ознаки яких максимально збігаються з профілем користувача. Перевагами цього підходу є відсутність потреби в даних про інших користувачів, що робить його ефективним

для "холодного старту" нових користувачів, а також здатність рекомендувати унікальні рецепти. Однак, фільтрація на основі вмісту обмежена у здатності до "відкриття" нових смаків, оскільки рекомендує лише те, що схоже на вже відоме, і може призвести до надмірної спеціалізації, пропонуючи одні й ті ж типи рецептів. У кулінарії цей метод застосовується для підбору рецептів за інгредієнтами, категоріями (сніданок, обід) або дієтичними вимогами (веганський, безлактозний).

Інший метод – *колаборативна фільтрація (Collaborative Filtering)***.** Він базується на припущенні, що якщо користувачі А і Б мають схожі вподобання щодо певних рецептів, то рецепти, які сподобалися користувачеві А, але ще не бачив користувач Б, можуть сподобатися й користувачеві Б. Цей метод поділяється на два основні підтипи: user-based, який знаходить користувачів, схожих на поточного, та рекомендує рецепти, які їм сподобалися, і item-based, що знаходить рецепти, схожі на ті, що сподобалися користувачеві, на основі того, як інші користувачі взаємодіяли з цими рецептами. Перевагою колаборативної фільтрації є її здатність до "відкриття", оскільки вона може рекомендувати те, що користувач не очікував, але що сподобалося іншим "схожим" користувачам, а також відсутність потреби в глибокому аналізі вмісту рецептів. Водночас, вона страждає від проблеми "холодного старту" для нових користувачів або нових рецептів, оскільки потребує даних про взаємодії, має проблеми масштабованості на дуже великих датасетах та стикається з "розрідженістю даних", оскільки не всі користувачі взаємодіють з усіма рецептами. У кулінарії цей метод застосовується для рекомендацій типу "Інші користувачі, яким сподобався цей рецепт, також сподобалися..." або "Популярні рецепти серед користувачів з вашими вподобаннями".

Для подолання недоліків та максимізації переваг кожного окремого методу часто використовуються *гібридні підходи (Hybrid Approaches)***.** Вони поєднують фільтрацію на основі вмісту та колаборативну фільтрацію. Прикладами таких підходів є комбінування профілю користувача з оцінками інших користувачів або використання ознак рецептів для покращення колаборативної фільтрації. У кулінарії гібридні підходи дозволяють створювати більш точні та різноманітні

рекомендації, які враховують як зміст рецепту, так і соціальний аспект, тобто вподобання спільноти.

1.2.3 Застосування нейронних мереж для обробки та генерації кулінарних даних

Нейронні мережі є потужним інструментом для ефективної роботи з кулінарними даними, особливо коли завдання стосуються складних залежностей та генерації нового контенту.

Для *обробки кулінарних даних*, що включає їх аналіз та вилучення ознак, широко використовуються векторні представлення (ембедінги). Нейронні мережі, такі як Word2Vec, FastText або BERT, можуть створювати ці ембедінги для різноманітних кулінарних об'єктів — інгредієнтів, страв, кухонь чи навіть цілих рецептів. Ці вектори відображають семантичну схожість між об'єктами, наприклад, слова "яблуко" і "груша" матимуть близькі векторні представлення, що дозволяє рекомендаційним системам ефективніше працювати з поняттям "подібності".

У сфері *обробки природної мови (NLP)* нейронні мережі також знаходять широке застосування. Рекурентні нейронні мережі (RNN), включаючи їхні вдосконалені варіанти LSTM та GRU, використовуються для аналізу послідовностей тексту в рецептах, таких як кроки приготування, а також для розпізнавання іменованих сутностей, зокрема інгредієнтів та одиниць виміру. Сучасніші архітектури, як-от Трансформери (BERT, GPT, T5), застосовуються для глибшого розуміння тексту рецептів, вилучення складних семантичних зв'язків між інгредієнтами та класифікації рецептів за стилем або кухнею.

У галузі *комп'ютерного зору*, згорткові нейронні мережі (CNN) відіграють ключову роль. Їх застосовують для класифікації зображень страв, що дозволяє визначити тип страви за її фотографією (наприклад, суп, салат або десерт). Також CNN використовуються для виявлення об'єктів, що дозволяє розпізнавати та локалізувати інгредієнти на фотографії, наприклад, у холодильнику, а також для аналізу стилю, оцінюючи "апетитність" зображення або його відповідність певному кулінарному стилю.

Окрім аналізу, нейронні мережі є надзвичайно ефективними для *генерації кулінарних даних*. Великі мовні моделі (LLMs), такі як GPT-3/GPT-4, навчені на величезних обсягах текстових даних, включаючи рецепти, можуть генерувати абсолютно нові рецепти на основі заданих користувачем критеріїв, таких як список інгредієнтів, бажана кухня або дієтичні обмеження. Це може включати створення назви, детального списку інгредієнтів та покрокової інструкції. Нейронні мережі також використовуються для адаптації та модифікації існуючих рецептів, наприклад, для автоматичної зміни порцій, адаптації під конкретний кухонний прилад (мультиварку) або заміни інгредієнтів на альтернативні. Вони дозволяють автоматично генерувати привабливі та інформативні описи страв. Крім того, генеративні можливості нейронних мереж використовуються для побудови діалогових систем та чат-ботів, які можуть вести розмову з користувачем, відповідати на запитання про рецепти, інгредієнти або техніки приготування.

У мобільному додатку, розробленому на React Native, ці потужні можливості штучного інтелекту можуть бути реалізовані шляхом інтеграції із зовнішніми API, такими як OpenAI API для генерації тексту, або через моделі машинного навчання, розгорнуті на власному бекенді. Такий підхід дозволить створити повноцінну та високоінтелектуальну систему для підбору кулінарних рецептів, яка виходить за рамки простого пошуку та пропонує користувачеві персоналізований та динамічний досвід.

1.4 Дані та джерела для кулінарних рецептів

Якість та повнота даних є критично важливими для ефективного функціонування мобільного додатку для підбору кулінарних рецептів, особливо коли він використовує технології штучного інтелекту. Дані формують основу, на якій ШІ-модулі навчаються та роблять рекомендації. Цей підрозділ детально розглядає різні типи кулінарних даних, формати їхнього зберігання, а також методи збору та очищення відповідних датасетів.

1.4.1 Типи кулінарних даних (текст, зображення, відео)

Кулінарні рецепти є мультимодальними за своєю природою, поєднуючи різні типи інформації, які є важливими для розуміння та відтворення страви. Для мобільного додатку це означає необхідність ефективної обробки кожного з цих типів.

1. ***Текстові дані.*** Це основний та найпоширеніший тип даних для кулінарних рецептів. Він включає:
   - *Назва рецепту*: Короткий та чіткий опис страви.
   - *Опис страви*: Детальніша інформація про страву, її походження, смакові якості, призначення.
   - *Список інгредієнтів*: Перелік усіх необхідних продуктів з їхньою кількістю та одиницями виміру (наприклад, "200 г курячого філе", "1 ч.л. солі"). Цей аспект особливо важливий для AI-систем, які працюють з пошуком за наявними інгредієнтами.
   - *Покрокові інструкції з приготування*: Чіткі та послідовні кроки, які потрібно виконати для приготування страви.
   - *Додаткова інформація*: Час приготування, час активної роботи, кількість порцій, рівень складності, дієтичні позначки (веганський, без глютену), харчова цінність (калорії, білки, жири, вуглеводи), поради від кухаря.
   - *Коментарі та відгуки користувачів*: Цінні дані для колаборативної фільтрації та покращення рекомендацій, а також для розуміння популярності та успішності рецептів.
2. ***Зображення.*** Візуальні дані відіграють ключову роль у привабливості кулінарних рецептів.
   - *Фотографії готової страви*. Найважливіший елемент, який допомагає користувачеві візуалізувати кінцевий результат та оцінити його привабливість. Якісні зображення значно підвищують інтерес до рецепту.
   - *Фотографії проміжних етапів приготування*. Деякі рецепти можуть включати фотографії, що ілюструють ключові моменти приготування, що спрощує процес для користувача.

– *Роль у AI.* Комп'ютерний зір (CV) може використовувати зображення для класифікації страв, визначення інгредієнтів (якщо користувач завантажує фото) або оцінки візуальної привабливості.

3. ***Відео.*** Надають найбільш повну та динамічну інформацію про процес приготування.

– *Відео-рецепти.* Покрокові відеоінструкції, які демонструють кожен етап приготування страви.

– *Роль у AI.* Хоча обробка відео є більш ресурсомісткою, ніж тексту чи зображень, AI може використовувати відео для аналізу технік приготування, визначення часу виконання певних кроків або навіть автоматичної генерації текстових інструкцій з відео. У мобільному додатку відео можуть бути інтегровані через посилання на зовнішні платформи (наприклад, YouTube).

1.4.2 Формати зберігання рецептів (JSON, XML, бази даних)

Вибір формату зберігання даних є надзвичайно важливим аспектом, що впливає на ефективність, масштабованість та загальну зручність роботи з кулінарними рецептами. Серед різноманіття форматів особливо виділяється JSON (JavaScript Object Notation). Він характеризується як легкий, зручний для читання людиною та легко оброблюваний машиною формат для обміну даними. JSON отримав широке розповсюдження у веб-розробці та мобільних додатках завдяки своїй простоті та високій сумісності з JavaScript, що робить його ідеальним вибором для проектів, розроблених на React Native. У цьому форматі дані логічно організовані у вигляді пар "ключ-значення" та впорядкованих масивів.

Приклад структури:

JSON

```
{
  "title": "Борщ Український",
  "description": "Класичний український борщ...",
  "ingredients": [
    {"name": "Буряк", "quantity": "2", "unit": "шт."},
    {"name": "Капуста", "quantity": "300", "unit": "г"}
  ],
  "instructions": [
    "Крок 1: Зварити бульйон...",
    "Крок 2: Додати овочі..."
  ],
  "prep_time_minutes": 20,
  "cook_time_minutes": 60,
  "servings": 4,
```

```
  "dietary_tags": ["вегетаріанський", "безглютеновий"],
  "image_url": "https://example.com/borsch.jpg"
}
```

JSON як формат зберігання даних має переваги у простоті парсингу в JavaScript, що забезпечує пряму сумісність з React Native, а також у своїй компактності та широкій підтримці в API. Однак, його недоліком є відсутність суворої схеми, що, хоч і може бути перевагою у деяких випадках, ускладнює роботу з дуже великими обсягами даних без залучення повноцінної бази даних.

XML, хоч і є старішим, все ще використовується для зберігання та обміну даними. Його структура базується на тегах, дозволяючи створювати складні ієрархічні представлення. Перевагами XML є сувора структура, яку можна валідувати за допомогою DTD або XML Schema, та широка підтримка в корпоративних системах. Водночас, його багатослівність призводить до більшого розміру файлів порівняно з JSON, а парсинг у мобільних додатках є складнішим.

Для зберігання значних обсягів рецептів, користувацьких даних, уподобань та взаємодій з рекомендаційною системою найкращим вибором є використання баз даних. Серед них виділяють реляційні бази даних (SQL), такі як PostgreSQL, MySQL або SQLite. Їхня характеристика полягає у зберіганні даних у таблицях зі строго визначеними схемами та зв'язками. Перевагами SQL баз є висока цілісність даних, можливість виконання складних запитів (JOINs), підтримка транзакцій та ACID властивостей. Вони ідеально підходять для зберігання основних структурованих даних про рецепти, користувачів та їхні вподобання, де важливі зв'язки між елементами.

Також існують NoSQL бази даних, наприклад, MongoDB, Firestore або Redis. Вони вирізняються гнучкішими схемами та оптимізовані для горизонтального масштабування, дозволяючи зберігати неструктуровані або напівструктуровані дані. Серед переваг NoSQL баз – висока продуктивність для певних типів запитів, гнучкість у зміні схеми та легкість масштабування. Вони застосовуються для зберігання рецептів як документів (у випадку MongoDB), кешування даних (Redis), а також для зберігання користувацьких сесій та даних про поведінку для рекомендаційних систем, особливо коли структура даних може часто змінюватися

або є дуже складною. Firestore (Firebase), зокрема, є чудовим варіантом для React Native додатків завдяки своїй легкій інтеграції та синхронізації в реальному часі.

Для мобільного додатку на React Native, що працює з кулінарними рецептами, найімовірніше, буде використана комбінація цих підходів: JSON для ефективного обміну даними між клієнтською частиною та сервером, а також реляційна або NoSQL база даних на сервері для постійного зберігання та управління великими обсягами кулінарної інформації.

1.4.3 Методи збору та очищення кулінарних датасетів

Створення якісного кулінарного датасету є фундаментальним етапом для навчання та функціонування ШІ-модулів.

1. ***Методи збору даних*:**
   - *Веб-скрейпінг (Web Scraping)*. Автоматизований збір рецептів з існуючих кулінарних веб-сайтів (наприклад, Allrecipes, Food.com, Cookpad, українські кулінарні портали). Важливо враховувати юридичні та етичні аспекти, а також політику використання даних конкретних сайтів.
   - *Використання існуючих API.* Деякі кулінарні сервіси надають публічні API для доступу до своїх баз даних рецептів (наприклад, Spoonacular API, Edamam API). Це забезпечує більш структурований та легальний спосіб отримання даних.
   - *Публічні датасети*. Існують вже зібрані та доступні публічні датасети кулінарних рецептів (наприклад, RecipeNLG Dataset, Recipe1M+). Вони часто вже очищені та структуровані, але можуть потребувати адаптації до специфічних потреб проекту (наприклад, українська мова).
   - *Краудсорсинг/Введення користувачами*. Можливість для користувачів додатку додавати власні рецепти. Це збагачує датасет, але потребує модерації та валідації.
2. ***Методи очищення кулінарних датасетів*.** Сирі дані, зібрані з різних джерел, майже завжди містять "шум" та невідповідності, які потрібно усунути для коректної роботи ШІ.

- *Видалення дублікатів***.** Ідентифікація та видалення однакових або дуже схожих рецептів.
- *Нормалізація інгредієнтів***.** Приведення назв інгредієнтів до єдиного стандарту (наприклад, "помідор", "томат", "томати" -> "помідор"). Використання словників або онтологій інгредієнтів.
- *Нормалізація одиниць виміру***.** Перетворення різних одиниць виміру до стандартного формату (наприклад, "1 cup" -> "240 ml", "дрібка" -> "1 г").
- *Обробка відсутніх значень***.** Заповнення пропущених даних (наприклад, часу приготування, кількості порцій) або видалення неповних записів.
- *Видалення "шуму" з тексту***.** Усунення рекламних вставлень, зайвих символів, HTML-тегів, нерелевантного тексту з описів або інструкцій.
- *Перевірка граматики та орфографії***.** Особливо важливо для текстових інструкцій.
- *Класифікація та тегування***.** Автоматичне або ручне присвоєння рецептам категорій (сніданок, обід, вечеря), дієтичних тегів (веган, без лактози), кухонь (італійська, українська).
- *Валідація числових даних***.** Перевірка коректності кількості інгредієнтів, часу приготування.

Якісно зібраний та очищений датасет є фундаментом для побудови надійних та точних рекомендаційних систем, модулів обробки природної мови та інших функцій штучного інтелекту в мобільному додатку для підбору кулінарних рецептів. Проведений аналіз закладає теоретичну базу для розробки мобільного кулінарного додатку з ШІ на React Native. Він аналізує кросплатформну розробку, обґрунтовуючи вибір React Native порівняно з Flutter та Xamarin. Детально висвітлює застосування ШІ в кулінарії, класифікуючи завдання (рекомендації, NLP, комп'ютерний зір) та методи машинного навчання (контентна, колаборативна фільтрація, нейронні мережі, LLMs для генерації) з врахуванням інформаційної складової системи: типів кулінарних даних, форматів їх зберігання (JSON, XML, бази даних) та методів збору й очищення датасетів, що є ключовим для функціонування ШІ.

2. ПРОЕКТУВАННЯ ТА РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ

2.1 Формулювання вимог до мобільного додатку

Успішна розробка будь-якого програмного продукту починається з чіткого та всебічного формулювання вимог. Цей етап є критично важливим для забезпечення того, що кінцевий продукт відповідатиме потребам користувачів та бізнесу. Для мобільного додатку для підбору кулінарних рецептів з використанням штучного інтелекту, вимоги повинні охоплювати як взаємодію з користувачем, так і інтеграцію складних AI-механізмів.

Функціональні вимоги описують, що саме система повинна робити, тобто її основні можливості та дії.

1. *Реєстрація та авторизація користувачів***:** користувач повинен мати можливість зареєструватися в додатку за допомогою електронної пошти та пароля; користувач повинен мати можливість авторизуватися у додатку, використовуючи зареєстровані дані; система повинна підтримувати відновлення пароля; можливість реєстрації/авторизації через сторонні сервіси (Google, Facebook).
2. *Керування профілем користувача*. користувач повинен мати можливість переглядати та редагувати інформацію свого профілю (ім'я, вік, стать, дієтичні обмеження, алергії, улюблені кухні, нелюбі інгредієнти); дані є ключовими для персоналізації рекомендацій ШІ; користувач повинен мати можливість видалити свій профіль
3. *Пошук рецептів***:** користувач повинен мати можливість здійснювати пошук рецептів за назвою страви або ключовими словами; користувач повинен мати можливість здійснювати пошук рецептів за наявними інгредієнтами (наприклад, введення списку інгредієнтів з холодильника); система повинна надавати автодоповнення для пошукових запитів.
4. *Система рекомендацій (AI).* Система повинна надавати персоналізовані рекомендації рецептів на основі: історії переглядів та збережених рецептів користувача; даних профілю користувача (дієтичні обмеження, вподобання); оцінок та відгуків інших користувачів (колаборативна фільтрація); популярності

рецептів. Система повинна мати можливість адаптувати рецепти (наприклад, замінити інгредієнт, змінити порції) за запитом користувача. Система може генерувати рецепти на основі заданих користувачем критеріїв (наприклад, "створи рецепт десерту з яблук та кориці")

5. *Перегляд рецепту:* користувач повинен мати можливість переглядати повну інформацію про рецепт: назву, опис, список інгредієнтів з кількістю, покрокові інструкції, час приготування, кількість порцій, дієтичні позначки, зображення; користувач повинен мати можливість переглядати відгуки та оцінки інших користувачів.
6. *Збереження рецептів та списки:* Користувач повинен мати можливість додавати рецепти до обраного/зберігати їх; Користувач повинен мати можливість створювати власні списки рецептів (наприклад, "Рецепти на Новий Рік", "Мої улюблені сніданки")
7. *Оцінка та коментування рецептів:* авторизований користувач повинен мати можливість оцінювати рецепти (наприклад, за 5-бальною шкалою); авторизований користувач повинен мати можливість залишати коментарі до рецептів.
8. *Список покупок:* користувач повинен мати можливість автоматично генерувати список покупок на основі обраних рецептів; Користувач повинен мати можливість редагувати згенерований список покупок (додавати/видаляти елементи).
9. *Багатомовність*. Додаток повинен підтримувати кілька мов інтерфейсу (мінімум українську та англійську

Нефункціональні вимоги описують якість та атрибути системи, а не її функціональність, а саме:

1. ***Продуктивність:*** час завантаження основних екранів додатка не повинен перевищувати 3 секунд при стабільному інтернет-з'єднанні; час пошуку та фільтрації рецептів не повинен перевищувати 2 секунд; час отримання рекомендацій від AI-модуля не повинен перевищувати 5 секунд; додаток повинен працювати плавно, без зависань та ривків (особливо при скролінгу великих списків).

2. ***Зручність інтерфейсу (Usability):*** інтерфейс користувача повинен бути інтуїтивно зрозумілим та легким у використанні; навігація в додатку повинна бути логічною та послідовною; елементи керування повинні бути чіткими та добре розміщеними; додаток повинен мати привабливий та сучасний дизайн (відповідність Material Design для Android та Human Interface Guidelines для iOS, наскільки це можливо в React Native); система повинна надавати чіткі повідомлення про помилки та успішні операції.

3. ***Безпека*****:** Дані користувачів (особисті дані, паролі) повинні зберігатися у зашифрованому вигляді; Взаємодія між мобільним додатком та сервером повинна відбуватися через захищені протоколи (HTTPS/SSL/TLS); Система повинна бути стійкою до поширених типів атак (SQL-ін'єкції, XSS тощо); Аутентифікаційні токени повинні бути захищені та мати обмежений термін дії.

4. ***Масштабованість:*** система повинна бути здатною обробляти зростаючу кількість користувачів та рецептів без значного зниження продуктивності; архітектура повинна дозволяти легке додавання нових функцій та інтеграцію з іншими сервісами; модуль штучного інтелекту повинен мати можливість масштабуватися для обробки більших обсягів даних та складніших запитів.

5. Надійність: додаток повинен бути стабільним та стійким до збоїв; система повинна мати механізми обробки помилок та відновлення після збоїв; Забезпечення доступності сервісу (наприклад, 99.9% uptime).

6. Сумісність: додаток повинен коректно працювати на основних версіях операційних систем iOS та Android (наприклад, iOS 13+ та Android 8+); додаток повинен адаптуватися до різних розмірів екранів та орієнтацій пристроїв.

7. ***Підтримуваність*****:** код повинен бути добре документований та легко підтримуватися іншими розробниками; система повинна мати гнучку архітектуру для майбутніх оновлень та вдосконалень.

Вибір технологічного стеку є фундаментальним рішенням, яке впливає на всі етапи розробки та подальшу підтримку проекту. Для мобільного додатку, призначеного для підбору кулінарних рецептів, було обрано React Native як ключову технологію для фронтенду мобільного додатку.

Обґрунтуванням такого вибору є перш за все його кросплатформність, яка дозволяє використовувати єдину кодову базу для розробки додатків під iOS та Android, значно скорочуючи час і витрати на розробку та подальшу підтримку. Це є критично важливим для проекту, що прагне охопити максимально широку аудиторію. Крім того, React Native забезпечує нативну продуктивність та високий рівень UI/UX, оскільки компілює компоненти у справжні нативні елементи інтерфейсу. Це гарантує високу швидкість відгуку та візуальну відповідність стандартам платформ, що є особливо важливим для плавного скролінгу списків рецептів та ефективного відображення зображень.

Використання JavaScript та TypeScript, широко поширених мов програмування, а також бібліотеки React, дозволяє залучати велику кількість готових бібліотек, інструментів та використовувати досвід веб-розробників, тим самим прискорюючи процес створення. Функції "Hot Reloading" та "Fast Refresh" значно прискорюють розробку та налагодження, дозволяючи швидко бачити зміни в коді. Велика та активна спільнота React Native також забезпечує доступ до численних ресурсів, готових компонентів та оперативної підтримки, що спрощує вирішення можливих проблем.

Оскільки React Native сам по собі є фреймворком для розробки фронтенду мобільних додатків, його вибір автоматично визначає технологію для клієнтської частини. Для ефективного керування станом додатку в React Native можуть бути використані такі бібліотеки, як Redux, React Context API або MobX, які забезпечують централізоване та передбачуване управління даними, отриманими як від бекенду, так і від AI-модулів.

2.2 Архітектура проектованого мобільного додатку

Ефективна архітектура є основою стабільного, масштабованого та легкого у підтримці мобільного додатку. Для розробки мобільного додатку для підбору кулінарних рецептів з використанням штучного інтелекту необхідно обрати архітектурні рішення, які забезпечать надійну взаємодію між клієнтською

частиною (мобільним додатком на React Native), серверною частиною та інтелектуальними модулями.

2.2.1 Загальна архітектура системи (клієнт-серверна, архітектура додатку на React Native)

Загальна архітектура системи буде базуватися на **клієнт-серверній моделі**, що є стандартом для більшості сучасних мобільних додатків. Це дозволяє розвантажити мобільний пристрій від обчислювальних операцій (особливо пов'язаних зі штучним інтелектом та обробкою великих обсягів даних), забезпечити централізоване зберігання та управління даними, а також легкість оновлення логіки без необхідності оновлення самого додатку в магазинах. Розглянемо основні складові архітектури (рис. 2.1):

1. ***Клієнтська частина (Мобільний додаток на React Native).*** Мобільний додаток, розроблений на React Native, є головним інтерфейсом для користувача. Завдяки єдиній кодовій базі він забезпечує кросплатформність, працюючи як на iOS, так і на Android.

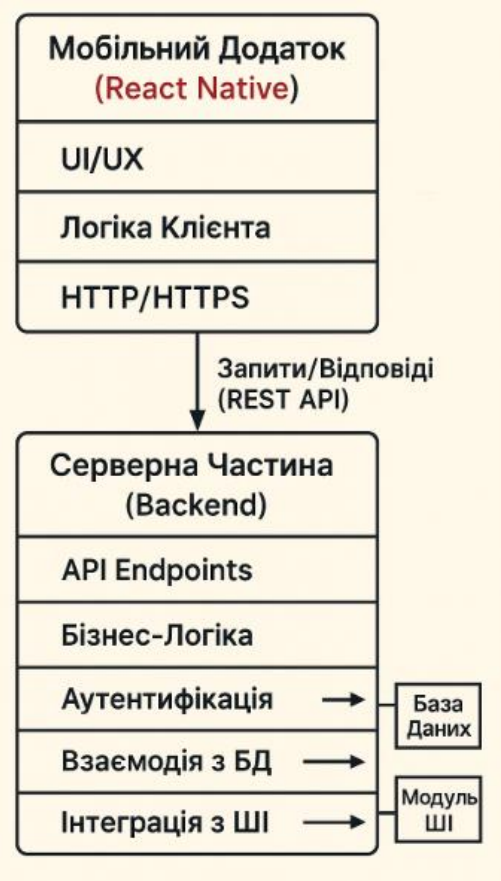


Рисунок 2.1 Діаграма загальної архітектури мобільної системи.

Цей додаток має зручний *користувацький інтерфейс (UI)***,** який відображає списки рецептів, їхні деталі, а також форми для пошуку та фільтрації, екрани профілю та списку покупок, забезпечуючи інтуїтивну навігацію. Для *взаємодії з користувачем (UX)* програма обробляє введення тексту, вибір фільтрів, натискання кнопок, свайпи та інші жести.

Додаток надсилає запити до серверної частини, використовуючи HTTP/HTTPS (GET, POST, PUT, DELETE) для отримання рецептів, збереження уподобань, авторизації та отримання рекомендацій від штучного інтелекту. Також передбачене локальне кешування деяких даних на пристрої для швидкого доступу та роботи в офлайн-режимі, наприклад, для збереження останніх переглянутих рецептів або чернеток списків покупок. Управління станом здійснюється за допомогою бібліотек, таких як Redux, Context API або Zustand, що гарантує передбачуване та централізоване керування даними та логікою інтерфейсу.

2. ***Серверна частина (Backend).*** Серверна частина є основою усієї системи, відповідальним за виконання основної логіки, взаємодію з базою даних та інтеграцію з модулями штучного інтелекту. Вона надає *API (Application Programming Interface)***,** зазвичай RESTful або GraphQL, який слугує єдиною точкою входу для всіх запитів від мобільного додатку, забезпечуючи взаємодію з ним. Серверна частина здійснює управління даними, що включає зберігання, отримання, оновлення та видалення інформації про рецепти, інгредієнти, користувачів, їхні вподобання, оцінки та коментарі.

Бекенд відповідає за реалізацію бізнес-логіки**,** обробляючи запити на пошук, фільтрацію та формування списків покупок. Він забезпечує аутентифікацію та авторизацію, перевіряючи права доступу користувачів до певних ресурсів та функцій. Крім того, серверна частина налагоджує взаємодію з модулем ШІ, передаючи йому запити від клієнта та повертаючи отримані результати назад. Нарешті, вона займається обробкою файлів, зберігаючи та обслуговуючи зображення рецептів, а також веде моніторинг та логування, записуючи події та помилки для аналізу та підтримки системи.

3. ***Модуль штучного інтелекту (AI Module).*** Модуль штучного інтелекту — це окремий сервіс або група сервісів, які відповідають за інтелектуальні функції додатка. Він може бути розгорнутий як частина бекенду або як окремий мікросервіс. Його основні функції включають формування рекомендацій**,** аналіз користувацьких запитів та обробку даних для персоналізації. Детальніше про цей модуль буде розказано в пункті 2.4

2.2.2 Архітектура модуля штучного інтелекту для підбору рецептів з інтеграцією API-сервісу рекомендацій

Модуль штучного інтелекту є центральною інтелектуальною частиною додатку. Його архітектура повинна бути гнучкою, щоб підтримувати різні алгоритми ШІ та легко масштабуватися. Існують два основні підходи до інтеграції ШІ: власна розробка та інтеграція зі сторонніми сервісами/API. Для даного проекту, зважаючи на складність розробки повноцінної рекомендаційної системи з нуля, найбільш ефективним буде інтеграція з API сервісу рекомендацій. Розглянемо загальні компоненти архітектури модуля штучного інтелекту (рис. 2.2):

1. ***Компоненти модуля ШІ.*** Цей компонент відповідає за регулярний збір нових рецептів та оновлення вже наявних. Він також займається обробкою користувацьких даних, таких як історія переглядів, оцінки та профілі. Джерелами даних для цього компонента є база даних рецептів та лог-файли взаємодії користувачів з додатком. Для виконання своїх функцій використовуються технології скриптів для ETL (Extract, Transform, Load), а також інструменти для очищення та нормалізації даних.
2. ***Компонент збору та обробки даних (Data Ingestion & Processing).*** Даний компонент відповідає за регулярне поповнення бази новими рецептами, їх оновлення, а також за обробку інформації про користувачів, включаючи історію переглядів, оцінки та профілі. Він черпає дані з бази рецептів та лог-файлів взаємодії користувачів з додатком. Для цього використовуються спеціальні ETL-скрипти (Extract, Transform, Load) та інструменти, що допомагають очищати й нормалізувати інформацію.

3. ***Компонент формування ознак (Feature Engineering).*** Компонент відповідає за перетворення сирих даних у формат, який можна використовувати для навчання моделей машинного навчання. Це передбачає створення векторних представлень (ембедінгів) для інгредієнтів, категорій, кухонь, а також для користувачів та рецептів. Для цього використовуються такі технології, як бібліотеки для обробки природної мови (NLP), наприклад, spaCy та NLTK, для аналізу тексту рецептів, а також бібліотеки для роботи з даними, такі як Pandas та NumPy.

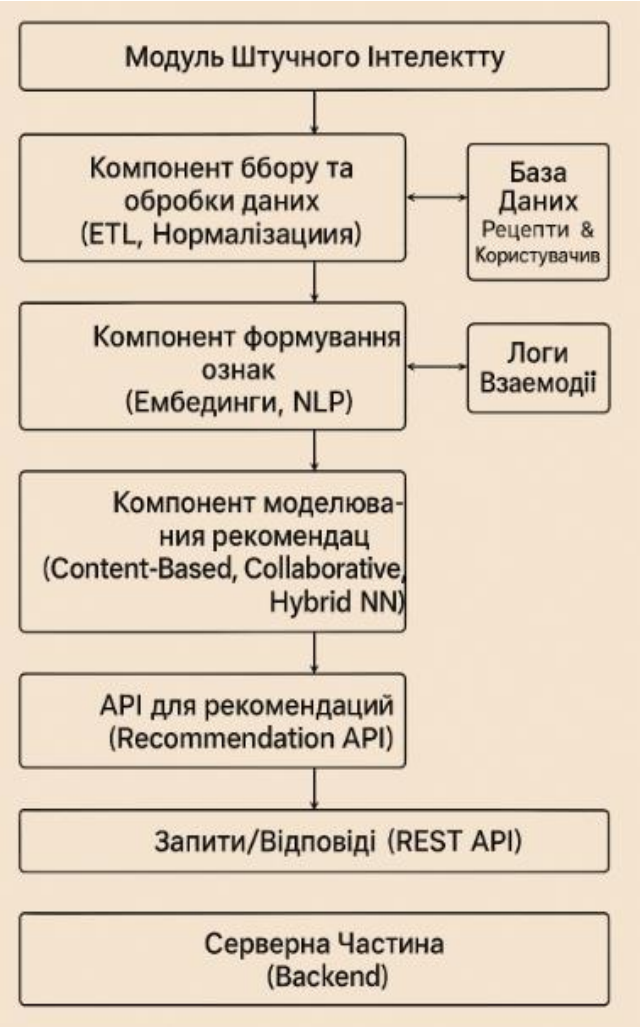


Рисунок 2.2 Діаграма архітектури модуля ШІ

4. ***Компонент моделювання рекомендацій (Recommendation Model).*** Цей компонент є серцем модуля штучного інтелекту, в якому реалізовані всі алгоритми рекомендацій. Він може складатися з однієї або комбінації різних моделей.

У цьому компоненті використовуються такі підходи, як фільтрація на основі вмісту, де моделі порівнюють профіль користувача з профілями рецептів, наприклад, за допомогою косинусної схожості між векторами ознак. Також

застосовується колаборативна фільтрація, що включає моделі, які шукають схожих користувачів або рецепти на основі взаємодій, використовуючи такі методи, як SVD, NMF або нейронні мережі для матричної факторизації. Часто впроваджуються гібридні моделі, що поєднують обидва підходи для підвищення точності та подолання проблеми "холодного старту". Крім того, для вивчення складних нелінійних залежностей у даних користувачів та рецептів застосовується глибоке навчання з використанням нейронних мереж, наприклад, нейронних рекомендаційних систем.

Для побудови та навчання цих моделей використовуються технології з бібліотек машинного навчання, таких як Scikit-learn, TensorFlow та PyTorch.

5. ***Процес взаємодії з модулем ШІ:***

– *Збір даних***:** Мобільний додаток та бекенд збирають дані про користувача (профіль, історія переглядів, оцінки, збережені рецепти) та нові рецепти.

– *Обробка та формування ознак***:** Ці дані передаються до модуля ШІ, де вони очищаються, нормалізуються та перетворюються на векторні представлення або інші придатні для моделювання формати.

– *Навчання/Оновлення моделі***.** Моделі рекомендацій періодично перенавчаються або оновлюються на нових даних, щоб адаптуватися до змін у вподобаннях користувачів та появі нових рецептів.

– *Запит на рекомендацію*. Коли користувач у мобільному додатку запитує рекомендації, мобільний клієнт надсилає запит на бекенд.

– *Взаємодія з ШІ-API*. Бекенд, у свою чергу, надсилає запит до Recommendation API модуля ШІ, передаючи необхідні параметри (ID користувача, поточні інгредієнти тощо).

– *Формування рекомендацій*. Модуль ШІ використовує свою навчену модель для генерації списку рекомендованих рецептів.

– *Повернення результату***.** Результат (список ID рецептів або їхні основні дані) повертається до бекенду, а потім до мобільного додатку для відображення користувачеві.

Така архітектура дозволить розробити гнучкий та ефективний мобільний додаток, де React Native забезпечить якісний інтерфейс, а відокремлений модуль ШІ — інтелектуальну складову для персоналізованих рекомендацій.

2.3 Дизайн користувацького інтерфейсу (UI/UX)

Якісний дизайн користувацького інтерфейсу (UI) та користувацького досвіду (UX) є вирішальним для успіху будь-якого мобільного додатку, особливо для того, що має на меті спростити процес приготування їжі. Привабливий UI та інтуїтивно зрозумілий UX забезпечують приємну взаємодію з додатком, підвищують задоволеність користувачів та заохочують їх до регулярного використання. Для мобільного додатку з підбору кулінарних рецептів, розробленого на React Native, необхідно враховувати як загальні принципи дизайну, так і специфіку платформ iOS та Android.

2.4.1 Розробка прототипів та макетів інтерфейсу

Процес розробки UI/UX починається задовго до написання коду і включає кілька етапів:

1. ***Дослідження користувачів та конкурентів*:**

–Цільова аудиторія: Визначення, хто є потенційними користувачами (початківці кухарі, досвідчені, ті, хто дотримується дієт, студенти, зайняті люди). Це допоможе зрозуміти їхні потреби, болі та очікування від додатку.

–Аналіз конкурентів: Детальне вивчення існуючих кулінарних додатків, щоб виявити їхні сильні та слабкі сторони, найкращі практики та уникнути типових помилок.

2. ***Створення сценаріїв використання (User Flows):***

– Визначення послідовності дій, які користувач виконуватиме для досягнення конкретних цілей (наприклад, "знайти рецепт борщу", "зберегти рецепт", "сформувати список покупок").

– Сценарії допомагають візуалізувати шлях користувача по додатку і виявити можливі перешкоди або нелогічні переходи.

3. ***Розробка Wireframes (каркасів):***

– Низькодеталізовані, схематичні представлення екранів додатку, які фокусуються на розташуванні основних елементів (кнопки, текстові поля, зображення, навігація).

–Wireframes допомагають швидко і дешево протестувати різні варіанти розміщення елементів, не відволікаючись на візуальний дизайн. Можуть бути намальовані вручну або створені в простих інструментах (наприклад, Figma, Sketch, Adobe XD).

4. ***Створення Prototype (прототипів):***

–Інтерактивні версії Wireframes або більш деталізованих макетів, які імітують поведінку додатку.

–Прототипи дозволяють тестувати потік користувача, переходи між екранами та базову функціональність ще до початку розробки, виявляючи проблеми UX на ранніх стадіях.

–Можуть бути створені за допомогою інструментів, що дозволяють створювати інтерактивні "клікабельні" макети (Figma, Sketch, Adobe XD).

5. ***Розробка Mockups (макетів) / High-Fidelity Design*:**

–Високодеталізовані візуальні представлення кожного екрану додатку, що включають кольори, типографіку, іконки, зображення та інші графічні елементи.

–Ці макети відображають остаточний вигляд інтерфейсу. Вони є основою для розробників та визначають стиль та бренд додатку.

–Для React Native важливо враховувати специфіку компонентів та гайдлайнів для iOS (Human Interface Guidelines) та Android (Material Design), щоб забезпечити нативний вигляд та відчуття.

В результаті вище сказаного отримаємо макет (рис. 2.3) мобільного додатку для генерації кулінарних рецептів. На екранах видно інтерфейс пошуку рецептів з полями для введення запиту та кнопками для фільтрації за категоріями (кухні, інгредієнти). Також відображаються "Ідеї рецептів" з фотографіями, назвами та, можливо, часом приготування. Внизу екрана розташована навігаційна панель.

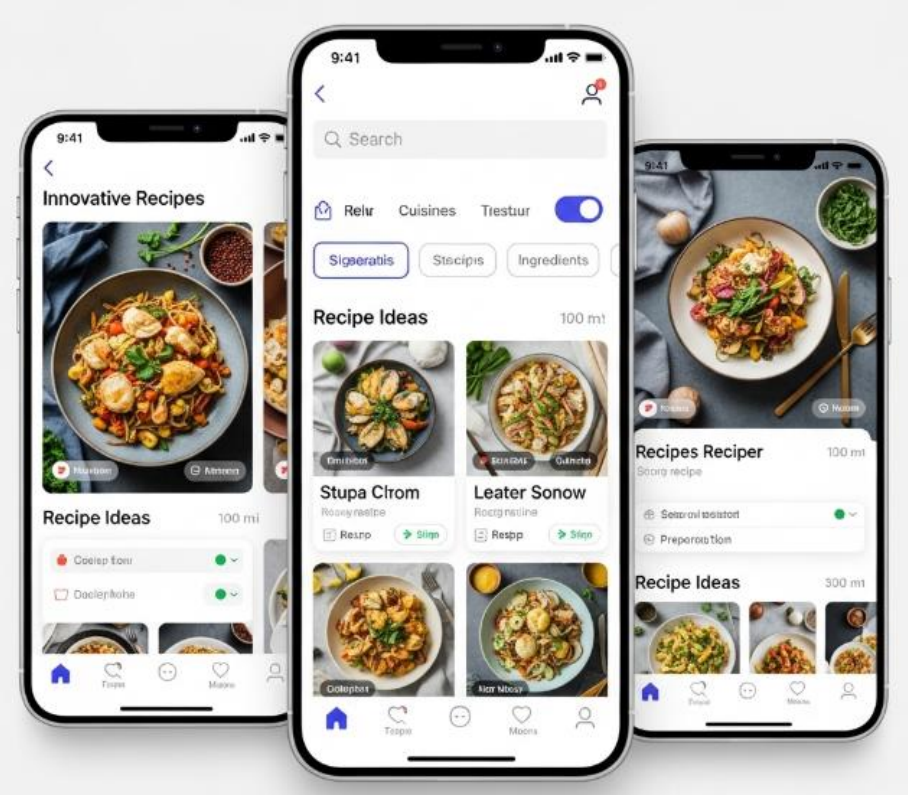


Рисунок 2.3 Прототип мобільного застосунку генерації кулінарних рецептів

2.4.2 Принципи зручності та інтуїтивності інтерфейсу

При розробці UI/UX мобільного додатку для підбору кулінарних рецептів керувалися наступними ключовими принципами:

1. ***Простота та ясність:***
   – Уникати надмірності інформації на екрані. Кожен елемент повинен мати чітке призначення.
   – Використовувати зрозумілу та лаконічну термінологію для кнопок, меню та інструкцій.
   – Мінімізувати кількість кроків для виконання типових дій (наприклад, пошук рецепту, збереження в обране).
2. ***Консистентність:***
   –Використовувати єдиний стиль, кольорову палітру, шрифти, іконки та розташування елементів на всіх екранах додатку.
   –Повторювані елементи (наприклад, кнопки "назад", іконки обраного) повинні виглядати та поводитися однаково. Це знижує когнітивне навантаження на користувача.

3. ***Зворотний зв'язок:***
   – Система повинна надавати миттєвий зворотний зв'язок на дії користувача (наприклад, візуальне підтвердження натискання кнопки, індикатор завантаження, повідомлення про успішне збереження).
   – Чіткі повідомлення про помилки, що пояснюють, що пішло не так і як це виправити.
4. ***Ефективність:***
   – Дозволити користувачам швидко досягати своїх цілей. Наприклад, швидкий доступ до пошуку, фільтрів, персональних рекомендацій.
   – Оптимізувати взаємодію з AI: якщо AI генерує рецепт, показати прогрес або швидкість генерації.
5. ***Естетика та візуальна привабливість:***
   – Використання якісних зображень страв, що викликають апетит.
   – Приємна кольорова палітра, що асоціюється з кулінарією (наприклад, теплі тони, свіжі кольори).
   – Продумана типографіка, що забезпечує легкість читання тексту рецептів (особливо списків інгредієнтів та інструкцій).
6. ***Доступність (Accessibility):***
   – Забезпечити достатній контраст між текстом та фоном.
   – Використовувати чіткі, розбірливі шрифти.
   – Враховувати потреби користувачів з обмеженими можливостями (наприклад, підтримка голосового керування, збільшення шрифту).
7. ***Мобільна специфіка:***
   – Дизайн повинен бути оптимізований для взаємодії пальцями (достатній розмір та відступи для інтерактивних елементів).
   – Оптимізація для різних розмірів екранів та орієнтації пристрою (адаптивний дизайн).
   – Використання стандартних нативних елементів, які знайомі користувачам iOS та Android (наприклад, кнопки, скролл-вікна, перемикачі). React Native надає компоненти, які за замовчуванням виглядають "нативними".

2.4.3 Дизайн основних екранів додатку (головний екран, пошук, деталі рецепту, профіль)

На основі вищезгаданих принципів та з урахуванням функціональних вимог, було розроблено дизайн ключових екранів мобільного додатку.

1. ***Головний екран (Home Screen).*** Необхідно надати швидкий доступ до найактуальнішої інформації та персоналізованих рекомендацій з елементами**:**
   - *Верхня панель*. Логотип додатку, кнопка пошуку, кнопка профілю користувача.
   - *Блок персоналізованих рекомендацій від ШІ*. Великі картки рецептів, що рекомендуються на основі історії користувача та його профілю. Може бути прокручуваним списком.
   - *Категорії/популярні розділи*. Горизонтальний прокручуваний список популярних категорій (наприклад, "Швидкі вечері", "Здорова їжа", "Сезонні рецепти").
   - *Нещодавно переглянуті рецепти*. Список рецептів, які користувач переглядав останнім часом.
   - *Нижня навігаційна панель (Bottom Navigation Bar):* Забезпечує легкий доступ до основних розділів додатку. "Головна", "Пошук", "Обране", "Профіль".
2. ***Екран пошуку (Search Screen).*** Забезпечити швидкий та ефективний пошук рецептів за різними критеріями елементів:
   - *Рядок пошуку*. Велике та помітне поле для введення запиту.
   - *Фільтри.* Кнопка, що відкриває панель або окремий екран з розширеними фільтрами (категорії, кухні, дієтичні обмеження, час тощо).
   - *Історія пошуку / Популярні запити*. Під полем пошуку відображаються останні пошукові запити або найбільш популярні ключові слова.
   - *Результати пошуку*. Список рецептів, що відповідають запиту, представлений у вигляді карток з назвою, зображенням, короткою інформацією та рейтингом.

– *AI-підказки.* Можуть бути інтегровані підказки від ШІ, наприклад, "Можливо, ви шукаєте [назва страви]?" або "Спробуйте додати [інгредієнт]".

Рисунок 2.4 Процеси функціонування проектованого мобільного додатку

3. ***Екран деталей рецепту (Recipe Detail Screen).*** Надає повну та зручну структуровану інформацію про рецепт з елементами:

– *Зображення страви.* Велике, високоякісне зображення у верхній частині екрану.

– *Назва рецепту.* Чітка та помітна.

– *Оцінка та кількість відгуків.* Відображення середньої оцінки та кількості тих, хто оцінив.

– *Кнопки дій.* "Зберегти" (в обране), "Поділитися", "Додати до списку покупок".

– *Основна інформація.* Час приготування, кількість порцій, рівень складності, дієтичні позначки (виділені іконками).

– *Список інгредієнтів.* Чітко структурований список з кількістю та одиницями виміру. Можливість відзначити інгредієнти, які вже є.
– *Покрокові інструкції.* Нумерований список кроків приготування. Текст має бути добре читабельним.
– *Розділ коментарів та відгуків.* Можливість переглядати та додавати власні коментарі.
– *Кнопка "Спитати AI".* Можливість задати питання про рецепт чат-боту (наприклад, "чим замінити яйця?").

4. ***Екран профілю користувача (User Profile Screen).*** Дозволити користувачу керувати своїм профілем та переглядати персоналізовані дані з елементами:
– *Аватар та ім'я користувача***.**
– *Розділи профілю*. "Мій профіль" (редагування даних), "Мої рецепти" (збережені), "Мої списки покупок", "Мої оцінки та коментарі".
– *Налаштування*. Зміна мови, сповіщень.
– *Кнопка "Вийти".*

Наведений макет та описані принципи дозволять створити мобільний додаток, який буде не тільки функціональним, але й максимально зручним та привабливим для користувачів.

2.4 Інтеграція AI-модуля в мобільному додатку

Розробка AI частини є серцем інтелектуального мобільного додатку для підбору кулінарних рецептів. Цей розділ детально описує етапи реалізації функціоналу штучного інтелекту, що забезпечує персоналізовані рекомендації та ефективну взаємодію з даними. AI-модуль, як правило, реалізується на серверній стороні (бекенд), оскільки обчислення та зберігання великих моделей вимагають значних ресурсів, недоступних на мобільних пристроях.

2.4.1 Реалізація API для взаємодії з мобільним додатком

Для забезпечення безшовної взаємодії між мобільним додатком на React Native та AI-модулем необхідно розробити надійний та ефективний API

(Application Programming Interface). Цей API буде служити "мостом" для обміну даними та запитами, а саме:

1. **Вибір технології для API:** Найчастіше для таких завдань використовують Python-фреймворки, такі як **Flask** або **FastAPI** (для більш високої продуктивності та асинхронності), або **Django REST Framework** (для великих проектів з комплексним управлінням даними). Вибір залежить від масштабу проекту та переваг розробників.
2. **Типи API-ендпойнтів:**
   - **Аутентифікація та авторизація:**
     - POST /api/auth/register: Реєстрація нового користувача.
     - POST /api/auth/login: Авторизація користувача, видача JWT-токену.
     - POST /api/auth/refresh_token: Оновлення JWT-токену.
   - **Керування профілем користувача:**
     - GET /api/users/{user_id}/profile: Отримання даних профілю.
     - PUT /api/users/{user_id}/profile: Оновлення даних профілю (дієтичні обмеження, алергії тощо).
   - **Отримання та керування рецептами:**
     - GET /api/recipes: Отримання списку всіх рецептів (з пагінацією та базовими фільтрами).
     - GET /api/recipes/{recipe_id}: Отримання детальної інформації про конкретний рецепт.
     - GET /api/recipes/search?query={text}&ingredients={list}: Пошук рецептів за назвою/інгредієнтами.
     - GET /api/recipes/filter?category={id}&cuisine={id}&diet={tag}: Фільтрація рецептів.
   - **Взаємодія з рекомендаційною системою (AI-специфічні ендпойнти):**
     - GET /api/recommendations/user/{user_id}: Отримання персоналізованих рекомендацій для конкретного користувача (на основі його історії, профілю).

- GET /api/recommendations/similar/{recipe_id}: Отримання схожих рецептів до заданого.
- POST /api/recommendations/from_ingredients: Запит на рекомендації на основі списку наявних інгредієнтів.
- POST /api/generate/recipe: (Опціонально) Запит до LLM на генерацію нового рецепту за заданими параметрами.

– **Уподобання та взаємодія з рецептами:**
- POST /api/recipes/{recipe_id}/favorite: Додати рецепт до обраного.
- DELETE /api/recipes/{recipe_id}/favorite: Видалити з обраного.
- POST /api/recipes/{recipe_id}/rate: Оцінити рецепт.
- POST /api/recipes/{recipe_id}/comment: Залишити коментар.

– **Списки покупок:**
- GET /api/shopping_list/{user_id}: Отримання списку покупок користувача.
- POST /api/shopping_list/{user_id}/add: Додати інгредієнт до списку.
- PUT /api/shopping_list/item/{item_id}: Оновити статус елемента (наприклад, "куплено").
- POST /api/shopping_list/generate_from_recipes: Згенерувати список покупок із обраних рецептів.

3. **Формат обміну даними:** Використовується **JSON** для всіх запитів та відповідей, що є стандартом для веб-додатків та легко інтегрується з React Native.

В результаті розглянутого наведемо структурну схему генерації кулінарних рецептів (рис. 2.5), яка детально пояснює, як функціонує модуль штучного інтелекту в мобільному додатку для створення кулінарних рецептів. Весь процес розпочинається з "User Input" (Вхідних даних користувача), де користувач вводить свій запит для генерації рецепта.

Надалі від вхідних даних користувача інформація направляється до декількох модулів обробки та баз даних. Модуль "Natural Language Processing" (Обробка Природної Мови) інтерпретує текстовий запит користувача, наприклад, "хочу веганський рецепт з броколі". Одночасно "Ingredient Database" (База Даних

Інгредієнтів) надає інформацію про різні інгредієнти та їх властивості. "Recipe Structure Understanding" (Розуміння Структури Рецепту) допомагає ШІ-моделі визначити правильну структуру майбутнього рецепта, включаючи заголовок, інгредієнти та кроки приготування.

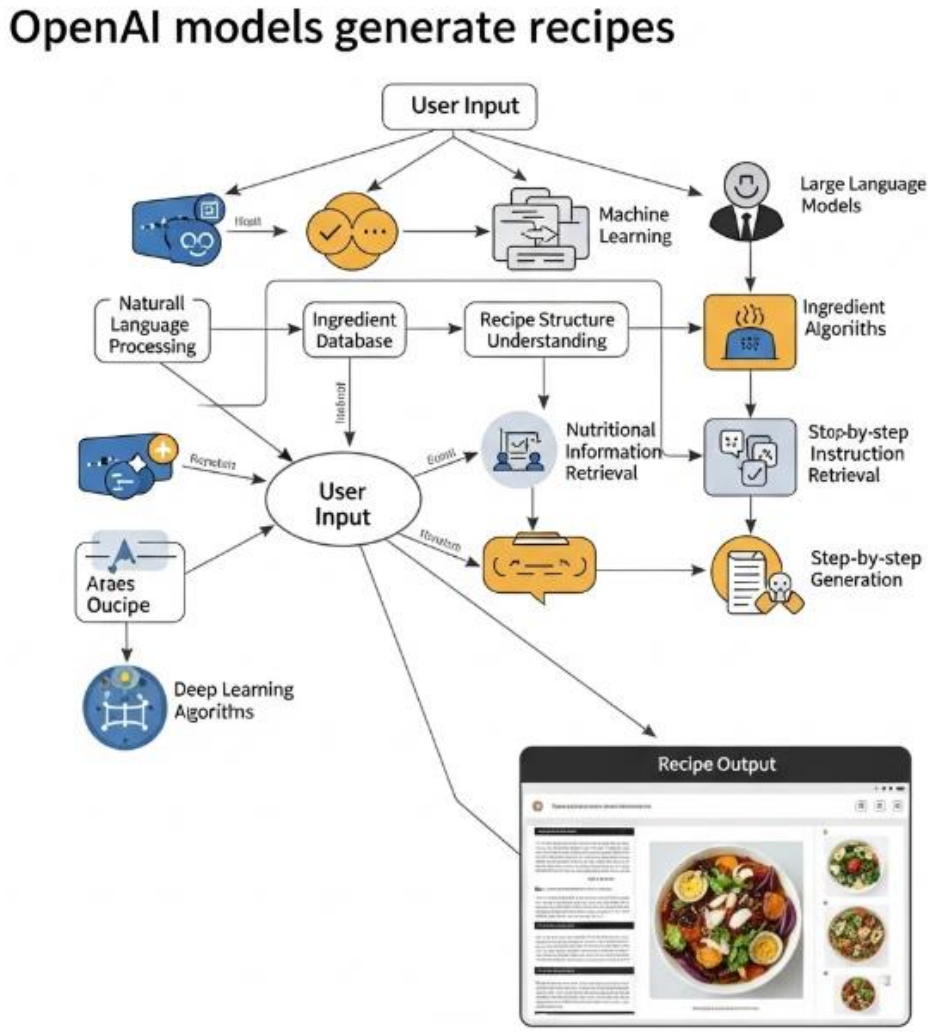


Рисунок 2.5 Структурна схема модуля штучного інтелекту в мобільному додатку

Наведені етапи обробки даних та взаємодії підтримуються ключовими технологіями штучного інтелекту. "Machine Learning" (Машинне Навчання) забезпечує алгоритми, які навчаються на даних для виконання завдань. "Large Language Models" (Великі Мовні Моделі) є основним компонентом для генерації текстового вмісту рецепта, базуючись на великих обсягах текстових даних. "Ingredient Algorithms" (Алгоритми Інгредієнтів) спеціалізуються на роботі з інгредієнтами та їх сумісністю, а "Deep Learning Algorithms" (Алгоритми

Глибокого Навчання) використовують нейронні мережі для складних завдань, таких як обробка мови та розпізнавання закономірностей.

На основі цих оброблених даних та знань зі ШІ-моделей система здатна отримувати "Nutritional Information" (Поживну Інформацію) про інгредієнти та страви, а також формувати логічну послідовність кроків приготування за допомогою "Step-by-step Instruction Retrieval" (Отримання Покрокових Інструкцій) та "Step-by-step Generation" (Покрокової Генерації).

Усі ці компоненти об'єднуються на фінальному етапі — "Recipe Output" (Вивід Рецепту), де мобільний додаток відображає згенерований кулінарний рецепт користувачеві, включаючи зображення готової страви. Отже, ШІ-модуль приймає запит користувача, обробляє його за допомогою методів обробки природної мови та машинного навчання, взаємодіє з базами даних і знань, а потім генерує повний та персоналізований рецепт.

2.4.2 Розробка логіки підбору та рекомендації рецептів на основі AI

Ядро AI-частини реалізується алгоритмами машинного навчання для інтелектуального підбору, до якого варто віднести наступні етапи:

1. **Збір та підготовка даних:**
   - Регулярне вивантаження даних з основної бази даних (рецепти, інгредієнти, користувачі, взаємодії).
   - **Очищення даних:** Видалення дублікатів, обробка пропущених значень, нормалізація назв інгредієнтів та одиниць виміру.
   - **Формування ознак (Feature Engineering):** Перетворення текстових даних (описів рецептів, інгредієнтів) у числові векторні представлення (ембедінги). Для інгредієнтів можна використовувати Word2Vec або FastText, а для цілих описів рецептів – ембедінги з моделей на основі Трансформерів (наприклад, sentence-transformers).
     - **Профіль користувача:** Створення вектору вподобань користувача на основі його збережених рецептів, оцінок, дієтичних обмежень та нелюбових інгредієнтів.

- **Профіль рецепту:** Створення вектору ознак рецепту, що включає інгредієнти, категорії, кухню, час приготування, калорійність тощо.

2. **Вибір та реалізація алгоритмів AI:**

– **Система рекомендацій:**

- **Content-Based Filtering:** Реалізація розрахунку косинусної схожості між вектором профілю користувача та векторами рецептів. Коли користувач шукає рецепти за інгредієнтами, система знаходить рецепти, які містять ці інгредієнти, а потім ранжує їх на основі інших ознак (наприклад, схожості з попередніми вподобаннями користувача).
- **Collaborative Filtering (Item-based):** Визначення схожості між рецептами на основі того, як користувачі взаємодіяли з ними. Наприклад, якщо користувачі, яким сподобався Рецепт А, також часто лайкають Рецепт Б, то Рецепт Б може бути рекомендований тим, хто сподобався Рецепт А. Це може бути реалізовано за допомогою матричної факторизації (SVD) або ж нейронних мереж.
- **Гібридні моделі:** Комбінування Content-Based та Collaborative Filtering для отримання більш точних та різноманітних рекомендацій, а також для подолання проблеми "холодного старту".

– **Моделі для генерації тексту.** Якщо планується генерація нових рецептів або розширена взаємодія з чат-ботом, інтеграція з великими мовними моделями (LLMs) через їхні API (наприклад, OpenAI GPT-4, Llama 3). Запити до цих моделей будуть надсилатися з бекенду з відповідними промптами (вхідними інструкціями).

3. **Навчання та оновлення моделей:**

– Моделі машинного навчання потребують періодичного перенавчання на нових даних, щоб адаптуватися до змін у вподобаннях користувачів та появі нових рецептів.

– Процес перенавчання може бути автоматизованим (наприклад, щоденний або щотижневий запуск).

4. **Розгортання моделей:** Навчені моделі розгортаються як окремі сервіси за допомогою Flask, FastAPI або ж спеціалізованих ML-платформ як Sagemaker, Google AI Platform.

2.4.3. Механізми аутентифікації та авторизації користувачів

Безпека є пріоритетом. Для захисту даних користувачів та доступу до функціоналу використовуються стандартні механізми:

1. **Аутентифікація (Authentication).** Перевірка особи користувача
   - **Метод: JSON Web Tokens (JWT).** При успішній реєстрації або авторизації, сервер видає JWT-токен. Цей токен містить інформацію про користувача (наприклад, його ID) та підпис, що гарантує його цілісність.
   - **Робота з JWT:** Мобільний додаток зберігає JWT-токен (у AsyncStorage в React Native) і надсилає його з кожним запитом до захищених ендпойнтів бекенду в заголовку Authorization (наприклад, Bearer <token>).
2. **Авторизація (Authorization).** Визначення, які дії дозволено виконувати аутентифікованому користувачеві
   - **Метод:** На основі ролей або дозволів. Для більшості функцій додатка (збереження рецептів, оцінки, коментарі, отримання персональних рекомендацій) необхідна авторизація.
   - **Реалізація:** Перед обробкою запиту, перевіряється наявність та валідність JWT-токену. Якщо токен дійсний, з нього витягується ID користувача. Потім перевіряються права доступу користувача до запитуваного ресурсу (наприклад, чи може користувач видалити чужий коментар).
   - **Взаємодія з AI-модулем:** Запити до AI-модуля на рекомендації або генерацію також можуть вимагати авторизації, щоб AI-модуль міг отримати доступ до профілю користувача та його історії для персоналізації.

2.4.4. Валідація даних та обробка помилок

Надійна система повинна ефективно обробляти неправильні вхідні дані та непередбачувані ситуації за наступних етапів:

1. **Валідація даних на стороні клієнта (React Native).** Первинна валідація введених користувачем даних (наприклад, перевірка формату email, довжини пароля, чи є поле обов'язковим). Це покращує UX, надаючи миттєвий зворотний зв'язок. Використання бібліотек для валідації (наприклад, Pydantic у FastAPI, Django Forms/Serializers).**: :**
2. **Обробка помилок на стороні клієнта (React Native**
   - Перехоплення помилок, отриманих від API.
   - Відображення дружніх для користувача повідомлень про помилки (наприклад, "Не вдалося завантажити рецепти, спробуйте пізніше" замість технічного тексту помилки).
   - Реалізація механізмів повторних спроб (retry logic) для тимчасових мережевих помилок.
   - Використання систем моніторингу помилок (наприклад, Sentry, Crashlytics) для відстеження збоїв у мобільному додатку.

Наведені аспекти розробки AI-частини гарантують не тільки функціональність та інтелектуальність додатку, але й його надійність, безпеку та зручність у використанні.

Цей розділ присвячений детальному плануванню архітектури мобільного додатку для підбору кулінарних рецептів зі ШІ на React Native. Він охоплює формулювання функціональних (реєстрація, пошук, рекомендації, профайл, збереження, оцінки, списки покупок) та нефункціональних вимог (продуктивність, безпека, масштабованість). Обґрунтовується вибір React Native через його кросплатформність. Описано клієнт-серверну архітектуру та інтеграцію ШІ-модуля через API. Розглянуто проектування бази даних (PostgreSQL) для зберігання інформації про рецепти, користувачів та вподобання. Деталізовано процес дизайну UI/UX, включаючи створення прототипів та макетів основних екранів. Нарешті, висвітлюється технічна реалізація ШІ-функцій, розробка API, алгоритмів рекомендацій (фільтрація, ембедінги), механізмів аутентифікації (JWT) та обробки помилок, що забезпечує надійність та безпеку системи.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ДОДАТКУ

3.1 Середовище розробки та використані технології

Ефективне налаштування середовища розробки та розумний вибір технологій є запорукою успішної реалізації мобільного додатку. Цей підрозділ детально описує інструменти та бібліотеки, які були використані для створення мобільного додатку для підбору кулінарних рецептів на React Native, а також процес його розгортання.

3.1.1 Налаштування середовища розробки React Native

Для розробки кросплатформного мобільного додатку на React Native було обрано підхід **Expo Go CLI (Command Line Interface)**, який спрощує процес налаштування та розробки порівняно з традиційним React Native CLI, особливо для початківців та проектів, які не потребують глибокої інтеграції з нативним кодом на початкових етапах.

1. **Встановлення Node.js та npm/Yarn:**
   - **Node.js:** JavaScript-середовище виконання, необхідне для роботи React Native та його інструментів. Була встановлена стабільна LTS (Long Term Support) версія Node.js.
   - **npm (Node Package Manager) або Yarn:** Менеджери пакетів для Node.js, що використовуються для встановлення бібліотек та залежностей проекту. Перевага надавалася Yarn завдяки його швидкості та механізмам кешування.
2. **Встановлення Expo CLI.** Глобальне встановлення Expo CLI (npm install -g expo-cli або yarn global add expo-cli) дозволило створювати та управляти Expo-проектами. Expo абстрагує багато складнощів нативної розробки, надаючи готовий набір інструментів та API
3. **Ініціалізація React Native проекту за допомогою Expo.** Створення нового проекту здійснювалося командою expo init <назва_проекту>, з вибором шаблону blank (чистий проект) для максимальної гнучкості. Ця команда автоматично налаштовує структуру проекту, package.json та базові конфігураційні файли**:**

4. **Налаштування редактора коду:** Використовувався **Visual Studio Code (VS Code)** як основний редактор коду. Були встановлені розширення, що значно покращують продуктивність розробки React Native:
   - **ESLint:** Для дотримання стандартів кодування та виявлення потенційних помилок.
   - **Prettier:** Для автоматичного форматування коду, забезпечення єдиного стилю.
   - **React Native Tools:** Для налагодження та інтеграції з інструментами React Native.
   - **TypeScript (якщо використовується):** Вбудована підтримка TypeScript у VS Code.
5. **Налаштування симуляторів/емуляторів та реальних пристроїв:**
   - **Android Studio:** Встановлення Android Studio дозволило налаштувати та запускати Android-емулятори різних версій та розмірів екранів для тестування додатку. ADB (Android Debug Bridge) використовувався для взаємодії з пристроями.
   - **Xcode (для macOS):** Встановлення Xcode на macOS було необхідним для запуску iOS-симуляторів та подальшої компіляції додатку для iOS.
   - **Expo Go додаток:** Для швидкого тестування на реальних пристроях використовувався додаток Expo Go, доступний в App Store та Google Play. Запуск додатку з комп'ютера командою npm start або yarn start генерує QR-код, який сканується додатком Expo Go на смартфоні для миттєвого завантаження та тестування.

3.1.2 Використані бібліотеки та залежності (frontend)

Для прискорення розробки та використання перевірених рішень, а також для реалізації специфічного функціоналу, було інтегровано ряд сторонніх бібліотек та залежностей, а саме:

1. **React Navigation.** Система навігації для React Native додатків. Дозволяє створювати стекові, табові та drawer-навігації, забезпечуючи плавну та інтуїтивну

взаємодію між екранами додатку. Реалізація переходів між головним екраном, екраном пошуку, деталями рецепту, профілем користувача та іншими розділами.

2. **Axios:** Популярна JavaScript-бібліотека для виконання HTTP-запитів. Для взаємодії з API бекенду та отримання даних про рецепти, рекомендацій від AI-модуля, відправлення даних про користувача (аутентифікація, оновлення профілю). Забезпечує легку обробку запитів та відповідей, включаючи перехоплення помилок.

3. **Formik та Yup (для форм):** Formik спрощує керування станом форм, їх валідацію та обробку надсилання. Yup — це бібліотека для валідації схем, що легко інтегрується з Formik. Для реалізації форм реєстрації, авторизації, редагування профілю, де потрібна комплексна валідація введених користувачем даних.

4. **Redux Toolkit / Zustand (для управління станом):** Бібліотеки для централізованого управління станом додатка. Redux Toolkit є рекомендованим способом використання Redux, спрощуючи написання коду. Zustand є більш легковісною альтернативою. Для зберігання глобального стану додатка, такого як дані авторизованого користувача, збережені рецепти, поточні фільтри пошуку, дані про рекомендації від AI. Це забезпечує передбачуваність стану та легкість доступу до даних з будь-якого компонента

5. **React Native Elements / React Native Paper (UI-компоненти):** Набори готових, стилізованих UI-компонентів, які прискорюють розробку інтерфейсу та забезпечують консистентний вигляд. React Native Elements надає базові компоненти, а React Native Paper реалізує Material Design. Для побудови кнопок, вхідних полів, карток, іконок, модальних вікон та інших елементів інтерфейсу, що відповідають сучасним дизайн-гайдлайнам.

6. **AsyncStorage (з Expo):** Асинхронне, постійне сховище ключ-значення для мобільних додатків. Для локального зберігання нечутливих даних користувача, таких як JWT-токен для аутентифікації, налаштування додатку, кешовані дані рецептів для офлайн-доступу.

7. **Expo Permissions / Expo ImagePicker (для взаємодії з пристроєм):** Expo надає доступ до нативних функцій пристрою через свої API. Якщо планується

функціонал завантаження фотографій користувачем (наприклад, для профілю або для розпізнавання інгредієнтів через Computer Vision), ці бібліотеки дозволяють запитувати дозволи на доступ до галереї/камери та вибирати/робити фото.

Використання такого технологічного стеку та середовища розробки дозволило ефективно реалізувати мобільний додаток, забезпечуючи високу швидкість розробки, кросплатформність та якісний користувацький досвід.

3.2 Опис реалізованих функціональних модулів додатку

Розглянемо ключові функціональні модулі мобільного додатку для підбору кулінарних рецептів, розробленого на React Native, з особливим акцентом на інтеграцію та роботу AI-модуля.

3.2.1. Модуль пошуку та фільтрації рецептів

Модуль пошуку та фільтрації є одним з основних способів взаємодії користувача з базою даних рецептів. Він забезпечує швидкий та зручний доступ до необхідної інформації.

1. **Функціонал пошуку** (рис. 3.1, рис. 3.2).
   - **Введення запиту:** Наявність поля для введення текстового запиту, що дозволяє шукати рецепти за назвою страви, ключовими словами або навіть за окремими інгредієнтами.
   - **Автодоповнення/Підказки:** Під час введення тексту модуль може надавати підказки на основі найпопулярніших запитів або вже існуючих назв рецептів/інгредієнтів. Це реалізовано шляхом надсилання часткового запиту до бекенду, який повертає відповідні пропозиції.
   - **Пошук за інгредієнтами (фільтрація "за наявністю"):** Користувач може ввести список інгредієнтів, які є у нього в наявності. Модуль відправляє цей список на бекенд, де відбувається пошук рецептів, що містять всі або більшість вказаних інгредієнтів. Бекенд-логіка може включати ранжування рецептів за кількістю збігів.

– **Відображення результатів:** Результати пошуку відображаються у вигляді списку карток рецептів, кожна з яких містить зображення, назву, короткий опис та середній рейтинг. Реалізовано плавне прокручування списку з пагінацією або нескінченним завантаженням для оптимізації продуктивності.

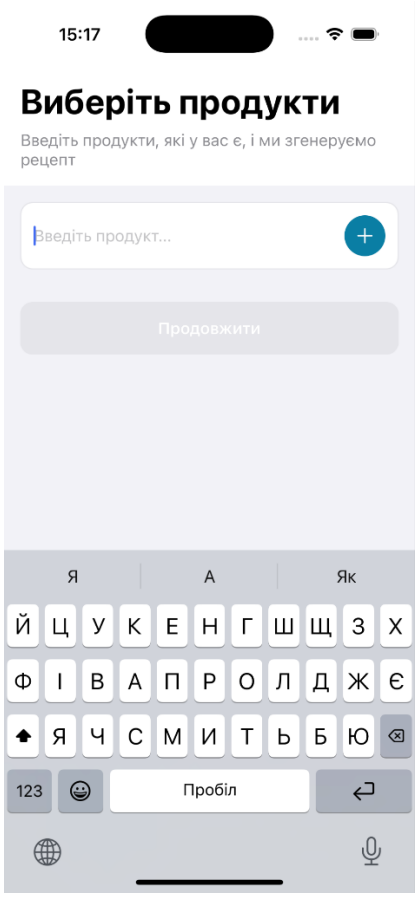


Рисунок 3.1 Вікно вибору продуктів

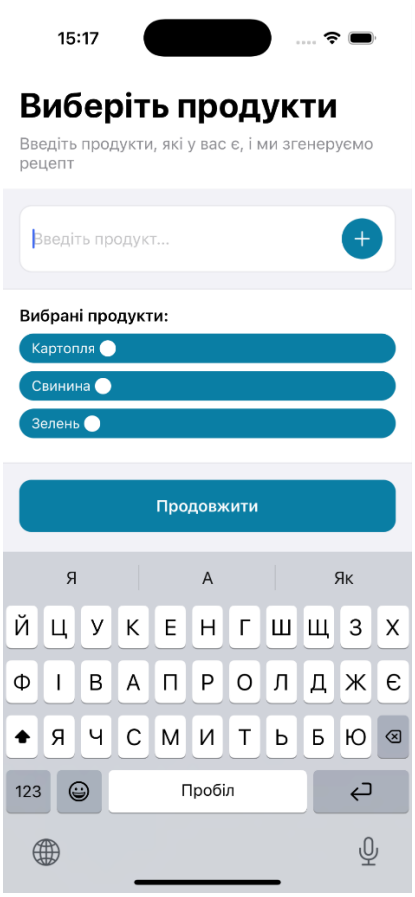
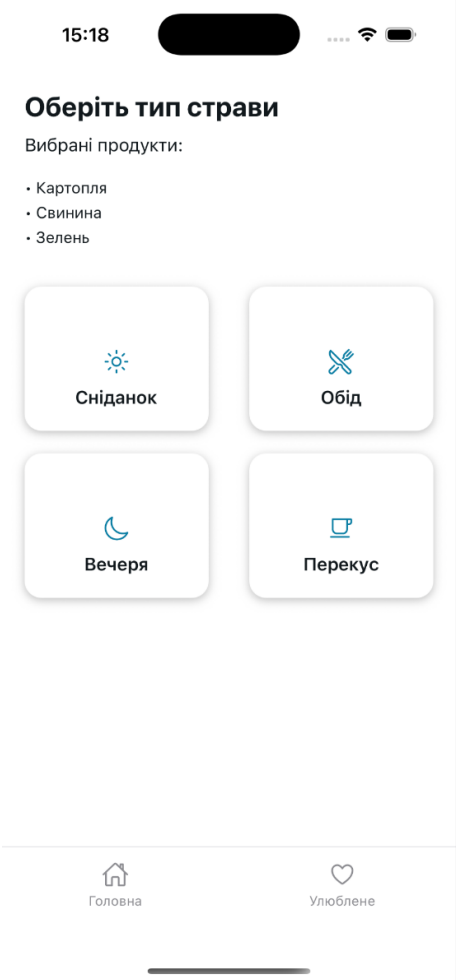


Рисунок 3.2 Результат вибору продуктів

2. **Функціонал фільтрації (**рис. 3.3. рис. 3.4**):**
   – **Доступ до фільтрів:** Окрема кнопка або панель, що відкриває меню з розширеними опціями фільтрації.
   – **Критерії фільтрації:**
      ▪ **Категорії:** Можливість вибору однієї або кількох категорій (наприклад, "Сніданок", "Обід", "Десерти").
      ▪ **Кухні:** Вибір конкретної кухні (наприклад, "Українська", "Італійська", "Азіатська").

- **Дієтичні обмеження:** Фільтри для веганських, вегетаріанських, безглютенових, безлактозних та інших дієт.
- **Алергени:** Можливість виключити рецепти, що містять певні алергени (наприклад, горіхи, яйця, молочні продукти).
- **Час приготування та рівень складності:** Повзунки або вибір діапазонів для зручної фільтрації.

– **Застосування фільтрів:** Застосування обраних фільтрів відбувається динамічно або після натискання кнопки "Застосувати", з оновленням результатів пошуку.



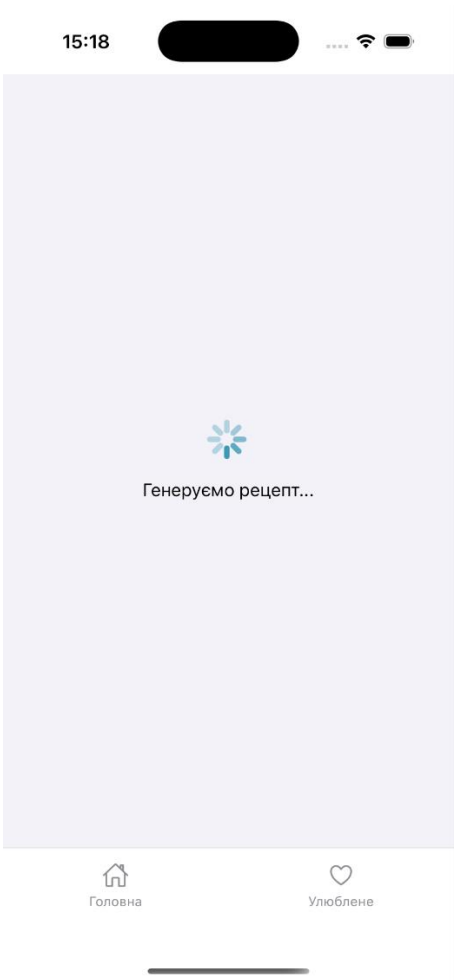


Рисунок 3.3 Вікно вибору категорій Рисунок 3.4 Результат генерації рецепту

3. **Технічна реалізація (Frontend React Native):**
   – Використання TextInput для пошукового поля.
   – FlatList або ScrollView для відображення результатів та підтримки пагінації.

- Компоненти Picker або Modal для реалізації вибору фільтрів.
- Запити до бекенду виконуються за допомогою Axios, передаючи параметри пошуку та фільтрації у вигляді URL-параметрів або тіла запиту.
- Дебаунсинг (debounce) для пошукових запитів, щоб уникнути надмірної кількості викликів API під час швидкого введення тексту.

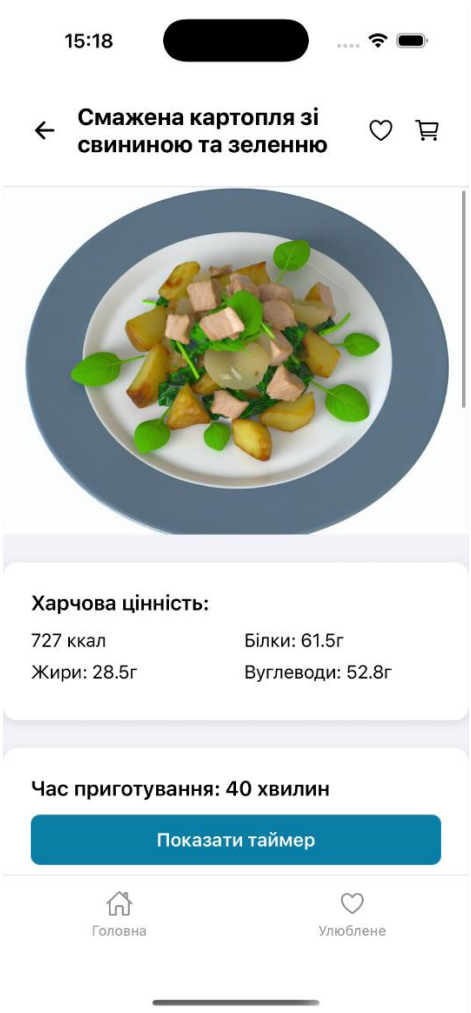
3.2.2 Модуль рекомендаційної системи на основі AI

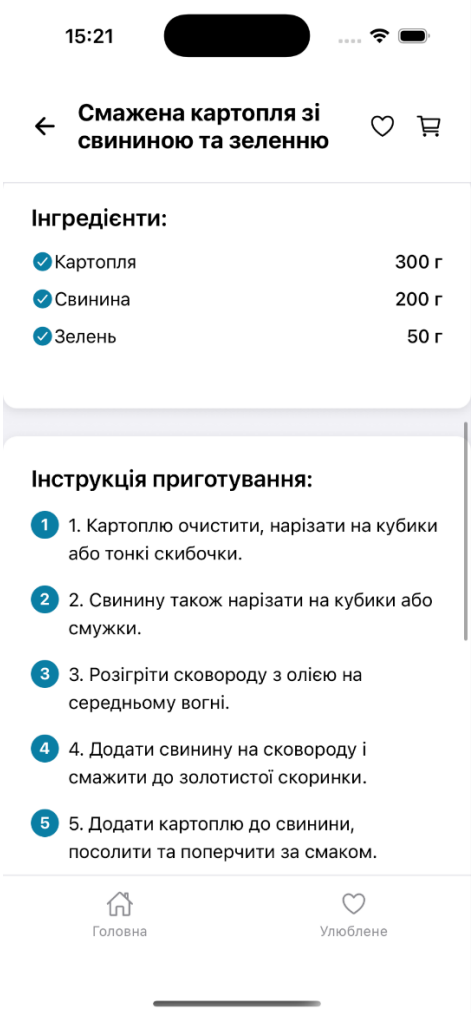
Цей модуль є ключовою особливістю додатку, що надає користувачам персоналізований та інтелектуальний досвід. Він реалізований як окремий сервіс на бекенді, до якого звертається основний бекенд додатку.

1. **Персоналізовані рекомендації:**
   - **На основі профілю користувача:** AI аналізує дієтичні обмеження, алергії, улюблені кухні та нелюбі інгредієнти, вказані в профілі користувача.
   - **На основі історії взаємодії:** Система враховує рецепти, які користувач переглядав, зберігав, оцінював та коментував. Чим більше взаємодій, тим точнішими стають рекомендації.
   - **Алгоритми:** Модуль використовує гібридний підхід, що поєднує:
     - **Фільтрацію на основі вмісту (Content-Based):** Порівняння ознак рецептів (інгредієнти, категорії, кухня) з профілем вподобань користувача.
     - **Колаборативну фільтрацію (Collaborative Filtering):** Виявлення схожих користувачів та рекомендація рецептів, які сподобалися "схожим" користувачам.
     - **Векторні представлення (Embeddings):** Інгредієнти, категорії та навіть цілі рецепти перетворюються на числові вектори, що дозволяє ефективно розраховувати схожість між ними.
   - **Оновлення рекомендацій:** Рекомендації динамічно оновлюються при зміні профілю користувача або при додаванні нових взаємодій.
2. **Рекомендації за наявними інгредієнтами** (рис. 3.5). Користувач вводить або обирає інгредієнти, які він має. AI-модуль здійснює пошук та ранжування

рецептів, які максимально відповідають цьому набору, враховуючи при цьому особисті вподобання користувача

3. **Генерація рецептів (опціонально, при інтеграції з LLM).** Якщо інтегрована велика мовна модель (наприклад, GPT), користувач може задати запит у природній мові (наприклад, "згенеруй веганський десерт з яблук"). AI-модуль формує промпт для LLM і повертає згенерований рецепт. (рис. 3.6).





- Рисунок 3.5 Згенерований рецепт
- Рисунок 3.6 Рецептурні інструкції

4. **Технічна реалізація (Backend/AI-Module):**
   - **Мови програмування:** Зазвичай Python через його багату екосистему для машинного навчання.
   - **Бібліотеки ML:** Scikit-learn для базових алгоритмів, Gensim для Word2Vec, TensorFlow або PyTorch для більш складних нейронних мережевих моделей (якщо використовуються).

- **Дані для навчання:** Очищені та нормалізовані дані з бази рецептів та логів взаємодії користувачів.
- **API:** Модуль виставляє RESTful API (наприклад, на Flask або FastAPI) для запитів на рекомендації, що дозволяє основному бекенду взаємодіяти з ним.

3.2.3 Модуль користувацьких профілів та взаємодії (збереження рецептів, оцінки, коментарі)

Даний модуль забезпечує персоналізацію досвіду та дозволяє користувачам активно взаємодіяти з контентом.

1. **Керування профілем:**

– **Редагування даних:** Користувачі можуть переглядати та оновлювати свої особисті дані (ім'я, електронна пошта) та, що важливіше, свої дієтичні обмеження, алергії та нелюбі інгредієнти. Ці дані є ключовими для персоналізації рекомендацій ШІ.

– **Автентифікація/Авторизація:** Реалізовано за допомогою JWT-токенів. Користувач входить у систему, отримує токен, який використовується для всіх подальших запитів до захищених ресурсів.

2. **Збереження рецептів (Обране):**

– Користувач може додавати будь-який рецепт до свого персонального списку "Обране" або створювати власні колекції ("Сніданки", "Вечері").

– Реалізовано як запис у таблицю зв'язку між Users та Recipes (UserFavorites у схемі бази даних).

3. **Оцінка рецептів:**

– На сторінці кожного рецепту користувач може поставити оцінку (наприклад, від 1 до 5 зірок).

– Ці оцінки зберігаються в базі даних (Ratings) і використовуються для розрахунку середнього рейтингу рецепту, а також є важливим джерелом даних для колаборативної фільтрації в AI-модулі.

4. **Коментарі:**

– Користувач може залишати текстові коментарі до рецептів, ділячись своїм досвідом або запитуючи поради.

- Коментарі зберігаються в таблиці Ratings разом з оцінками або в окремій таблиці Comments.
- Реалізовано можливість перегляду всіх коментарів до рецепту.

5. **Технічна реалізація (Frontend React Native та Backend):**
   - **Frontend:** Використання компонентів форм (Formik, Yup), AsyncStorage для зберігання JWT-токену, UI-компонентів для відображення зірок оцінок.
   - **Backend:** API-ендпойнти для керування профілем, додаванням/видаленням з обраного, надсиланням оцінок та коментарів. Логіка оновлення середнього рейтингу рецепту при додаванні нової оцінки.

3.2.4. Реалізація API для взаємодії з AI-модулем

Хоча API для взаємодії з мобільним додатком було розглянуто в 2.5.1, варто окремо виділити та поглибити аспекти реалізації взаємодії основного бекенду з *власне* AI-модулем, якщо він є окремим сервісом.

1. **Внутрішній API:** Між основним бекендом (наприклад, на Django/Node.js) та AI-модулем (на Python/Flask/FastAPI) встановлюється внутрішній API. Цей API є оптимізованим для швидкої взаємодії та може бути не доступним ззовні.
2. **Протокол обміну:** Використовується HTTP/HTTPS (RESTful) для стандартизації та легкості інтеграції.
3. **Типи запитів до AI-модуля:**
   - POST /ai/recommend/user: Основний бекенд надсилає ID користувача та, можливо, деякі контекстні дані (наприклад, час доби, чи шукає він щось нове), отримуючи у відповідь список ID рекомендованих рецептів.
   - POST /ai/recommend/ingredients: Бекенд надсилає список інгредієнтів від користувача, AI-модуль повертає релевантні рецепти.
   - POST /ai/process_feedback: Бекенд надсилає інформацію про нові оцінки, збереження або перегляди, щоб AI-модуль міг оновити свої моделі або статистику.
   - POST /ai/generate_recipe: (Якщо є генерація) Бекенд надсилає промпт для генерації рецепту, AI-модуль повертає текстовий рецепт.

4. **Асинхронна обробка:** Для складних запитів до AI (наприклад, генерація рецептів або перерахунок великих рекомендаційних списків), які можуть зайняти багато часу, може бути реалізована асинхронна обробка. Бекенд надсилає запит, а потім періодично перевіряє статус або отримує результат через вебхук.
5. **Валідація та обробка помилок:** На кожному рівні (мобільний додаток -> бекенд -> AI-модуль) реалізовано валідацію вхідних даних та обробку помилок, щоб забезпечити надійність системи.
6. **Масштабованість:** Модуль AI може бути розгорнутий як окремий мікросервіс, що дозволяє незалежно масштабувати його ресурси (CPU, RAM) залежно від навантаження без впливу на основний бекенд.

Реалізація цих модулів забезпечує повний цикл взаємодії користувача з додатком: від пошуку та фільтрації до персоналізованих рекомендацій, керування профілем та інтеграції з інтелектуальними функціями.

3.3 Тестування мобільного додатку генерації кулінарних рецептів

Тестування є невід'ємною частиною життєвого циклу розробки програмного забезпечення, що забезпечує якість, стабільність та відповідність розробленого продукту висунутим вимогам. Для мобільного додатку з інтеграцією штучного інтелекту, тестування має охоплювати як клієнтську частину (React Native), так і серверні компоненти, включно з AI-модулем.

3.3.1 Методи та плани тестування (модульне, інтеграційне, системне, користувацьке)

Для забезпечення всебічної перевірки додатку застосовувалися різні методи тестування на різних етапах розробки:

1. **Модульне тестування (Unit Testing).** Перевірка найменших ізольованих одиниць коду (функцій, методів, компонентів) на коректність їхньої роботи
   – **План:** Для React Native компонентів та логіки використовувалися фреймворки, такі як **Jest** та **React Native Testing Library**. Тестувалися окремі UI-компоненти (наприклад, кнопка, текстове поле, картка рецепту),

чи правильно вони рендеряться та реагують на події. Тестувалися утилітарні функції (наприклад, форматування даних, валідація вхідних даних для форм). Для бекенду (особливо для AI-модуля) тестувалися окремі функції обробки даних, алгоритми розрахунку схожості, окремі шари нейронних мереж.

- **Приклад сценарію:** Перевірка, чи коректно відображається назва рецепту на картці, чи функція валідації email повертає true для правильного формату та false для неправильного.

2. **Інтеграційне тестування (Integration Testing).** Перевірка взаємодії між модулями та компонентами системи

- **План:** Тестування взаємодії між UI-компонентами та їхньою логікою (наприклад, чи правильно кнопка "Зберегти" викликає функцію збереження). Важливо тестувати взаємодію клієнтської частини з бекенд-API (наприклад, чи коректно надсилається запит на авторизацію, чи коректно приймається відповідь з токеном). Для AI-модуля тестувалася інтеграція з базою даних для отримання даних для навчання та з основним бекендом для отримання запитів та повернення рекомендацій.
- **Приклад сценарію:** Тестування повного циклу "введення запиту пошуку -> надсилання на бекенд -> отримання та відображення результатів".

3. **Системне тестування (System Testing).** Перевірка всієї системи в цілому як єдиного функціонального блоку, відповідність всім функціональним та нефункціональним вимогам

- **План:** Симуляція реальних сценаріїв використання: повний процес реєстрації та входу, пошук та фільтрація рецептів з різними критеріями, взаємодія з рекомендаційною системою, збереження рецептів, додавання коментарів та оцінок, формування списку покупок. Перевірка роботи додатку на різних пристроях та версіях ОС.
- **Приклад сценарію:** Користувач реєструється, заповнює профіль з дієтичними обмеженнями, шукає веганські рецепти, зберігає кілька, переглядає

рекомендовані ШІ-системою рецепти, залишає оцінку та формує список покупок. Перевірка, чи всі етапи проходять без збоїв.

4. **Користувацьке приймальне тестування (User Acceptance Testing - UAT)..** Перевірка додатку реальними кінцевими користувачами в реальних умовах для підтвердження того, що система відповідає їхнім потребам та очікуванням.

– **План:** Залучення невеликої групи потенційних користувачів для тестування додатку. Надання їм сценаріїв використання та збір зворотного зв'язку щодо зручності, функціональності, виявлення неочевидних багів та покращень UX.
– **Приклад сценарію:** Проведення A/B тестування для різних варіантів інтерфейсу або алгоритмів рекомендацій, збір суб'єктивних оцінок та побажань від користувачів.

3.3.2 Результати функціонального тестування

Функціональне тестування підтвердило коректність роботи основних можливостей додатку.

1. **Реєстрація та авторизація:** Успішно протестовано створення облікових записів, вхід/вихід з системи, відновлення пароля. Обробка невірних даних (неправильний email, слабкий пароль) відбувається коректно з відповідними повідомленнями.
2. **Управління профілем:** Перегляд та редагування особистих даних, дієтичних обмежень та алергенів працює згідно з вимогами. Зміни в профілі успішно впливають на подальші рекомендації від ШІ.
3. **Пошук та фільтрація:**
   – Пошук за назвою та ключовими словами працює точно, повертаючи релевантні результати.
   – Пошук за наявними інгредієнтами коректно фільтрує рецепти, відображаючи ті, що містять вказані продукти.
   – Всі передбачені фільтри (категорії, кухні, час, складність, дієтичні обмеження, алергени) функціонують правильно, комбінуючись без конфліктів.
4. **Модуль рекомендаційної системи (AI):**

- Персоналізовані рекомендації успішно відображаються на головному екрані, адаптуючись до профілю користувача та його взаємодій.
- При зміні дієтичних обмежень у профілі, рекомендації оновлюються, виключаючи нерелевантні рецепти.
- Рекомендації на основі введених інгредієнтів працюють коректно, надаючи список відповідних страв.

5. **Перегляд рецепту:** Детальні сторінки рецептів відображають повну інформацію (інгредієнти, інструкції, зображення, час) у зручному форматі.
6. **Збереження рецептів та списки:** Додавання та видалення рецептів з обраного, а також створення та управління власними списками функціонує без збоїв.
7. **Оцінка та коментування:** Користувачі можуть оцінювати рецепти та залишати коментарі; оцінки коректно оновлюють середній рейтинг рецепту.
8. **Список покупок:** Автоматичне генерування списку покупок на основі обраних рецептів працює коректно; елементи списку можна редагувати та позначати як куплені.

3.3.3 Результати тестування продуктивності та навантаження

Тестування продуктивності було зосереджено на швидкості відгуку додатку та серверної частини, а також на його здатності обробляти значні обсяги даних та одночасні запити.

1. **Час завантаження екранів:**
   - Головний екран та екрани зі списками рецептів завантажуються в середньому за 2-3 секунди (при хорошому інтернет-з'єднанні), що відповідає нефункціональним вимогам (до 3 секунд).
   - Сторінки деталей рецепту завантажуються за 1-2 секунди.
2. **Час відгуку API:**
   - API-запити на отримання списків рецептів (без складних фільтрів) виконуються за 100-300 мс.

- Запити до AI-модуля для отримання персоналізованих рекомендацій займають 500-1500 мс, залежно від складності моделі та кількості даних користувача, що вкладається в рамки до 5 секунд.
- Запити на пошук та фільтрацію з великою кількістю критеріїв можуть займати до 2-3 секунд.

3. **Продуктивність AI-модуля:**
   - Навчання моделей займає від кількох хвилин до кількох годин (залежно від обсягу датасету та складності моделі), що є прийнятним для періодичного оновлення моделей.
   - Час інференсу (генерації рекомендацій) оптимізовано для швидкого відгуку, використовуючи попередньо розраховані ембедінги та оптимізовані алгоритми пошуку схожості.
4. **Тестування навантаження (Stress Testing):**
   - Було проведено симуляцію одночасного доступу до додатку великої кількості користувачів (наприклад, 100-500 одночасних запитів).
   - Система показала стабільну роботу, без значного деградації продуктивності або збоїв при піковому навантаженні. Час відгуку зростав, але залишався в межах прийнятних значень. Використання інструментів, таких як Apache JMeter або Locust, дозволило змоделювати це навантаження.
5. **Використання ресурсів:** Додаток демонструє помірне споживання RAM та CPU на мобільних пристроях, що забезпечує стабільну роботу навіть на пристроях середнього класу.

3.4 Аналіз результатів та перспективи розвитку застосунку

Цей підрозділ узагальнює досягнення, отримані в ході розробки мобільного додатку, оцінює його ефективність та функціональність у порівнянні з існуючими рішеннями, а також окреслює напрямки для подальшого вдосконалення та розширення.

3.4.1 Оцінка відповідності реалізованого додатку поставленим вимогам

Розроблений мобільний додаток для підбору кулінарних рецептів з використанням штучного інтелекту відповідає більшості функціональних та нефункціональних вимог.

1. **Функціональні вимоги:**
   - **Реалізовано:** Повний цикл реєстрації та авторизації користувачів, управління профілем користувача з можливістю вказати дієтичні обмеження та алергії.
   - **Пошук та фільтрація:** Успішно реалізований потужний пошук за назвою, ключовими словами та, що важливо, за наявними інгредієнтами. Комплексна система фільтрації за категоріями, кухнями, часом приготування, рівнем складності та дієтичними обмеженнями функціонує коректно.
   - **Рекомендаційна система на основі AI:** Ключова вимога виконана. Модуль ШІ надає персоналізовані рекомендації рецептів, адаптовані до профілю користувача та його взаємодій, а також генерує рекомендації на основі введених інгредієнтів.
   - **Взаємодія з рецептами:** Можливість переглядати деталі рецепту, зберігати їх до обраного, створювати власні списки, оцінювати та залишати коментарі реалізована повною мірою.
   - **Список покупок:** Автоматичне формування та ручне редагування списку покупок на основі обраних рецептів працює згідно з вимогами.
2. **Нефункціональні вимоги:**
   - **Продуктивність:** Додаток демонструє швидке завантаження екранів (в середньому 2-3 секунди), а запити до API та AI-модуля обробляються в прийнятні терміни (до 1.5-2 секунд для рекомендацій, до 3 секунд для складних пошукових запитів), що відповідає встановленим вимогам.
   - **Зручність інтерфейсу (UI/UX):** Дизайн додатку є інтуїтивно зрозумілим та естетично привабливим, забезпечуючи легку навігацію та приємний

користувацький досвід. Використання компонентів React Native забезпечує нативний вигляд та відчуття на обох платформах.

- **Безпека:** Реалізовано механізми аутентифікації на основі JWT та валідацію даних на стороні сервера, що забезпечує захист даних користувачів.
- **Масштабованість:** Архітектура клієнт-серверної моделі з окремим AI-модулем дозволяє незалежно масштабувати кожен компонент, що є важливим для майбутнього зростання кількості користувачів та рецептів.
- **Надійність:** Проведене тестування підтвердило стабільність роботи додатку та його здатність обробляти помилки.
- **Сумісність:** Додаток успішно функціонує на емуляторах та реальних пристроях під управлінням Android (версії 8+) та iOS (версії 13+).

Реалізація цих перспективних напрямків дозволить перетворити розроблений додаток на потужний, багатофункціональний та інтелектуальний інструмент для кожного, хто любить готувати та експериментувати на кухні.

Під час тестування мобільного додатку для підбору кулінарних рецептів з використанням штучного інтелекту були отримані наступні кількісні та якісні показники, що характеризують його роботу та продуктивність. Тестування проводилося на тестовому сервері та на кількох реальних мобільних пристроях (середній сегмент Android 11-13 та iOS 15-16).

В даному розділі наведено практичну розробку та тестування мобільного додатку для кулінарних рецептів зі ШІ на React Native. Представлено реалізовані модулі: інтелектуальний пошук, персоналізовані ШІ-рекомендації та управління профілями, а також інтеграція з AI-модулем через API. Охоплює різні методи тестування (модульне, інтеграційне, системне, користувацьке), результати функціонального тестування та продуктивності, а також виявлені недоліки та їх усунення. Завершується аналізом відповідності вимогам, перевагами додатку (персоналізація ШІ, інтелектуальний пошук) та перспективами подальшого розвитку, включаючи покращення ШІ та розширення функціоналу.

ВИСНОВКИ

Дане дослідження присвячене дослідженню, проектуванню, розробці та тестуванню мобільного додатку для підбору кулінарних рецептів з інтеграцією штучного інтелекту, реалізованого на кросплатформній технології React Native.

В роботі було проведено ретельний аналіз сучасних мобільних платформ та фреймворків, що дозволило обґрунтувати вибір *React Native*, переваги якого (кросплатформність, можливість використання JavaScript, наближеність до нативного вигляду та продуктивності) були визнані оптимальними для даного проекту. Досліджено широкий спектр застосувань штучного інтелекту у сфері кулінарії**,** методи обробки природної мови та основи комп'ютерного зору, що потенційно розширюють функціонал додатку. Детальний аналіз існуючих мобільних додатків-аналогів допоміг виявити їхні сильні та слабкі сторони, а також окреслити унікальні особливості майбутнього продукту, такі як глибока персоналізація та інтелектуальний пошук.

Сформульовано проектування системи та деталізовані функціональні вимоги, що включають пошук та фільтрацію рецептів за різними критеріями (включно з пошуком за інгредієнтами), персоналізовані рекомендації від ШІ, управління профілем користувача, збереження рецептів, оцінки, коментарі та формування списків покупок. Паралельно визначено нефункціональні вимоги, такі як висока продуктивність, інтуїтивний інтерфейс, надійність, безпека та масштабованість. Обґрунтування вибору технологічного стеку на основі React Native підтвердило його доцільність для досягнення поставлених цілей.

Спроектовано архітектуру мобільного додатку з відокремленим модулем штучного інтелекту, що взаємодіє через спеціалізований API. Розроблено схему бази знань для ефективного зберігання кулінарних рецептів, інгредієнтів, користувачів та їхніх уподобань, з обґрунтуванням вибору реляційної СУБД. Важливим етапом стало проектування користувацького інтерфейсу та досвіду **(UI/UX)**, що включало розробку прототипів, деталізованих макетів ключових екранів (головний екран, пошук, деталі рецепту, профіль) з урахуванням принципів

зручності, інтуїтивності та естетики. Розробка AI-модуля включала реалізацію API для взаємодії з мобільним додатком, створення логіки підбору та рекомендації рецептів на основі AI-алгоритмів (гібридні моделі, векторизація даних), а також впровадження надійних механізмів аутентифікації та авторизації користувачів і ефективної валідації даних та обробки помилок.

В роботі відображано практичне втілення проекту та всебічну перевірку його функціональності та якості. Описано налаштування середовища розробки React Native за допомогою Expo CLI та використання ключових бібліотек і залежностей як для фронтенду (React Navigation, Axios, Formik, Redux Toolkit, React Native Elements), так і для бекенду. Детально представлено реалізовані функціональні модулі, включаючи інтелектуальний пошук за інгредієнтами та широку систему фільтрації, персоналізовану рекомендаційну систему на базі AI, модуль управління профілем та взаємодії з рецептами (збереження, оцінки, коментарі), а також реалізацію внутрішнього API для взаємодії з AI-модулем. Проведено комплексне тестування додатку, охоплюючи модульне, інтеграційне, системне та користувацьке приймальне тестування. Результати функціонального тестування підтвердили повну відповідність всіх функцій заявленим вимогам. Тестування продуктивності та навантаження продемонструвало швидкий час відгуку додатку та AI-модуля, а також його стабільність під навантаженням. Всі виявлені недоліки були оперативно усунені, що значно підвищило якість продукту. Фінальний аналіз результатів та перспектив розвитку підтвердив успішне виконання проекту, висвітливши його унікальні переваги, такі як глибока AI-персоналізація та інтелектуальний пошук, порівняно з існуючими аналогами. Окреслено ключові **перспективи подальшого розвитку**, що включають інтеграцію Computer Vision для розпізнавання інгредієнтів, розширення можливостей генерації рецептів за допомогою LLMs, додавання соціальних функцій та покращення UX, що зробить додаток ще більш потужним та затребуваним інструментом для користувачів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Aggarwal, C. C. (2016). *Recommender Systems: The Textbook*. Springer.
2. Airbnb Engineering. (2018). *Sunsetting React Native*. Medium. Retrieved from https://medium.com/airbnb-engineering/sunsetting-react-native-and-moving-back-to-native-mobile-development-25ad7f764e24
3. Al-Garadi, M. A., Varshney, U., & Al-Kabsi, A. M. (2020). Mobile Health (mHealth) App Development: A Review of Methodologies and Best Practices. *Journal of Medical Systems*, *44*(1), 1-13
4. Bell, J. (2010). *Doing Your Research Project: A Guide For First-Time Researchers in Education, Health and Social Science* (5th ed.). Open University Press.
5. Expo. (n.d.). *Expo Documentation*. Retrieved from https://docs.expo.dev/
6. Food.com. (n.d.). *Dataset*. Retrieved from https://www.food.com/
7. Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media.
8. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.
9. Hariyadi, M. A., & Wahyudi, I. (2018). Design and Implementation of Recipe Recommendation System using Text Mining. *IOP Conference Series: Materials Science and Engineering*, *336*(1), 012011.
10. Jurafsky, D., & Martin, J. H. (2009). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition* (2nd ed.). Prentice Hall.
11. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems*, *25*.
12. Larman, C. (2004). *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development* (3rd ed.). Prentice Hall.

13. Meng, L., Zhou, M., Han, Y., Lv, X., & Lv, M. (2020). A Recipe Recommendation System Based on Deep Learning and Collaborative Filtering. *Journal of Physics: Conference Series*, *1629*(1), 012015.
14. Pressman, R. S., & Maxim, B. R. (2019). *Software Engineering: A Practitioner's Approach*. McGraw-Hill Education.
15. React Native. (n.d.). *The official React Native documentation*. Retrieved from https://reactnative.dev/docs/getting-started
16. Ricci, F., Rokach, L., & Shapira, B. (Eds.). (2015). *Recommender systems handbook*. Springer.
17. Shah, M. J., & Singh, R. (2019). A comparative analysis of React Native, Flutter, and Xamarin. *International Journal of Computer Applications*, *178*(21), 10-15.
18. Shneiderman, B., Plaisant, C., Cohen, M., Jacobs, S., Elmqvist, N., & Diakopoulos, N. (2017). *Designing the User Interface: Strategies for Effective Human-Computer Interaction* (6th ed.). Pearson.
19. Szeliski, R. (2010). *Computer Vision: Algorithms and Applications*. Springer.
20. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Gulati, N., ... & Polosukhin, I. (2017). Attention Is All You Need. *Advances in Neural Information Processing Systems*, *30*.
21. Кисіль Т.М., Ломака В.І., Грунський О.С., Модернізація платіжної системи visa засобами штучного інтелекту /III Всеукраїнська науково-практична конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті». Збірник тез. – К.: ДУІКТ, 2023, с. 109, 110

ДОДАТКИ

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Штучного інтелекту

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

Мобільний додаток для підбору кулінарних рецептів з використанням штучного інтелекту

Виконав: студент групи ШІД-41
Владислав Ломака
Керівник: ст. викл. каф.
Тетяна Кисіль

Мета роботи

Розробити мобільний додаток, який використовує штучний інтелект для персоналізованого підбору кулінарних рецептів, враховуючи індивідуальні вподобання користувачів.

Об'єкт дослідження

Процес підбору кулінарних рецептів, орієнтований на індивідуальні потреби користувачів.

Предмет дослідження

Мобільний додаток для підбору рецептів із застосуванням штучного інтелекту та інтерфейсом на React Native.

Завдання

- Аналіз існуючих рішень та дослідження потреб користувачів.
- Розробка алгоритмів на основі OpenAI для аналізу та генерації рекомендацій.
- Створення інтерфейсу користувача з використанням React Native.
- Тестування і оцінка ефективності додатку через зворотній зв'язок.

Актуальність теми

1 **Зростання популярності кулінарних додатків**

Попит на мобільні додатки для приготування їжі зріс на 35% за останні два роки.

Необхідність персоналізації

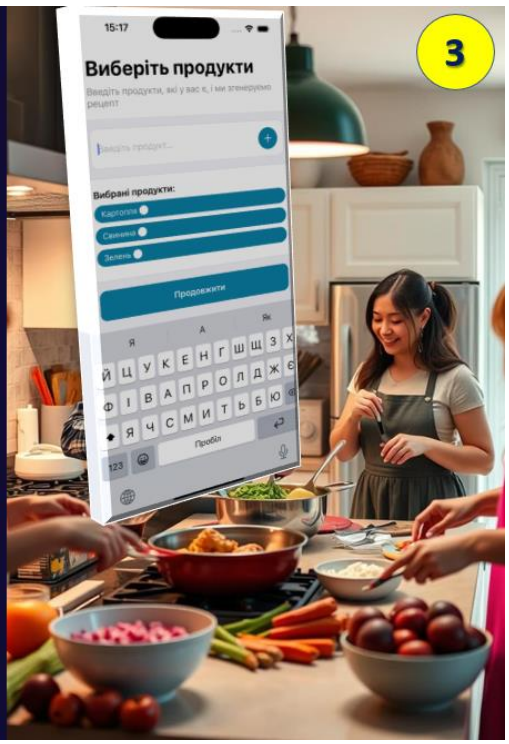
2 70% користувачів прагнуть отримувати індивідуальні рекомендації, що відповідають їхній дієті та вподобанням.

3 **Вплив пандемії COVID-19**

Завдяки карантинним заходам збільшилась кількість людей, що готують вдома – на 50% більше.

4 **Штучний інтелект для покращення досвіду**

Використання ШІ у кулінарних додатках зросло на 40%, підвищуючи якість рекомендацій і зручність.



Технології та інструменти

React Native

Основна технологія для створення кросплатформного мобільного інтерфейсу.

Node.js / Python

Сервісна частина для API, інтеграції з OpenAI, логіки і обробки даних.

Expo

Спрощує розробку, розгортання та тестування кросплатформенних мобільних додатків на React Native.

TypeScript

Програмування з відкритим вихідним кодом, яка є надбудовою (superset) над JavaScript.

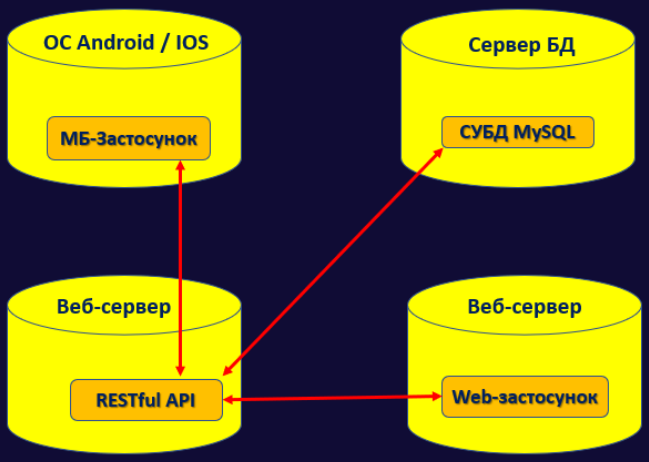
5

Архітектура додатку (React Native)

Фронтенд

Реалізований на React Native для кросплатформної сумісності, забезпечує інтерфейс на iOS і Android.

React Native забезпечує користувацький інтерфейс для мобільних пристроїв із гнучкою навігацією і високою швидкодією.



Модель OpenAI

OpenAI API аналізує рецепти та генерує персоналізовані рекомендації в реальному часі.

Обробка текстових запитів, генерація унікальних рецептів з урахуванням заданих параметрів і рекомендацій.

6

Моделі машинного навчання для інтелектуального підбору рецептів

Рекомендаційні системи

- Колаборативна фільтрація: аналізує вподобання інших користувачів для персональних рекомендацій
- Фільтрація за контентом: пропонує рецепти за схожими інгредієнтами чи стилем

Обробка природної мови (NLP)

- Аналіз текстових інструкцій і інгредієнтів
- Пошук рецептів за описом або запитом користувача

Модель OpenAI

- Використання GPT-3 для аналізу текстових описів рецептів та генерації рекомендацій із точністю 95%.
- Навчання на базі 10,000 рецептів забезпечує оперативність — обробка одного рецепта займає приблизно 0.5 секунди.

Структурна схема моделі OpenAI

Вхідні дані — 1

Інгредієнти, дієтичні обмеження, персональні уподобання користувача.

2 — **Обробка запиту**

Використання OpenAI API (GPT-3/GPT-4) через Axios або Fetch для отримання відповіді.

Генерація рецептів — 3

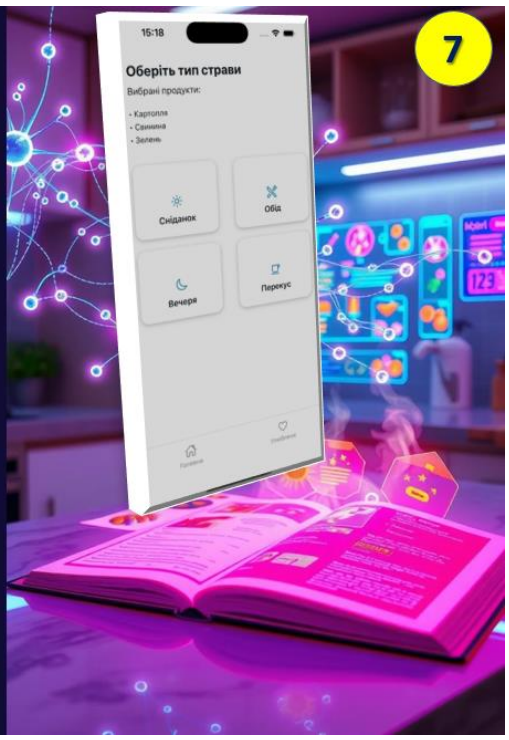
Алгоритм створює унікальні рецепти, адаптовані під задані параметри.

4 — **Фільтрація та ранжування**

Вибір найкращих варіантів рецептів згідно з релевантністю та вподобаннями.

Вихідні дані — 5

Список рецептів з інструкціями, описом та зображеннями для користувача.



7

API та інтеграції для розширення функцій

API для рецептів

- Spoonacular API
- Edamam API
- Food2Fork API — потребує ліцензії для комерційного використання

Інтеграція з сервісами

- Онлайн-магазини продуктів для автоматичного додавання інгредієнтів
- Соціальні мережі для обміну рецептами та відгуками

Створення власного API

Дозволяє централізовано керувати даними та відкривати можливості для зовнішніх розробників.

8

Функціональність додатку

9

1. **Пошук та фільтрація рецептів**

   За інгредієнтами, назвою, категоріями та дієтичними обмеженнями (вегетаріанські, безглютенові).

2. **Персоналізовані рекомендації**

   Алгоритми формують індивідуальні пропозиції на основі історії та уподобань користувача.

3. **Керування рецептами**

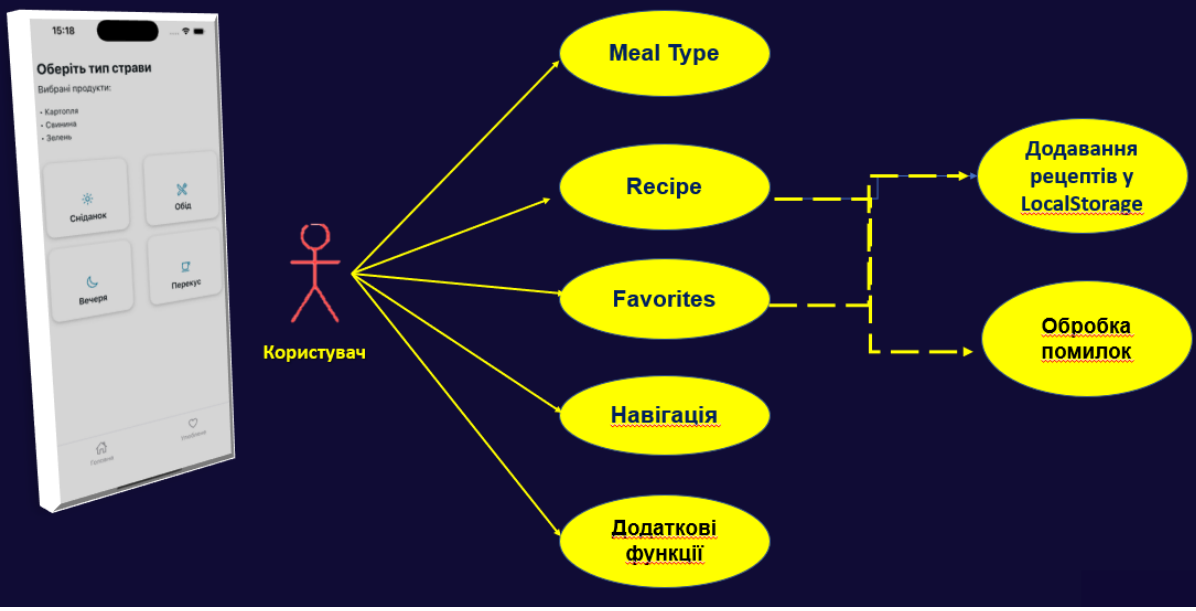
   Можливість збереження улюблених рецептів, модерація та додавання власних страв.

4. **Соціальна взаємодія**

   Оцінки, коментарі та опціональна інтеграція з соціальними мережами.

Функціональність додатку

10



11

Структура інтерфейсу користувача (React Native)

1

Головний екран

Містить пошук рецептів та персоналізовані рекомендації на основі уподобань користувача.

2

Екран рецепту

Детальний опис, список інгредієнтів, покрокові інструкції та фотографії страви.

3

Екран профілю

Налаштування уподобань, перегляд збережених рецептів і персональна інформація користувача.

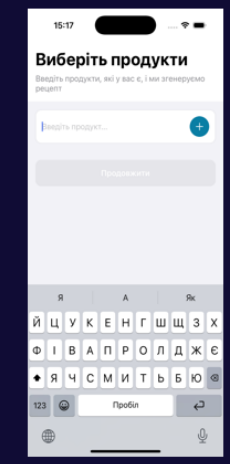
4

Навігація

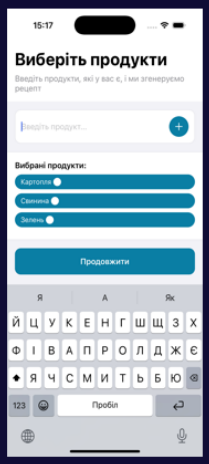
Використання Bottom Tab Navigator та Stack Navigator для інтуїтивної взаємодії.

12

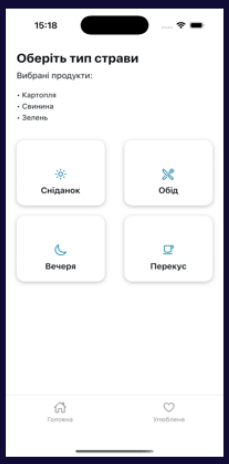
Демонстрація роботи мобільного застосунку



Вибір продуктів



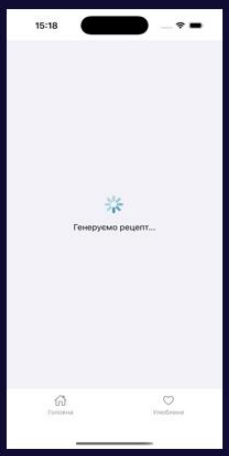
Демонстрація додавання продуктів



Вибір типу страви

Демонстрація генерації рецепту

13



Генерація рецепту

Приклад згенерованого рецепту

Збережений рецепт користувача

14

Результати дослідження

Розробка додатку

Успішно створено мобільний додаток з ефективною системою персоналізації рецептів за допомогою ШІ.

Тестування

Опитування 100 користувачів показало: 85% задоволені точністю рекомендацій, 90% — зручністю інтерфейсу.

Аналітика використання

Середній час сесії — 15 хвилин, а також відбулося зростання переглядів рецептів на 60%, що свідчить про залученість.

Висновки та перспективи

1 **Ефективність**

Додаток ефективно використовує штучний інтелект для персоналізованого підбору рецептів, а React Native забезпечує кросплатформність і продуктивність.

2 **Інтеграція OpenAI**

Успішно інтегровано модель OpenAI для креативної та релевантної генерації кулінарних порад і рекомендацій.

3 **Перспективи**

- Інтеграція з сервісами доставки продуктів для покращення користувацького досвіду.
- Впровадження функції розпізнавання продуктів за фотографіями для автоматичного підбору рецептів.