

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Мобільний додаток "Військова кухня"»

Виконав студент групи ПЗ-20-4
ДУБИНЧЕНКО Богдан Миколайович

Керівник роботи:
САВІЦЬКИЙ Роман Святославович

Рецензент:
ЧИЖМОТРЯ Олексій Володимирович

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Завідувач кафедри інженерії
програмного забезпечення

Тетяна ВАКАЛЮК

«26» лютого 2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ДУБИНЧЕНКО** Богдан Миколайович

Керівник роботи: **САВІЦЬКИЙ** Роман Святославович

Тема роботи: «Додаток "Військова кухня"»,

затверджена Наказом закладу вищої освіти від «23» лютого 2024 р., №74с

Вихідні дані для роботи: Об'єктом дослідження є використання інформаційних технологій для спрощення планування трапез у військових підрозділах та обмін локалізованими рецептами у мережі інтернет.

Предметом дослідження є використання сучасних мобільно-орієнтованих технологій розробки для створення платформи, що допомагає спростити логістику у сфері харчування.

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Савіцький Р.С.	27.02.2024р.	27.02.2024р.
2	Савіцький Р.С.	27.02.2024р.	27.02.2024р.
3	Савіцький Р.С.	27.02.2024р.	27.02.2024р.

РЕФЕРАТ

Випускна робота бакалавра складається з програмного комплексу мобільного додатку з плануванням трапез у військових підрозділах та пояснювальної записки.

Пояснювальна записка до випускної роботи містить текстову частину, викладену на 83 сторінках друкованого тексту, має 6 сторінок додатків. Список використаних джерел містить 35 найменувань і займає 3 сторінки. У роботі наведено 52 ілюстрації та 47 таблиць. Загальний обсяг роботи – 91 сторінок.

Метою роботи є розробка клієнтської та серверної частини додатку планування трапез засобами Node.JS, TypeScript, HTML, CSS із використанням бази даних PostgreSQL.

У роботі визначено та обґрунтовано основні завдання на розробку концептуальної системи. Також було проаналізовано аналоги програмних продуктів. Обґрунтовано використання архітектури додатку. Наведено сценарії роботи програмного комплексу, загальна структура програмного комплексу, опис алгоритмів взаємодії окремих модулів системи.

КЛЮЧОВІ СЛОВА: БАЗА ДАНИХ, МОБІЛЬНИЙ ЗАСТОСУНОК, ПЛАНУВАННЯ, РЕЦЕПТИ, СЕРВЕР.

					ІПЗ.КР.Б – 121 – 24 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Мобільний додаток "Військова кухня"	Літ.	Арк.	Аркушів
Розроб.	Б.М. Дубинченко						3	91
Керівник	Р.С. Савіцький					Житомирська політехніка, група ІПЗ-20-4		
Рецензент	О.В. Чижмотря							
Зав. каф.	Т. А. Вакалюк				Пояснювальна записка			

ABSTRACT

The bachelor's thesis consists of a software package of a mobile application for meal planning in military units and an explanatory note.

The explanatory note to the thesis contains a textual part set out on 83 pages of printed text and has 6 pages of appendices. The list of references includes 35 titles and covers 3 pages. The work contains 52 illustrations and 47 tables. The total volume of the work is 91 pages.

The purpose of the work is to develop the client and server parts of the meal planning application using Node.JS, TypeScript, HTML, CSS using the PostgreSQL database.

The paper defines and substantiates the main tasks for the development of the conceptual system. Analogs of software products were also analyzed. The use of the application architecture is substantiated. The scenarios of the software system operation, the general structure of the software system, and the description of the algorithms of interaction of individual modules of the system are presented.

KEYWORDS: DATABASE, MOBILE APPLICATION, PLANNING, RECIPES, SERVER.

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	7
ВСТУП.....	8
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМИ ПЛАНУВАННЯ ТРАПЕЗ	11
1.1 Опис проекту та постановка задачі	11
1.2 Аналіз аналогів програмного продукту	13
1.3 Вибір архітектури мобільно-орієнтованої системи планування траpez.....	17
1.4 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	22
Висновки до першого розділу.....	24
РОЗДІЛ 2. ПРОЕКТУВАННЯ МОБІЛЬНО-ОРІЄНТОВАНОЇ СИСТЕМИ ПЛАНУВАННЯ ТРАПЕЗ.....	25
2.1 Визначення варіантів використання та об'єктно-орієнтованої структури системи	25
2.2 Розробка бази даних системи.....	34
2.3 Проектування та реалізація алгоритмів роботи системи	44
2.4 Реалізація мобільно-орієнтованої системи планування траpez.....	46
Висновки до другого розділу.....	59
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З МОБІЛЬНО- ОРІЄНТОВАНОЮ СИСТЕМОЮ ПЛАНУВАННЯ ТРАПЕЗ	60
3.1 Порядок встановлення та налаштування параметрів системи	60
3.2 Структура інтерфейсу та порядок взаємодії системи планування траpez.....	61
3.3 Тестування роботи програмного комплексу	76
Висновки до третього розділу	80
ВИСНОВКИ	81

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 82
ДОДАТКИ..... 85

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – Набір правил та інструментів, які дозволяють одному програмному засобу взаємодіяти з іншим

CRUD – create, read, update, delete (з англ. створити, прочитати, оновити, видалити)

CSS – Спеціальна мова стилю сторінок, що використовується для опису їхнього зовнішнього вигляду

HTML – Стандартизована мова розмітки документів для перегляду вебсторінок у браузер

MVC – Модель–вигляд–контролер (або Модель–представлення–контролер, Model-view-controller, MVC) – архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення

RBAC – Role Based Access Control, Керування доступом на основі ролей

SPA – Тип вебзастосунку, в якому весь контент завантажується одноразово при завантаженні сторінки (англ. Single Page Application, SPA)

TCP/IP – Набір протоколів мережі Інтернет

UI – Дизайн інтерфейсу користувача

БД – База даних

ПЗ – Програмне забезпечення

ІІІ – Штучний інтелект

ВСТУП

Актуальність теми. У сучасних реаліях війни, мобільні пристрої та спеціалізовані додатки набули особливого масштабу для військового призначення.

Для військовослужбовців мобільні додатки дозволяють полегшити комунікацію між солдатами або підрозділами, що є критично важливим в умовах бойових дій або на полігоні. Такі технології можна використовувати як засоби для надання звітів, навчання медичній допомозі або обмін повідомленнями та геолокаціями для координації дій.

Військова тематика не закінчується тільки на плануванні тактики або бойових діях. Вона безумовно включає такі побутові речі як харчування, управління ресурсами, відпочинок або розваги, незважаючи на місце перебування службовців.

Наприклад організація їжі в армії вимагає не лише врахування харчових норм, але й планування по днях, з урахуванням енергетичних потреб військового персоналу на різних етапах тренувань та операцій. Кожен день на військовій кухні – це виклик для кухарів і логістів, які повинні забезпечити якісне харчування, щоб забезпечити фізичну та психічну витривалість військовослужбовці. Тому створення зручного мобільного додатку планування трапез є хорошим рішенням для забезпечення підвищення ефективності готування та автоматизації процесів.

Мета даної роботи полягає у розробці клієнтської та серверної частини додатку планування трапез засобами Node.JS, TypeScript, HTML, CSS із використанням бази даних PostgreSQL. Сервіс має забезпечувати функціонал для повної і безперебійної роботи, навіть при відсутності інтернет підключення та взаємодіяти з різними мобільними платформами.

Поставлена мета обумовлює необхідність виконання наступних завдань:

- аналіз теоретичних та функціональних потреб проектування та реалізації мобільного додатку планування трапез у військових підрозділах;
- аналіз існуючих аналогів на обрану тематику, визначення унікальності;
- визначення технологій реалізацій сервісу;
- визначення інформаційних потреб предметної області дослідження;
- обрання оптимальної в сучасних умовах архітектури функціонування програмного комплексу;
- виконання опису методів для взаємодії з системою;
- проектування бази даних, для збереження інформації сервісу та користувачів;
- проектування алгоритмів системи;
- безпосередня реалізація застосунку;
- опис структури інтерфейсу та порядку взаємодії зі системою;
- тестування розробленого додатку;
- розгортання готового продукту в магазині додатків.

Об’єктом дослідження є методи та засоби розробки мобільного застосунку для полегшення процесів роботи планування трапез у військових підрозділах.

Предметом дослідження є моделі, технології, структури, засоби та методи розробки та взаємодії мобільних застосунків та модулів управління мобільних середовищ.

Методи дослідження. У роботі було використано методи об’єктно-орієнтованого проектування та програмування для створення серверної частини застосунку. Для клієнтської частини було використано методи дизайну користувацьких інтерфейсів, функціональне та вебпрограмування.

Практичне значення одержаних результатів. Розроблений додаток може бути використаний на різних мобільних системах та в подальшому за

можливості адаптований для вебзастосунку або використання в цивільному просторі.

Тези до даної кваліфікаційної роботи приведені в CSNT 2024 [1].

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМИ ПЛАНУВАННЯ ТРАПЕЗ

1.1 Опис проекту та постановка задачі

На сьогоднішній день допомога військовим є необхідністю для кожної цивільної людини. Це може виражатися не тільки в матеріальних речах, але і в розвитку різних проектів та стартапів, які можуть забезпечити більш продуктивну та просту роботу військовослужбовців.

Автоматизація таких процесів як приготування їжі – це впровадження різних функцій, які надають стабільну логістику в плануванні трапез на певну кількість днів, містить велику базу даних рецептів для різних раціонів та місць перебування службовців, допомагає зі стратегічними запасами.

Слід зазначити особливості, які потрібно передбачити для отримання якісного кінцевого продукту. Насамперед, розробка сервісу планування трапез у підрозділах має передбачати створення клієнтської та серверної частини, для майбутнього розширення під різні платформи.

Отриманий застосунок повинен бути нативний, що надає користувачу можливість отримувати інформацію незалежно від підключення до мережі та підтримувати різні мобільні платформи.

Програмний продукт повинен мати наступні основні функціональні характеристики:

- авторизація та реєстрація користувачів з можливістю скидання і зміни пароля;
- авторизація та реєстрація користувачів за допомогою Google акаунту;
- відображення рецептів та застосування до їх різноманітних параметрів сортувань та фільтрів;
- пошук рецептів за назвою або переліком інгредієнтів;
- перегляд рецепту, модифікування або додавання його до улюблених;

- додавання рецепту до налаштованого розкладу;
- відображення налаштованих днів розкладу;
- налаштування прийомів їжі;
- створення та редагування своїх рецептів;
- формування списку продуктів по днях;
- генерація порад для приготування та зберігання продукції;
- перегляд та редагування профілю.

Програмний комплекс повинен бути побудований за допомогою клієнт-серверної архітектури з використанням мобільних-технологій, що не вимагають додаткового ліцензування. Серверна частина програмного комплексу має базуватися на Node.JS [2] та системі керування базами даних PostgreSQL [3].

Клієнтська частина має бути побудована за сучасними стандартами створення мобільних застосунків, тому за основу буде взято фреймворк React Native [4]. Він використовує архітектурну структуру, яку іноді називають "JavaScript bridge" або "міст JavaScript". Основна ідея – використання JavaScript [5] для опису компонентів і логіки додатку, яка потім взаємодіє з нативним кодом платформи через "міст".

Розробка системи передбачає наступні основні етапи:

1. Створення бекенд частини додатку шляхом написання приватного API [6], використовуючи фреймворк Nest.JS [7], що включає реалізацію наступних основних модулів:
 - модуль “User”;
 - модуль “Auth”;
 - модуль “Recipe”;
 - модуль “Ingredients”;
 - модуль “Planner”;
 - модуль “Advice”;
 - модуль “MealPlan”.

2. Створення дизайну інтерфейсу, з яким будуть взаємодіяти користувачі в системі:

- обрати бібліотеку для реалізації користувацького інтерфейсу;
- розробити відповідні сторінки за допомогою обраної бібліотеки.

3. Налаштувати отримання даних від створеного API на клієнті та розробити роутинг застосунку за допомогою Expo Router [8]:

- налаштувати реалізацію виклику API за допомогою відповідної бібліотеки;
- налаштувати роутинг додатку за допомогою відповідної бібліотеки.

Результатом виконання цих етапів є створення мобільної системи, за допомогою серверних та нативних архітектурних підходів, яка надасть стабільну та зручну роботу користувачам для полегшення логістики планування трапез.

1.2 Аналіз аналогів програмного продукту

В наш час важко вигадати щось абсолютно нове, оскільки багато сфер життя вже насичені інноваціями. Тому під час створення додатків важливо ретельно аналізувати існуючі аналоги, зазначаючи їхні переваги та недоліки. Це дозволяє не лише зрозуміти вимоги ринку, але і забезпечити більш повне розуміння потреб користувачів. Для аналізу було обрано наступні додатки:

1. RealPlans;
2. Emeals;
3. Mealpy.

RealPlans [9] – це проєкт, що допомагає персоналізувати планування здорових прийомів їжі для родини, який враховує час та бюджет.

На рисунку 1.1 зображено головну сторінку рецептів.

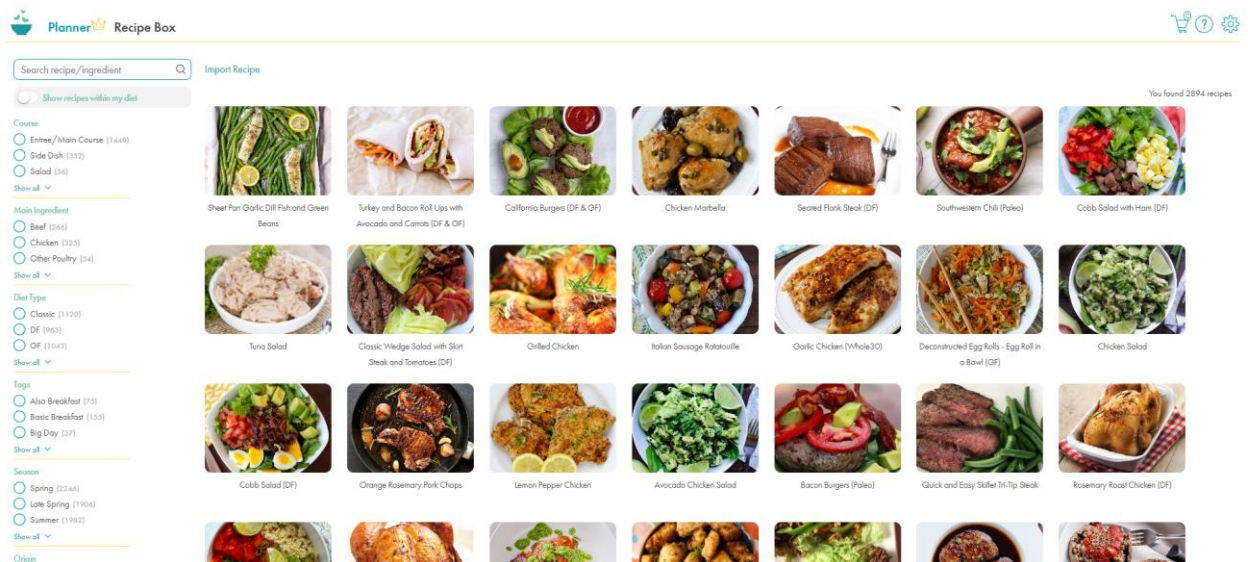


Рисунок 1.1 – RealPlans (відображення рецептів)

Можна виокремити наступні переваги даного вебсайту:

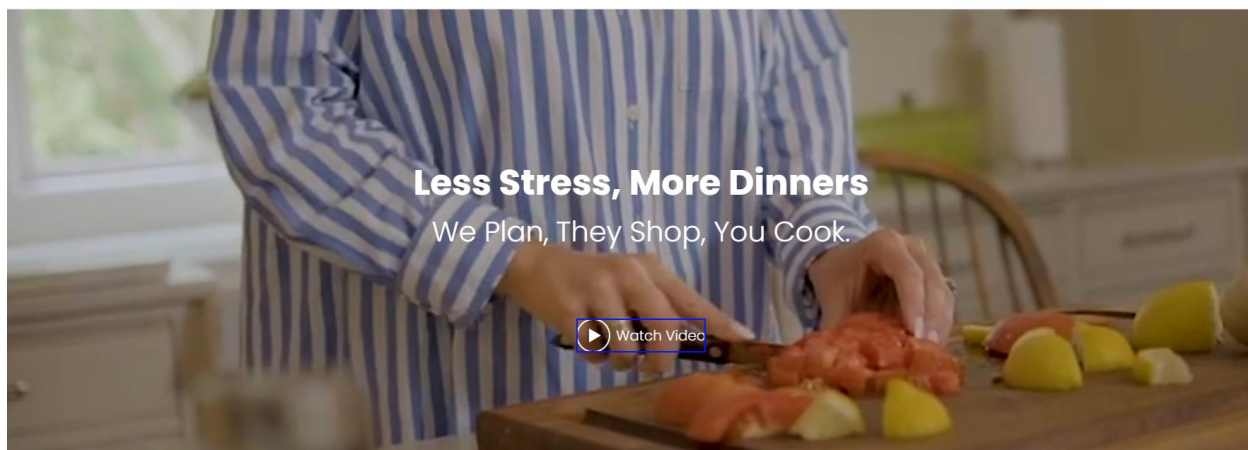
- гарний та інтуїтивний дизайн;
- рецепти під різні дієти;
- закупівля продуктів через різноманітні сервіси доставок;
- присутня веб та мобільна версія.

До недоліків можна віднести наступне:

- необхідно мати підписку для доступу до планувальника;
- сервіси доставок не працюють на території України;
- мало гнучкий розподіл трапез;
- відсутність української мови;
- відсутність порад щодо приготування або зберігання продукції.

Emeals [10] – це простий у користуванні додаток для планування їжі, який полегшує споживання здорової домашньої їжі завдяки щотижневим рецептам, списку покупок і розумним варіантам покупок.

На рисунку 1.2 зображено початкову сторінку додатку.



How It Works



Рисунок 1.2 – Emeals (початкова сторінка)

До переваг можна віднести:

- гарний та інтуїтивний дизайн;
- автоматичне формування щотижневих рецептів;
- рецепти під різні дієти;
- закупівля продуктів через різноманітні сервіси доставок.

Недоліками є наступне:

- необхідно мати підписку для доступу до планувальника;
- сервіси доставок не працюють на території України;
- відсутність української мови;
- відсутність порад щодо приготування або зберігання продукції.

Mealpy [11] – проєкт для планування власного харчування та приготування їжі на тиждень за допомогою спрощеної та легкої програми.

На рисунку 1.3 зображено сторінку планування рецептів.



Рисунок 1.3 – Mealpy (сторінка планування трапез)

Можна виокремити наступні переваги даного вебсайту:

- рецепти під різні дієти;
- присутня мобільна версія;
- простота та легкість використання;
- доступний функціонал без підписок.

До недоліків можна віднести наступне:

- мало гнучкий розподіл трапез;
- відсутність української мови;
- відсутність порад щодо приготування або зберігання продукції;
- трохи застарілий інтерфейс.

Структуруємо отримані дані за допомогою таблиці.

Таблиця 1.1

Характеристика	RealPlans	emeals	Mealpy	Наш додаток
Інтуїтивний та зрозумілий дизайн	+	+	+-	+

Закупівля продуктів через різноманітні сервіси доставок	+	+	-	-
Присутня вебверсія	+	+	-	-
Пробний період користування додатком	-	-	+	-
Українська локалізація	-	-	-	+
Автоматичний переклад та перевірка публічних рецептів	-	-	-	+
Поради щодо приготування та зберігання продукції	-	-	-	+
Гнучке налаштування трапез	+/-	+/-	-	+

Грунтуючись на отриманих результатах, отримуємо наступні критерії для додатку, що розроблюється:

- інтуїтивний та зрозумілий дизайн;
- присутність мобільної версії;
- відсутність тимчасових пробних періодів;
- українська та англійська локалізація;
- автоматичний переклад та перевірка публічних рецептів;
- поради щодо приготування та зберігання продукції;
- гнучке налаштування трапез.

1.3 Вибір архітектури мобільно-орієнтованої системи планування трапез

Для того, щоб розроблювана інформаційна система могла добре функціонувати довгий час, вона має забезпечувати поставленим вимогам, мати високу якість та бути доступною та легкою у використанні для звичайних

користувачів. На всі ці параметри впливає архітектура програмного забезпечення.

Архітектура – це базова організація системи, втілена в її компонентах, зовнішніми та внутрішніми відношеннями у системі, а також принципи, що визначають проектування та розвиток системи [12].

Правильно та своєчасно спроектована архітектура полегшить подальшу розробку, впровадження, користування, модифікацію інформаційної системи.

В сучасному світі існує кілька архітектур для побудови мобільного або вебдодатку. Розглянемо декілька з них, такі як:

- схема MVC (Model-View-Controller);
- схема SPA (Single-Page Application);
- JavaScript bridge ("міст JavaScript") [13].

Тепер докладніше розберемо кожен схему.

MVC (рис. 1.4) – це архітектурний шаблон, який дозволяє розділити компоненти додатку на три основні частини: модель (дані), вид (інтерфейс користувача) і контролер (бізнес-логіка). Модель представляє дані додатку, вид відповідає за відображення інформації користувачу, а контролер обробляє запити користувача та взаємодіє з моделлю і видом [14].

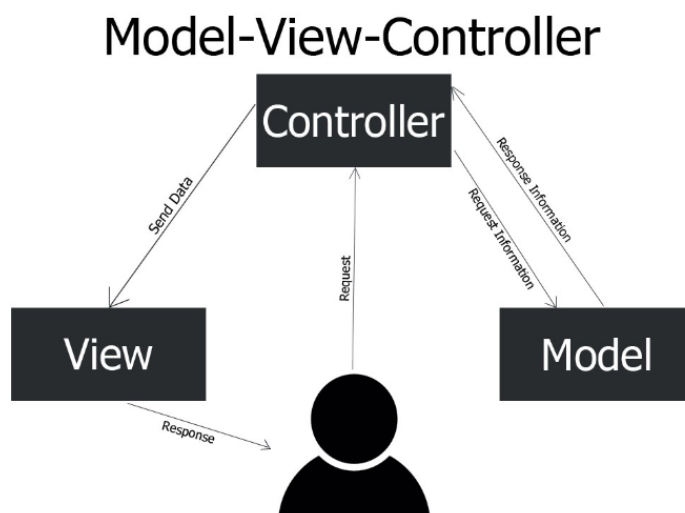


Рисунок 1.4 – Схема MVC

До його переваг належать:

- чіткий розподіл обов'язків додатку на три основні компоненти – модель, вид та контролер. Це забезпечує чітку структуризацію логіки додатку та полегшує розуміння і підтримку коду, а також спрощує тестування та забезпечує більшу модульність;
- розділення відповідальностей між різними компонентами дозволяє команді розробників працювати паралельно над різними частинами додатку, що сприяє прискоренню процесу розробки;
- підтримка одночасної розробки дає можливість різним членам команди працювати над окремими компонентами незалежно один від одного. Це сприяє швидкому розвитку проекту і дозволяє легко масштабувати команду розробників.

До недоліка можна віднести наступне:

- в деяких випадках контролер може стати централізованою точкою в додатку і надмірно навантажити з логічної точки зору. Це може вплинути на швидкість і продуктивність додатку.

SPA – це архітектурний підхід, в якому весь вебдодаток завантажується один раз, а взаємодія з сервером відбувається через асинхронні запити. Такий підхід дозволяє створювати більш різноманітні інтерфейси, зменшувати завантаження сторінок та збільшувати продуктивність додатку [15].

На рисунку 1.6 зображено схему SPA.



Рисунок 1.6 – Схема SPA

До його переваг належать:

- весь код програми завантажується один раз при початковому завантаженні, а надалі навігація і взаємодія з додатком без повного перезавантаження сторінки, що забезпечує більш плавну та швидку реакцію на дії користувача і поліпшує загальний користувацький досвід;
- сторінка не перезавантажується, тому немає необхідності завантажувати всі ресурси повторно, такі як HTML, CSS та JavaScript, під час кожної навігації, що в свою чергу скорочує час завантаження сторінки та покращує продуктивність програми;
- частину обробки відображення інтерфейсу користувача виконується на стороні клієнта. Це надає менше навантаження на сервер, оскільки він відповідає за надання даних, а не за відмальовування сторінок;
- забезпечує більш гнучку розробку, оскільки клієнт та сервер працюють незалежно один від одного. Це дозволяє команді розробників працювати паралельно і використовувати різні технології для клієнтської та серверної частини.

До недоліків належать:

- оскільки клієнтська сторона більш завантажена та весь код програми завантажується на клієнтському пристрої, для виконання таких операції потрібно більше ресурсів, наприклад: потужність процесора, кількість оперативної пам'яті і т.д.
- традиційні пошукові системи не завжди ефективно індексують динамічно генерований вміст сторінок на стороні клієнта для SEO-оптимізації.

"JavaScript bridge" - це механізм взаємодії між нативним кодом (написаним для конкретної платформи, такої як iOS чи Android) та кодом, написаним на мові програмування JavaScript, який використовується для розробки мобільних додатків, зокрема в технології React Native.

До його переваг належать:

- мова програмування JavaScript може використовуватися для розробки додатків для різних платформ, що зменшує витрати часу та ресурсів;
- застосування "JavaScript bridge" дозволяє використовувати однаковий код для реалізації бізнес-логіки на обох платформах, що полегшує підтримку та розширення;
- розробка на мові JavaScript може бути швидшою та зручнішою для багатьох розробників, оскільки це відома та популярна мова програмування;
- існує можливість використання бібліотек та фреймворків для створення інтерфейсу, таких як React або Vue, що дозволяє швидко розробляти інтерактивні та естетичні компоненти.

До недоліків належать:

- взаємодія між JavaScript та нативним кодом через "міст" може призводити до додаткових витрат на ресурси, зокрема, збільшення використання пам'яті та затримки у виконанні коду;
- деякі специфічні функції платформи можуть бути менш доступними або потребувати специфічного нативного коду, що може ускладнити інтеграцію деяких функцій;
- для великих та важких застосунків може виникнути питання з продуктивністю через переключення між JavaScript і нативним кодом.

Проаналізувавши вищеописані архітектури, було вирішено використовувати архітектуру "JavaScript bridge", щоб надати користувачам досвід, аналогічному до використання традиційних мобільних додатків.

1.4 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Розробка обраного програмного продукту передбачає використання певних інструментів. Для написання серверної частини додатку було вирішено скористатися редактор WebStorm.

Безпосередньо для розробки серверної частини було обрано мову програмування JavaScript. Для неї існує кілька фреймворків, але найпопулярнішими є – це NestJS [7] та Express. З цих фреймворків було обрано NestJS, бо він, на відміну від Express, вводить додаткові концепції та функції, надає модульну систему, використовує шаблон Dependency Injection [16] та підтримує TypeScript [17].

На стороні клієнта буде використовуватися бібліотека мови програмування JavaScript – Ехро [18]. На даний момент, це найпопулярніше рішення для створення мобільних додатків згідно архітектури JavaScript bridge. Ехро працює на основі React, який має одну з найбільших спільнот розробників, що забезпечує більш широку підтримку, більшу кількість розширень та доступів до багатофункціональних бібліотек.

В якості системи управління базами даних було обрано pgAdmin [19]. Вибір на користь цієї технології був зупинений через наступні переваги:

- є вільно поширюваним програмним забезпеченням з відкритим вихідним кодом, що дозволяє вам використовувати його без витрат на ліцензії та адаптувати до власних потреб;
- підтримується на різних операційних системах, включаючи Windows, macOS та Linux, що робить його універсальним для розробників;
- має зручний та інтуїтивний візуальний інтерфейс, що полегшує адміністрування та взаємодію з базою даних, особливо для користувачів, які не хочуть використовувати командний рядок;

- надає багато корисних функцій для адміністрування бази даних, таких як створення та редагування таблиць, виконання SQL-запитів, керування користувачами та правами доступу.

В якості емулятора для розробки мобільного застосунку буде використовуватися Android Studio [20]. Це інтегроване середовище розробки (IDE), спеціально призначене для створення додатків на платформі Android. Розроблене компанією Google, Android Studio надає розробникам широкий спектр інструментів та можливостей для зручної та ефективної роботи над Android-проектами. Основні характеристики включають:

- графічний інтерфейс для розміщення та редагування елементів інтерфейсу користувача;
- вбудований емулятор, який дозволяє тестувати додатки на різних пристроях та версіях Android;
- інтеграція з Android Software Development Kit, що дозволяє користувачам легко взаємодіяти з різними версіями Android API;
- інструменти для виявлення та виправлення помилок в коді, включаючи відладку та профілювання додатка.

Для забезпечення ефективної роботи системи, максимальної продуктивності та оптимального використання апаратних ресурсів, були встановлені наступні вимоги:

1. Вимоги до серверного обладнання:

- оперативна пам'ять – 8 GB;
- жорсткий диск - 128 GB;
- операційна система – Linux;
- процесор з частотою від 3 GHz;
- node.js версії 20 і вище.

2. Вимоги до робочої станції користувача:

- підтримка операційної системи Android версії 8.0+;
- підтримка операційної системи iOS версії 14+.

Висновки до першого розділу

У цьому розділі було виконано постановку задачі на кваліфікаційну роботу. Визначено функціональні вимоги та основні етапи розробки.

Проведено аналіз аналогів до розроблювального продукту. І завдяки цьому були отримані наступні критерії для розробки: зрозумілий та інтуїтивний дизайн; наявність мобільної версії; відсутність щомісячної підписки; українська та англійська локалізація; поради щодо приготування та зберігання продукції; гнучке налаштування трапез.

Було проаналізовано різні архітектури та вирішено використовувати схему JavaScript bridge, щоб надати користувачам досвід, близького до користування звичайних мобільних додатків.

Крім того, було обґрунтовано вибір використання наступного технологічного стека для реалізації мобільного застосунку: NestJS для побудови серверної частини, Ехро – для клієнтської частини; в якості системи керування базами даних обрано pgAdmin; в якості емулятора для розробки мобільного застосунку буде використовуватися Android Studio.

Були встановлені вимоги до серверного обладнання та робочих станцій користувачів.

РОЗДІЛ 2. ПРОЕКТУВАННЯ МОБІЛЬНО-ОРІЄНТОВАНОЇ СИСТЕМИ ПЛАНУВАННЯ ТРАПЕЗ

2.1 Визначення варіантів використання та об'єктно-орієнтованої структури системи

Мобільна платформа для управління своїми харчовими раціонами має наступні основні цілі: централізоване сховище рецептів за різними категоріями, оптимізацію розподілу прийомів їжі, контролювання постачання продукції у вигляді підсумованих списків на кожен день, гнучка варіативність налаштування днів трапез, генерування додаткових порад для забезпечення продуктивного використання додатку.

Ця платформа забезпечує місце для поширення своїх рецептів іншим користувачам, навіть якщо вони англomовні. Спрощує логістику та підвищує ефективність роботи для поварів. Крім того, платформа має бути зручною та доступною, з інтуїтивним інтерфейсом, який зможуть використовувати користувачі різної вікової категорії.

Внутрішній користувач:

Внутрішнім користувачам надаються ряд функцій, що включають:

1. На початку роботи, створення облікового запису за допомогою логіну та паролю або за допомогою запису Google, додавання персональних даних для майбутніх публікацій. Також скидання паролю потребує виконання додаткових дій, а саме підтвердження за допомогою тимчасового коду який відсилається на пошту користувача;
2. Можливість пошуку, фільтрування та перегляд загальної інформації про рецепт по різним категоріям. Переглядати більш детальну інформацію, включаючи перелік інгредієнтів, інструкцію по приготуванню, дані про калорійність та харчову цінність;

3. Можливість створювати рецепти як для персонального використання та і для публікацій, що в свою чергу надасть автоматичну локалізацію для англомовних користувачів;
4. Постійне отримання генерованих порад по приготуванню, наприклад: зберігання та транспортування готової продукції або правила гігієни в залежності від кожного налаштованого дня, так як для цього буде використано штучний інтелект користувач зможе отримувати кожного дня новий досвід користуючись цим функціоналом;
5. Налаштування дня зі створенням персональних прийомів їжі, з урахуванням часу приготування, часу трапези та кількості порцій. Від змін певних критерій можуть змінюватися обрахунки для доданих рецептів;
6. Перегляд трапез на теперішній тиждень або поточний рік. На кожному дні відображатимуться всі заплановані трапези зі списками доданих рецептів, за потреби користувач може редагувати їх, маніпулюючи рецептами та параметрами;
7. Формування списку покупок по датам, який буде для кожної трапези обраховувати коефіцієнти для формування кількості продукції. Додатково користувач зможе додавати свої нотації та позначати купленими інші;
8. Можливість налаштовувати додаток під свої вподобання, змінюючи тему додатку або локалізацію. Також модифікувати свої персональні данні. В залежності від обраної мови користувачу будуть відображатися нові данні зі серверу.

Для наглядної взаємодії між елементами додатку було створено діаграму варіантів використання (рис 2.1)

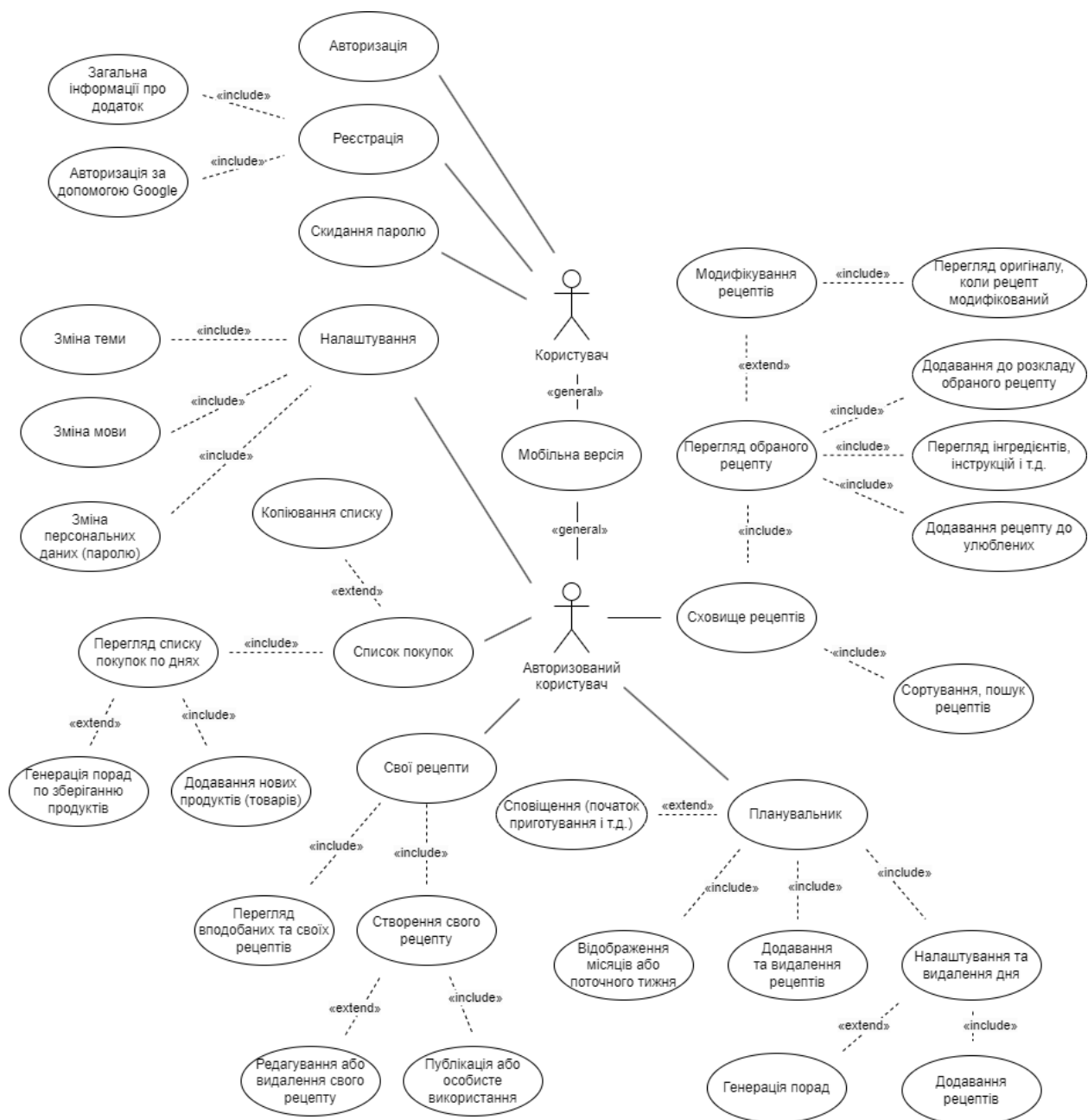


Рисунок 2.1 – Діаграма варіантів використання

Функціональні вимоги:

Мобільна система управління трапезами надає користувачам такі можливості:

1. Перегляд, додавання до планів, вподобання, пошук та сортування рецептів в системі;

2. Створення, редагування інформації про рецепти, включаючи інгредієнти з детальним описом та інформацією про поживну цінність;
3. Управління планами, часом трапез, списками продуктів;
4. Публікація рецепту з автоматичною генерацією локалізацій;
5. Перегляду порад для різних умов приготування та запланованих рецептів;
6. Редагування профілю зі зміною персональних даних, мовою додатку або темою.

Нефункціональні вимоги:

Навчання роботи з системою:

- Час навчання для звичайних користувачів – 1-2 години, для досвідчених – 30 хвилин;
- Швидкість відповіді максимум 1 секунда на звичайні запити, 5 секунд – на складніші запити;
- Інтуїтивний та легкий у використанні інтерфейс, який не потребує додаткової підготовки;
- Стабільна та надійна робота системи без збоїв;
- Час обслуговування не перевищує 1-2% загального часу роботи;
- Безперебійна робота на протязі 20 робочих днів;
- Максимум 1 помилка на 1000 запитів користувача;
- Продуктивність системи є важливою нефункціональною вимогою, яка включає швидкість відповіді, ефективну обробку даних, масштабованість, економне використання ресурсів, оптимізацію запитів, паралельну обробку, надійність та ефективне керування ресурсами. Система повинна забезпечувати швидкість відповіді не більше 1-3 секунд, оптимальне використання ресурсів та надійну роботу. Крім того, важливо, щоб система була масштабованою, щоб впоратися з великим обсягом даних та багатьма користувачами.

Аналіз функціональних вимог дозволив виділити наступні основні сутності, які забезпечать реалізацію програмного комплексу системи управління проектами:

- User – сутність для роботи з персональними даними користувачів;
- Recipe – сутність, яка містить методи роботи з рецептами та їх перекладами;
- Plan – дозволяє працювати з планами користувачів по кожному дню;
- MealPlan – сутність для взаємодії з трапезами планів.

Так як система розроблюється з використанням NestJS, то стандартна діаграма класів не зовсім підходить для побудови архітектури додатку. Так як у NestJS функціональність додатка розділяється на модулі, то вони є основними будівельними блоками додатка і дозволяють організувати код в окремі, автономні частини з чіткою структурою та залежностями. Тому діаграма класів буде представлена як діаграма модулів на рисунку 2.2.

Дана діаграма демонструє загальну ієрархію системи, в свою чергу кожен модуль містить такі класи як:

- module – описує внутрішні залежності;
- controller – відповідають за обробку вхідних запитів та відповідей клієнту;
- service – містить бізнес логіку системи, репозиторії, фабрики або допоміжні функції;
- mapper – трансформує дані для клієнта.

В розділі 2.2 буде наведена модель бази даних системи, що відповідає цим сутностям та організована на рівні моделі.

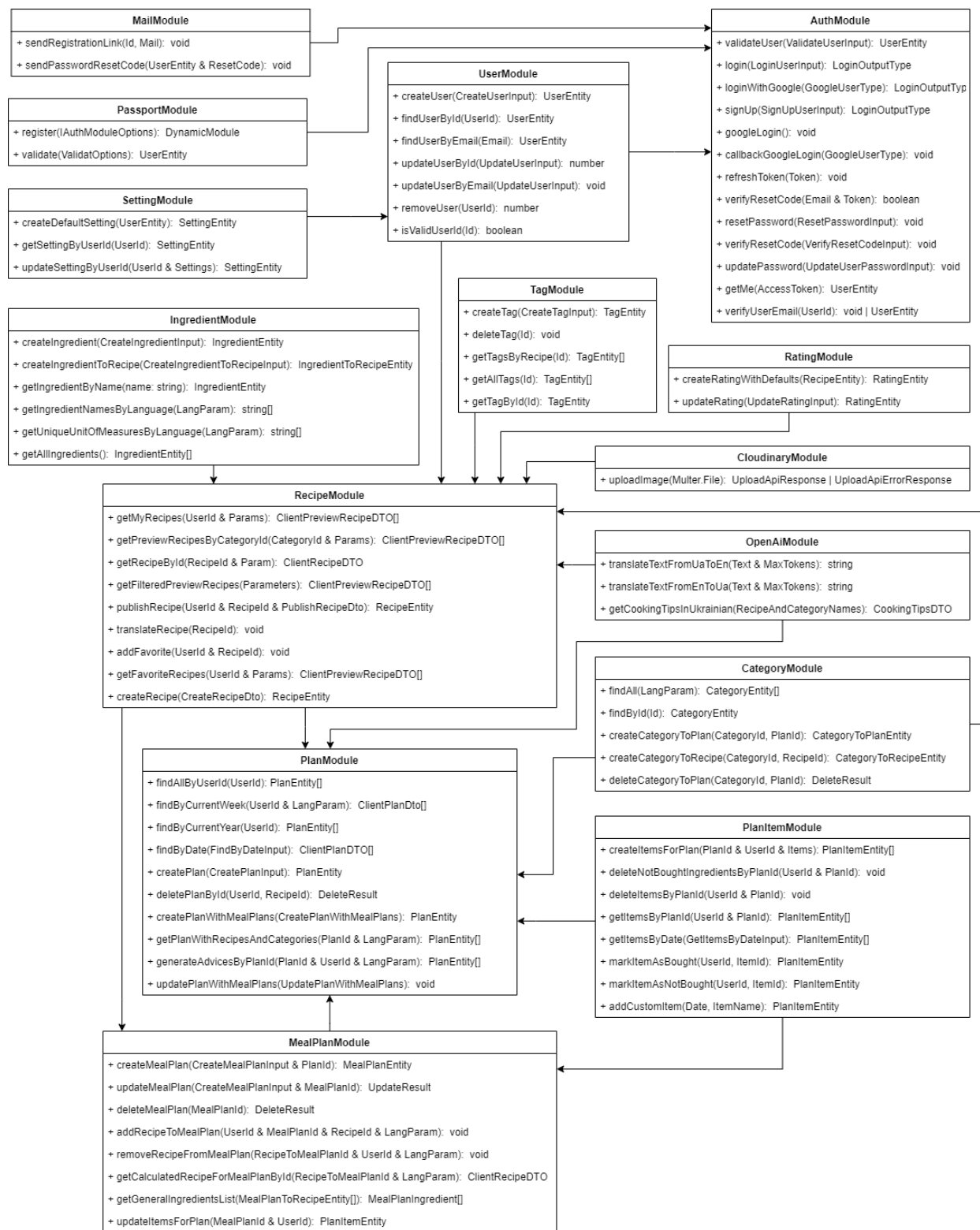


Рисунок 2.2 – Діаграма модулів

На даному рисунку можна виділити наступні модулі:

- Recipe – модуль роботи з рецептами які належать користувачам та використовуються для створень харчових планів.

Методи:

1. createRecipe(CreateRecipeDto) – метод дозволяє створювати свої персональні рецепти для подальшого використання в планах або для публікації;
 2. getMyRecipes(UserId & Params) – цей метод дозволяє отримати список рецептів користувача враховуючи мовні параметри;
 3. getPreviewRecipesByCategoryId(CategoryId & Params) – використовується для отримання основної інформації рецептів по певному критерію та обраній мові;
 4. getRecipeById(RecipeId & Param) – даний метод повертає певний рецепт по унікальному полю ID враховуючи мову;
 5. getFilteredPreviewRecipes(Parameters) – повертає список рецептів з основною інформацією враховуючи параметрами для пагінації, фільтрування, обраної мови;
 6. publishRecipe(UserId & RecipeId & PublishRecipeDto) – цей метод публікує приватний рецепт, перекладаючи його на інші мови та з подальшим відображенням іншим користувачам;
 7. translateRecipe(RecipeId) – даний метод використовується для утворень генерованих перекладів рецептів, інгредієнтів, одиниць виміру за допомогою штучного інтелекту;
 8. addFavorite(UserId & RecipeId) – додає обраний рецепт користувачем до списку вподобаних;
 9. getFavoriteRecipes(UserId & Params) – цей метод повертає список вподобаних рецептів певному користувачу та з відповідною обраною мовою.
- Plan – модуль роботи з харчовими планами.

Методи:

1. findAllByUserId(UserId) – цей метод повертає всі налаштовані плани певному користувачу;

2. `findByCurrentWeek(UserId & LangParam)` – використовується для отримання списку харчових планів на поточний тиждень враховуючи мовні параметри;
 3. `findByCurrentYear(UserId & LangParam)` – даний метод дозволяє отримати список планів на поточний рік;
 4. `findByDate(FindByDateInput)` – знаходить запис харчового плану по певному дню, та повертає його відповідно до вказану мову;
 5. `createPlan(CreatePlanInput)` – метод для створення нового плану для обраного дня;
 6. `deletePlanById(UserId, RecipeId)` – цей метод дозволяє видалити план разом зі всіма залежностями, наприклад з доданими налаштуваннями та рецептами;
 7. `createPlanWithMealPlans(CreatePlanWithMealPlans)` – створює план разом з налаштованими харчовими раціонами на день, вказуючи кількість порцій, час приготування;
 8. `getPlanWithRecipesAndCategories(PlanId & LangParam)` – отримання списку планів з повною інформацією враховуючи обрану мову та унікальний ідентифікатор для плану;
 9. `generateAdvicesByPlanId(PlanId & UserId & LangParam)` – генерує поради на основі обраного плану, використовуючи штучний інтелект та трансформували результату для клієнта;
 10. `updatePlanWithMealPlans(UpdatePlanWithMealPlans)` – цей метод потрібен для оновлення плану разом з внутрішніми раціонами.
- OpenAi – модуль роботи з генеруванням порад, перекладів.

Методи:

1. `translateTextFromUaToEn(Text & MaxTokens)` – цей метод перекладає текст з української на англійську мову за допомогою штучного інтелекту;
2. `translateTextFromEnToUa(Text & MaxTokens)` – перекладає текст з англійської на українську мову;

3. `getCookingTipsInUkrainian(RecipeAndCategoryNames)` – метод отримує назви рецептів та категорій по певному дню, що в свою чергу допомагає створювати правильний промпт для отримання порад по приготуванню, зберіганню готової продукції та гігієни.
- `AuthModule` – модуль роботи з генеруванням порад та перекладів.

Методи:

1. `validateUser(ValidateUserInput)` – цей метод є основою для сервісів стратегій Google, JWT, local авторизацій разом зі перевіркою паролю та електронної адреси;
2. `login(LoginUserInput)` – дозволяє авторизуватися користувачу в системі повертаючи токен;
3. `loginWithGoogle(GoogleUserType)` – метод авторизує або реєструє користувача по Google токену перевіряючи його та повертає внутрішній токен для подальшого доступу до додатку;
4. `signUp(SignUpUserInput)` – валідує вхідні данні та створює обліковий запис для користувача;
5. `googleLogin()` – використовується для веб частини, перенаправляє користувача на гугл сторінку авторизації;
6. `callbackGoogleLogin(GoogleUserType)` – використовується для останнього кроку авторизації користувача за допомогою Google;
7. `refreshToken(Token)` – валідує та оновлює токен для підтримки сесії користувача в додатку;
8. `resetPassword(ResetPasswordInput)` – створює та відправляє тимчасовий код для скидання паролю на електронну пошту користувача;
9. `verifyResetCode(Email & Token)` – цей метод перевіряє валідність тимчасового коду для подальшої зміни паролю;
10. `updatePassword(UpdateUserPasswordInput)` – надає змогу оновити пароль для облікового запису, перевіряючи вхідний код з електронного листа;

11.getMe(AccessToken) – повертає інформацію та налаштування користувача по токену;

12.verifyUserEmail(UserId) – перевіряє чи існує вхідний email в системі.

Таким чином, дана система реалізує функціонал підбору різноманітних рецептів, гнучке налаштування трапез та планів, отримання генерованих порад по кожному дню, локалізацію серверної та клієнтської частини, відстежування забезпечення та зручний етап реєстрації та авторизації різними методами. Безпосередньо взаємодіючи з різноманітними контролерами на кшталт: RecipeController, PlanController, AuthController, MealPlanController та т.д.

2.2 Розробка бази даних системи

Для збереження файлів та інформації у додатку використовується PostgreSQL, це система управління базами даних, яка базується на реляційній моделі.

База даних складається з 21 таблиці, які містять у собі всі дані для стабільної роботи програми. Інформація про таблиці та опис кожної з них наведений у таблицях 2.1 – 2.18.

Таблиця 2.1

Інформація про таблиці бази даних.

Назва таблиці	Призначення
User	Таблиця користувачів
Recipe	Таблиця рецептів
Setting	Таблиця налаштувань користувача
FavoriteRecipe	Проміжна таблиця для зберігання вподобаних рецептів між користувачами та рецептами
Rating	Таблиця рейтингу для рецептів
Tag	Таблиця тегів
TagTranslation	Додаткова таблиця для зберігання полів тегів для перекладу
Ingredient	Таблиця інгредієнтів
IngredientTranslation	Додаткова таблиця для зберігання полів інгредієнтів для перекладу

RecipeToIngredient	Проміжна таблиця зв'язку між рецептами та інгредієнтами
RecipeToIngredientTranslation	Додаткова таблиця для зберігання полів одиниць виміру та коментарів інгредієнтів для перекладу
RecipeTranslation	Додаткова таблиця для зберігання полів рецептів для перекладу
Category	Таблиця категорій рецептів
RecipeToCategory	Проміжна таблиця зв'язку між рецептами та категоріями
CategoryTranslation	Додаткова таблиця для зберігання полів категорій для перекладу
Plan	Таблиця планів
PlanToCategory	Проміжна таблиця зв'язку між планами та категоріями
AdditionalIngredients	Таблиця додаткових інгредієнтів
MealPlan	Таблиця трапез для кожного плану
PlanToRecipe	Проміжна таблиця зв'язку між планами та рецептами

Структура бази даних наведена на рисунку 2.3.

Таблиця 2.2

Опис полів таблиці “Recipe”

Назва	Тип даних	PK	FK	Опис поля
RecipeId	integer	+	-	Унікальний ідентифікатор
ImageUrl	varchar	-	-	Посилання на фото рецепту
CalorieContent	integer	-	-	Кількість калорій
Weight	float	-	-	Вага страви
NumberOfServings	float	-	-	Кількість порцій
UserId	integer	-	User.Id	Ідентифікатор користувача
CookingTime	integer	-	-	Час приготування в хвилинали
Protein	float	-	-	Кількість білків
Fat	float	-	-	Кількість жирів
Carbohydrates	float	-	-	Кількість вуглеводів

Таблиця “RecipeTranslation” призначена для зберігання полів додаткової інформації про рецепт, які є залежними від мови, наприклад назва рецепту, опис, інструкція приготування. Також записи зберігають код мови якою було описано поля та посилання на рецепт до якого додають нову локалізацію. Структура таблиці наведена нижче (табл. 2.3).

Таблиця 2.3

Опис полів таблиці “RecipeTranslation”

Назва	Тип даних	PK	FK	Опис поля
RecipeTranslationId	integer	+	-	Унікальний ідентифікатор
Title	varchar	-	-	Назва рецепту
Description	varchar	-	-	Опис рецепту
CookingInstructions	varchar	-	-	Інструкція приготування
LanguageCode	enum	-	-	Скорочений код мови
RecipeId	integer	-	Recipe.Id	Ідентифікатор рецепту

Таблиця “User” призначена для зберігання облікових даних користувачів, може в собі містити персональні данні користувача, токен для Google авторизації, або тимчасовий код для скидання паролю. Нижче наведена структура таблиці (табл. 2.4).

Таблиця 2.4

Опис полів таблиці “User”

Назва	Тип даних	PK	FK	Опис поля
UserId	integer	+	-	Унікальний ідентифікатор
Email	varchar	-	-	Електронна адреса
LastName	varchar	-	-	Ім'я користувача
FirstName	varchar	-	-	Прізвище користувача
Password	varchar	-	-	Хешований пароль
GoogleId	varchar	-	-	Ідентифікатор Google ід
Gender	enum	-	-	Стать користувача
Token	varchar	-	-	Тимчасовий токен скидання паролю

Таблиця “Setting” призначена для зберігання налаштувань користувачів для забезпечення зберігання різних параметрів між різними пристроями. Записи можуть містити налаштування обраної мови, теми, нотифікацій та містити зв'язок з таблицею користувачів. Нижче наведена структура таблиці (табл. 2.5).

Таблиця 2.5

Опис полів таблиці “Setting”

Назва	Тип даних	PK	FK	Опис поля
SettingsId	integer	+	-	Унікальний ідентифікатор
Theme	enum	-	-	Тема застосунку
Language	enum	-	-	Мова застосунку
AdviceNotifications	boolean	-	-	Прапорець для погодження нотифікацій порад
ReminderNotifications	boolean	-	-	Прапорець для погодження нотифікацій нагадувань
StorageNotifications	boolean	-	-	Прапорець для погодження нотифікацій забезпечення
UserId	integer	-	User.Id	Стать користувача

Таблиця “FavoriteRecipe” призначена для зберігання вподобаних рецептів користувачами. Являється додатковою між зв'язком багато до багатьох між таблицями “User” та “Recipe”, відповідно має необхідні для цього поля. Структура таблиці наведена нижче (табл. 2.6).

Таблиця 2.6

Опис полів таблиці “FavoriteRecipe”

Назва	Тип даних	PK	FK	Опис поля
FavoriteRecipeId	integer	+	-	Унікальний ідентифікатор
RecipeId	integer	-	Recipe.Id	Унікальний ідентифікатор рецепту
UserId	integer	-	User.Id	Унікальний ідентифікатор користувача

Таблиця “Rating” призначена для зберігання рейтингу рецепту, зберігаючи саме середню оцінку та кількість відгуків для підрахування подальших значень. Структура таблиці наведена нижче (табл. 2.7).

Таблиця 2.7

Опис полів таблиці “Rating”

Назва	Тип даних	PK	FK	Опис поля
RatingId	integer	+	-	Унікальний ідентифікатор
RecipeId	integer	-	Recipe.Id	Унікальний ідентифікатор рецепту
totalRating	float	-	-	Середній рейтинг рецепту
totalUsersRated	integer	-	-	Кількість відгуків

Таблиця “Ingredient” кожен рецепт повинен складатися з різних інгредієнтів, тому ця сутність призначена для зберігання різноманітних складових. Так як більшість полів мають залежність від локалізації, в цій таблиці ми зберігаємо формальні дані на кшталт дати створення, щоб не відходити від патерну з перекладами. Нижче наведена структура таблиці (табл. 2.8).

Таблиця 2.8

Опис полів таблиці “Ingredient”

Назва	Тип даних	PK	FK	Опис поля
IngredientId	integer	+	-	Унікальний ідентифікатор
CreatedAt	timestampz	-	+	Час створення рецепту

Таблиця “IngredientTranslation” призначена для зберігання всієї інформації про інгредієнти, що підлягають перекладу. Містить в собі зв’язок з

певним інгредієнтом та кодом мови. Структура таблиці наведена нижче (табл. 2.9).

Таблиця 2.9

Опис полів таблиці “IngredientTranslation”

Назва	Тип даних	PK	FK	Опис поля
IngredientTranslationId	integer	+	-	Унікальний ідентифікатор
Name	varchar	-	-	Назва інгредієнту
Description	varchar	-	-	Опис інгредієнту
IngredientId	integer	-	Ingredient.Id	Унікальний ідентифікатор інгредієнту
LanguageCode	enum			Скорочений код мови

Таблиця “RecipeToIngredient” призначена для зв’язку між рецептами та інгредієнтами, оскільки маємо відношення багато до багатьох. Також в цій сутності зберігається кількість кожного доданого інгредієнту. Структура таблиці наведена нижче (табл. 2.10).

Таблиця 2.10

Опис полів таблиці “RecipeToIngredient”

Назва	Тип даних	PK	FK	Опис поля
RecipeToIngredientId	integer	+	-	Унікальний ідентифікатор
IngredientId	integer	-	Ingredient.Id	Унікальний ідентифікатор інгредієнту
RecipeId	integer	-	Recipe.Id	Унікальний ідентифікатор рецепту
Quantity	integer	-	-	Кількість обраного інгредієнту

Таблиця “RecipeToIngredientTranslation” призначена для додаткових даних між інгредієнтами та рецептами, які підлягають до перекладу. Містить в собі обрану мову та унікальний ідентифікатор інгредієнту. Нижче наведена структура таблиці (табл. 2.11).

Таблиця 2.11

Опис полів таблиці “RecipeToIngredientTranslation”

Назва	Тип даних	PK	FK	Опис поля
-------	-----------	----	----	-----------

Продовження таблиці 2.11

RecipeIngredientTranslationId	integer	+	-	Унікальний ідентифікатор
UnitOfMeasure	varchar	-	-	Одиниці виміру
Comment	varchar	-	-	Коментар
RecipeToIngredientId	integer	-	RecipeToIngredient.Id	Унікальний ідентифікатор
LanguageCode	enum	-	-	Код мови

Таблиця “Category” призначена для розділення рецептів по категоріям містить в собі час створення, оскільки основні поля підлягають до локалізації. Нижче наведена структура таблиці (табл. 2.12).

Таблиця 2.12

Опис полів таблиці “Category”

Назва	Тип даних	PK	FK	Опис поля
CategoryId	integer	+	-	Унікальний ідентифікатор
CreatedAt	timestampz	-	+	Час створення категорії

Таблиця “CategoryTranslation” призначена для зберігання полів категорій під різні мови та має зв’язок з таблицею “Category”. Структура таблиці наведена нижче (табл. 2.13).

Таблиця 2.13

Опис полів таблиці “CategoryTranslation”

Назва	Тип даних	PK	FK	Опис поля
CategoryTranslationId	integer	+	-	Унікальний ідентифікатор
CategoryId	integer	-	Category.Id	Унікальний ідентифікатор категорії
Description	varchar	-	-	Опис категорії
Name	varchar	-	-	Ім’я категорії
LanguageCode	enum	-	-	Код мови

Додаткова таблиця “RecipeToCategory” призначена для зв’язку між сутностями “Recipe” та “Category”, щоб кожен запис рецепту міг мати декілька

категорій, які найкраще опишуть його. Нижче наведена структура таблиці (табл. 2.14).

Таблиця 2.14

Опис полів таблиці “RecipeToCategory”

Назва	Тип даних	PK	FK	Опис поля
RecipeToCategoryId	integer	+	-	Унікальний ідентифікатор
CategoryId	integer	-	Category.Id	Унікальний ідентифікатор категорії
RecipeId	integer	-	Recipe.Id	Унікальний ідентифікатор рецепту

Таблиця “Plan” призначена для записів планування трапез, має зв’язок з користувачем та зберігає дату планування. Нижче наведена структура таблиці (табл. 2.15).

Таблиця 2.15

Опис полів таблиці “Plan”

Назва	Тип даних	PK	FK	Опис поля
PlanId	integer	+	-	Унікальний ідентифікатор
Date	timestampz	-	-	Дата запланованого плану

Таблиця “MealPlan” призначена для зберігання раціонів на день, вказуючи кількість порцій, час трапези та приготування. Містить зв’язок з таблицею “Plan”. Структура таблиці наведена нижче (табл. 2.16).

Таблиця 2.16

Опис полів таблиці “MealPlan”

Назва	Тип даних	PK	FK	Опис поля
MealPlanId	integer	+	-	Унікальний ідентифікатор
PreparationStartTime	timestampz	-	-	Час початку приготування
PreparationEndTime	timestampz	-	-	Час закінчення приготування
MealStartTime	timestampz	-	-	Час початку трапези
MealEndTime	timestampz	-	-	Час закінчення трапези
PlanId	integer	-	Plan.Id	Унікальний ідентифікатор плану
Name	varchar	-	-	Назва трапези
TotalNumberOfServings	integer	-	-	Кількість порцій

Допоміжна таблиця “PlanToRecipe” призначена для зв’язку між запланованим раціоном та рецептом, оскільки маємо зв’язок багато до багатьох. Також ця таблиця зберігає коефіцієнт для прорахування кількості продукції під кожен рецепт. Нижче наведена структура таблиці (табл. 2.17).

Таблиця 2.17

Опис полів таблиці “PlanToRecipe”

Назва	Тип даних	PK	FK	Опис поля
PlanToRecipeId	integer	+	-	Унікальний ідентифікатор
RecipeId	integer	-	Recipe.Id	Унікальний ідентифікатор рецепту
MealPlanId	integer	-	MealPlan.Id	Унікальний ідентифікатор запланованої трапези
ServingRatio	float	-	-	Коефіцієнт розрахунку інгредієнтів

Таблиця “PlanItem” призначена для зберігання обрахованих по кількості інгредієнтів, даних про купівлю, або доданих додатково користувачем. Містить зв’язок з таблицею “Plan”. Структура таблиці наведена нижче (табл. 2.18).

Таблиця 2.18

Опис полів таблиці “PlanItem”

Назва	Тип даних	PK	FK	Опис поля
PlanItemId	integer	+	-	Унікальний ідентифікатор
Name	varchar	-	-	Назва продукту
Bought	boolean	-	-	Прапорець для позначення, що продукт куплено
IsCustom	boolean	-	-	Прапорець для позначення, що продукт додатковий
PlanId	integer		Plan.Id	Унікальний ідентифікатор плану

Таким чином реалізована база даних відповідно до вимог реляційної моделі, забезпечує зручний та ефективний спосіб збереження всіх необхідних даних для зручної роботи як користувачам так і розробникам, щоб підтримувати додаток. За допомогою таких таблиць як: User, Recipe,

Ingredient, Plan, MealPlan та PlanItem, можна організувати та управляти полегшеною логістикою у сфері забезпечення харчових процесів, що надає в свою чергу зручний інструментарій для роботи з різною інформацією.

2.3 Проектування та реалізація алгоритмів роботи системи

Клієнтський додаток системи має бути реалізований як мобільний застосунок, який встановлюється з відповідного файлу інсталяції або з мобільних магазинів додатків. При роботі користувачем з програмою та в залежності від його дій, мобільна система надсилає різні запити до серверу.

Сервер виступає за обробку та збереження даних у різних його видах. Після отримання відповіді від серверу, клієнт буде бачити нову інформацію, яка не є частиною встановленого застосунку. Додаток є простим у використанні, має всі основні функції платформи, що дозволяє користувачам легко взаємодіяти з платформою. Алгоритм роботи з програмою зображено на рисунку 2.4 та рисунку 2.5, у вигляді діаграми діяльності.

Перша частина демонструє діяльність нового або не зареєстрованого користувача. Відповідно друга частина розкриває всю діяльність після успішної автентифікації.

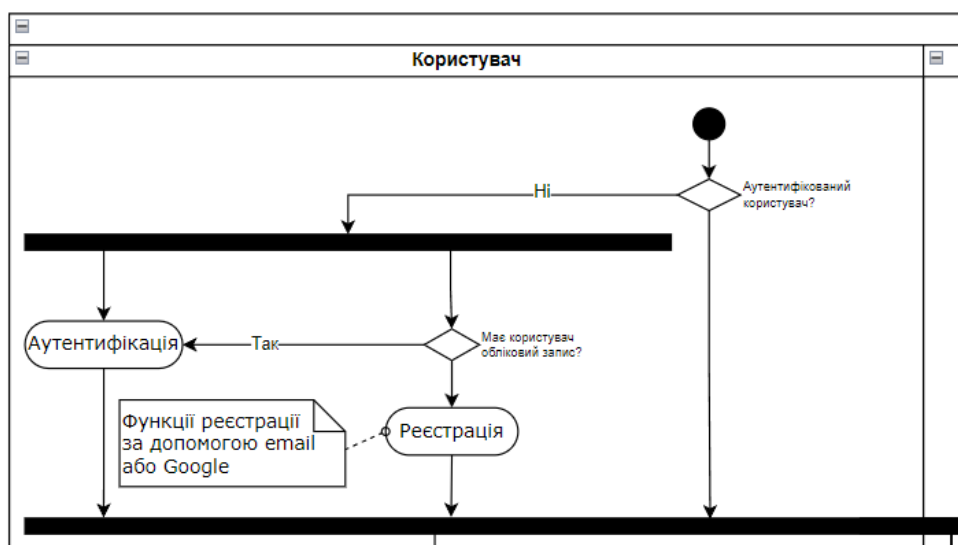


Рисунок 2.4 – Діаграма діяльності для не автентифікований користувача

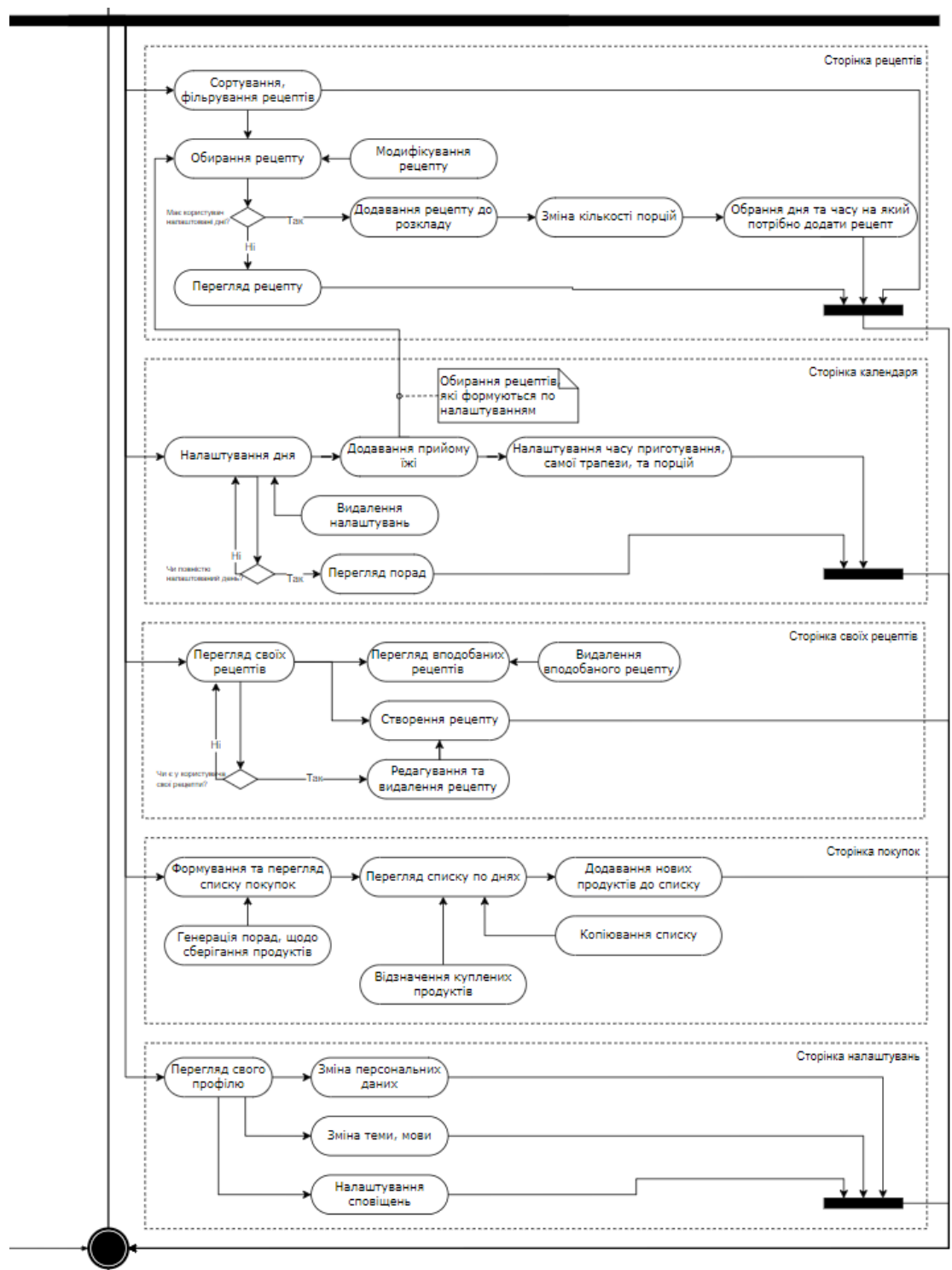


Рисунок 2.5 – Діаграма діяльності для автентифікований користувача

Опис діаграми діяльності:

1. Автентифікація (так – перехід до використання системи, ні – перехід на початок з повідомленням про необхідність наявності акаунту);
2. Перегляд рецептів
 - сортування, фільтрування, пошук рецептів;

- обирання рецепту;
- модифікування рецепту;
- додавання рецепту до обраного дня.

3. Перегляд календарю

- налаштування нового дня;
- додавання прийому їжі;
- перегляд порад.

4. Перегляд свої рецептів та вподобаних

- редагування та видалення рецепту;
- створення рецепту;
- видалення вподобаних рецептів.

5. Перегляд сторінки покупок

- формування списку продуктів по обраному дню;
- генерація порад, щодо зберігання продуктів.

6. Перегляд свого профілю

- зміна персональних даних;
- зміна мови та теми додатку.

2.4 Реалізація мобільно-орієнтованої системи планування трапез

Розглянемо основні етапи роботи платформи. Для цього користувачу потрібно здійснити реєстрацію нового аккаунту або за наявності його, здійснити авторизацію після чого він має можливість переглядати зміст та отримувати нові дані.

Для реалізації цього було використано PassportJS [21] та декілька стратегій для різних способів входу а саме: GoogleStrategy, JwtStrategy, LocalStrategy. Данні сервіси допомагають валідувати вхідні дані користувача, наприклад JWT або Google токени та порівнювати хешовані паролі. Сервіс по налаштуванню авторизації за допомогою Google [22] представлена на рисунку 2.6.

```

@Injectables()
export class GoogleStrategy extends PassportStrategy(Strategy) {
  constructor() {
    super({
      clientID: process.env.GOOGLE_CLIENT_ID,
      clientSecret: process.env.GOOGLE_CLIENT_SECRET,
      callbackURL: process.env.GOOGLE_CALLBACK_URL,
      scope: ['email', 'profile'],
    });
  }
  async validate(
    accessToken: string,
    refreshToken: string,
    profile: any,
    done: VerifyCallback,
  ): Promise<any> {
    const { name, emails, id } = profile;

    const user: GoogleUserType = {
      sub: id,
      email: emails[0].value,
      given_name: name.givenName,
      family_name: name.familyName,
    };

    done( err: null, user);
  }
}

```

Рисунок 2.6 – Сервіс стратегії Google

Також в даному сервісі налаштовуються доступи, які будуть використовуватися та запитуватися у користувача при створенні нового облікового запису. Подані стратегії використовуються як декоратори до відповідних запитів для того, щоб при успішному їх виконанні користувач отримав новий токен для авторизації. Використання декораторів та приклад авторизації зображено на рисунку 2.7 та рисунку 2.8.

```

@UseGuards(GoogleAuthGuard)
@Get( path: 'google')
async googleLogin(): Promise<void> {}

@Get( path: 'google-redirect')
@UseGuards(GoogleAuthGuard)
async callback(@Req() req: Request & { user: GoogleUserType })
  return this.authService.googleLogin(req.user);
}

```

Рисунок 2.7 – Приклад використання декораторів для авторизації

За допомогою засобів Google ми можемо як авторизуватися на платформі, так і створювати нові облікові записи без додаткових кроків з введення паролів та електронних адрес, що забезпечує кращий досвід роботи з системою у перші хвилини. Так як користувач надав згоду на використання даних, ми створюємо новий запис до таблиці User зі значеннями, які приходять після перевірки Google токена, а саме ім'я, прізвище, електронну адресу. На рисунку 2.8 зображено створення користувача у разі його відсутності в базі даних.

```
async googleLogin({ email, sub, given_name, family_name }: GoogleUserType) : Promise<...> {
  const user : UserEntity = await this.usersService.findUserByEmail(email);

  if (!user) {
    await this.usersService.createUser( createUserInput: {
      email,
      googleId: sub,
      firstName: given_name,
      lastName: family_name,
      password: null,
      gender: null,
    });
  }

  const { access_token : string } = await this.login( {email}: { email });

  return {
    access_token,
  };
}
```

Рисунок 2.8 – Авторизація – реєстрація користувача за допомогою Google

Якщо користувач не може пригадати свій пароль, він має змогу відновити його. У такому разі при відправці відповідного запиту, сервер згенерує тимчасовий код для скидання паролю, який буде надісланий на вказану користувачем електронну адресу. Далі юзер вказує у відповідне поле мобільного застосунку код, та при валідності його, зможе створити новий пароль для його облікового запису. Методи для відновлення та оновлення паролю зображені на рисунку 2.9.

```

async resetPassword(email: string) : Promise<void> {
    const user : UserEntity = await this.usersService.findUserByEmail(email);

    if (!user) throw new NotFoundException( objectOrError: 'User not found');

    const token : string = this.speakeasyService.generateTotp(user.email);
    await this.usersService.updateUserById(user.id, updateUserDto: { ...user, token });

    return this.mailService.sendUserResetCode(user, token);
}

async verifyResetCode(email: string, code: string) : Promise<boolean> {
    const user : UserEntity = await this.usersService.findUserByEmail(email);

    if (!user) throw new NotFoundException( objectOrError: 'User not found');
    if (user.token !== code) throw new NotFoundException( objectOrError: 'Invalid code');

    return this.speakeasyService.verifyTotp(user.email, code);
}

async updatePassword(token: string, { email, password } : Pick<UserEntity, 'email' | 'password'>) : Promise<void> {
    const isCodeValid : boolean = await this.verifyResetCode(email, token);

    if (!isCodeValid) throw new NotFoundException( objectOrError: 'Invalid code');

    const hashedPassword : string = await this.passwordService.hashPassword(password);
    await this.usersService.updateUserByEmail(email, updateUserDto: { password: hashedPassword, token: null });
}

```

Рисунок 2.9 – Методи відновлення та оновлення паролю

Так як один з етапів відновлення паролю є підтвердження через тимчасовий код, який відправляється на електронну адресу користувача. Для відправки таких повідомлень використано сервіс Mailer [23] та для формування листа та надання йому стилів було використано шаблонізатор Pug [24]. На рисунку 2.10 та рисунку 2.11 зображено метод для відправки повідомлень та шаблон тіла повідомлення відповідно.

```

@Injectable()
export class MailService {
  constructor(private mailerService: MailerService) {}

  async sendUserResetCode(user: UserEntity, resetCode: string) : Promise<void> {
    await this.mailerService.sendMail( {
      to: user.email,
      from: process.env.MAIL_USER,
      subject: 'Password Reset',
      template: './password-reset.pug',
      context: {
        code: resetCode,
      },
    });
  }
}

```

Рисунок 2.10 – Сервіс відправки повідомлень на пошту користувачам

```

<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title Password Reset>
    <style...>
  </head>
  <body>
    <div class="container">
      <h1 Password Reset>
      <p>You requested to reset your password. Please use the following code to reset your password:<br>
      <p>Code: {code}<br>
      <p>If you didn't request a password reset, you can ignore this email.<br>
      <p>Regards,<br>
      <p>Your MKP app
    </div>
  </body>
</html>

```

Рисунок 2.11 – Шаблон повідомлення для скидання паролю

Далі після всіх етапів успішної авторизації або реєстрації користувачу надається доступ до платформи. У першу чергу він отримає різні по категоріям рецепти відповідно до його мовного параметру. Розглянемо роботу методів для отримання передогляду рецептів по певному критерію, що зображено на рисунках 2.12 – 2.14.

На сервер вхідними даними для методу `GetPreciewRecipesByCategoryId` є ідентифікатор категорії, параметри для пагінації та мови, які допоможуть відфільтрувати рецепти для правильної відповіді. Якщо користувач не вказує

мовний параметр, автоматично пошук буде відбуватися по українським записам.

Для роботи з сутностями бази даних реалізуються сервіси – репозиторії, які працюють виключно для будування запитів. Тому після валідації вхідних даних вони передаються до `RecipeRepository`, що у відповіді отримуємо відповідні рецепти. Оскільки вони містять переклади, ці записи утворюють велику вкладеність об’єктів, тому для зменшення даних у відповідях від серверу розроблено сервіси для модифікування вихідних даних, для більш зручної роботи з ними на клієнтській частині.

```
@Get( path: 'getPreviewRecipesByCategoryId/:id')
async getPreviewRecipesByCategoryId(
  @Param( property: 'id') id: string,
  @Query() queryParams: Partial<PaginationParams & LangParam>,
) : Promise<ClientPreviewRecipeDTO... {
  const { lang : LangEnum , ...result : {offset?: number, limit?: numb... } = queryParams;

  return this.recipeService.getPreviewRecipesByCategoryId(id, parameters: {
    lang: getExistingLangCode(lang),
    ...result,
  });
}
```

Рисунок 2.12 – Контролер для обробки запиту

```
async getPreviewRecipesByCategoryId(
  categoryId: string,
  parameters: Partial<PaginationParams & LangParam>,
) : Promise<ClientPreviewRecipeDTO[]> {
  const previewRecipesByCategoryId : RecipeEntity[] = await this.recipeRepository.previewRecipesByCategoryId( parameters: {
    categoryId,
    ...parameters,
  });

  return this.recipeMapper.entityToClientPreviewRecipesDTO(previewRecipesByCategoryId);
}
```

Рисунок 2.13 – Сервіс для отримання передогляду рецептів по категорії

```

async previewRecipesById(
  parameters: Partial<PaginationParams & LangParam & { categoryId: string }>,
): Promise<RecipeEntity[]> {
  const { lang : LangEnum , limit : number , offset : number , categoryId : string } = parameters;

  return await this.recipeRepository
    .createQueryBuilder( alias: 'recipe')
    .innerJoinAndSelect(
      {
        property: 'recipe.categoryToRecipes',
        alias: 'categoryToRecipe',
        condition: 'categoryToRecipe.categoryId = :categoryId',
        parameters: { categoryId },
      }
    )
    .innerJoinAndSelect( {property: 'recipe.translations', alias: 'translations'})
    .where( where: 'translations.languageCode = :languageCode', parameters: { languageCode: lang })
    .andWhere( where: 'recipe.isPublished = true')
    .select( selection: ['recipe.id', 'recipe.imageUrl', 'translations.title'])
    .offset(offset)
    .limit(limit)
    .getMany();
}

```

Рисунок. 2.14 – Репозиторій для формування запиту до БД для отримання рецептів

Для роботи з більшою кількістю параметрів для пошуку та фільтрацій рецептів створено додаткові методи в RecipeRepository. Для побудови більш складніших вибірок в базі даних. Метод getFilteredPreviewRecipes для вибірок по багатьом параметрам зображено на рисунку 2.15.

```

async getFilteredPreviewRecipes(
  parameters: Partial<PaginationParams & LangParam & SearchParam & FilterRecipeParams>,
): Promise<RecipeEntity[]> {
  const { searchQuery : string , lang : LangEnum , offset : number , limit : number , ...filterParams : {tagIds?: number[]}}

  const queryBuilder : SelectQueryBuilder<RecipeEntit... = this.recipeRepository
    .createQueryBuilder( alias: 'recipe')
    .innerJoinAndSelect( {property: 'recipe.translations', alias: 'translations'})
    .innerJoinAndSelect( {property: 'recipe.ingredientToRecipes', alias: 'ingredientToRecipes'})
    .where( where: 'translations.languageCode = :languageCode', parameters: { languageCode: lang })
    .andWhere( where: 'recipe.isPublished = true');

  if (searchQuery) {
    queryBuilder.andWhere( where: 'LOWER(translations.title) LIKE LOWER(:searchQuery)', parameters: {
      searchQuery: `${searchQuery}%`,
    });
  }

  if (Array.isArray(filterParams.categoryIds)) {
    queryBuilder.innerJoinAndSelect( {property: 'recipe.categoryToRecipes', alias: 'categoryToRecipes'});
    queryBuilder.andWhere( where: 'categoryToRecipes.categoryId IN (:...categoryIds)', parameters: {
      categoryIds: filterParams.categoryIds,
    });
  }
}

```

Рисунок 2.15 – Метод getFilteredPreviewRecipes для вибірок рецептів

Користувач має змогу додати свій власний рецепт для персонального використання по замовченню або для подальшої публікації. Для створення користувач описує всю необхідну інформацію та додає фото, інгредієнти та т.д. На сервері картинка вантажиться на хмарний сервіс Cloudinary [25] для хостингу. Інша інформація зберігається в базі даних, та список інгредієнтів валідується формуючи нові зв'язки до рецепту або створюючи нові записи інгредієнтів за нової назви. Контролер обробки створення рецепту зображено на рисунку 2.16.

```
@Post()
@UseInterceptors(FileInterceptor('image', multerOptions))
async createRecipe(
  @Body() createRecipeDto: CreateRecipeInput,
  @UploadedFile() image: Express.Multer.File,
  @Req() req: Request & { user: Omit<UserEntity, 'password'> },
  @Query() { lang }: Partial<LangParam>,
) : Promise<RecipeEntity> {
  if (!image) {
    throw new BadRequestException( objectOrError: 'Image file is required. ');
  }

  const uploadedImage : UploadApiResponse | UploadApiE... = await this.cloudinaryService.uploadImage(image).catch(() : void => {
    throw new BadRequestException( objectOrError: 'Invalid file type. ');
  });

  const ... = createRecipeDto;

  const categoryIds : number[] = Array.from(JSON.parse(createRecipeDto.categoryIds)).map(
    (id: string) => +id,
  );

  return await this.recipeService.createRecipe({userId: req.user.id...});
}
```

Рисунок 2.16 – Контролер обробки створення рецепту

Для управління своїми планами та трапезами, користувач має можливість додавати потрібні рецепти до трапез, що в свою чергу додаються до планів по днях. Для цього на сервер приходять різноманітні дані, наприклад оновлені параметри трапез (час приготування, кількість порцій), категорії планів. Після кожного оновлення даних, відбувається оновлення дочірніх залежностей, підрахунок загальних інгредієнтів, розрахунок коефіцієнтів для визначення кількості порцій. Метод оновлення планів зображено на рисунку 2.17.

```

async updatePlanWithMealPlans(
  userId: number,
  planId: number,
  updatePlanWithMealPlans: UpdatePlanWithMealPlans,
): Promise<void> {
  const plan : PlanEntity = await this.planRepository.findOne( options: { where: { id: planId, user: { id: userId } } });

  if (!plan) throw new Error('Plan not found');

  await this.planRepository.update(planId, partialEntity: {
    date: updatePlanWithMealPlans.date.toString(),
  });

  await Promise.all(
    updatePlanWithMealPlans.mealPlans.map(async mealPlan : CreateMealPlanWithId => {
      const { id : number , ...rest : {name: string, preparationStar... } = mealPlan;

      if (!id) {
        return await this.mealPlanService.createMealPlan(rest, planId);
      }

      await this.mealPlanService.updateMealPlan(rest, id);
    }),
  );

  if (Array.isArray(updatePlanWithMealPlans?.deletedMealCardIds)) {
    await Promise.all(
      updatePlanWithMealPlans.deletedMealCardIds.map(async mealCardId : number => {
        await this.mealPlanService.deleteMealPlan(mealCardId);
      }),
    );
  }
}

```

Рисунок 2.17 – Метод оновлення планів

Користувач на вже створені плани може отримати згенеровані поради за допомогою штучного інтелекту. Для цього на сервері зберігаються промти певних типів, наприклад за допомогою Generate knowledge prompting [26], отримуються точні відповіді для отримання порад по вхідним назвам рецептів та категорій (умов приготування), наприклад поради по гігієні, способу зберігання готової продукції або транспортування.

Цей тип промту описує приклади готових рішень, вказуючи як має відповідати штучний інтелект. За допомогою 2-3 прикладів можна добитися найбільш очікуваного результату для наступних кроків парсингу. На рисунку 2.18 та рисунку 2.19 зображено метод для генерування порад та утворення промту відповідно.

```

async getCookingTipsInUkrainian(
  recipeNames: string[],
  categoryNames: string[],
): Promise<{ [key: string]: string[] }> {
  const prompt : string = cookingTipsInUkrainianPrompt(recipeNames, categoryNames);

  const response : OpenAI.Chat.Completions.ChatCo... = await this.openai.chat.completions.create( body: {
    messages: [{ role: 'user', content: prompt }],
    model: this.model,
    max_tokens: 2000,
    temperature: this.temperature,
  });

  const generatedAdvices : string = response.choices[0].message.content;

  const categories : string[] = generatedAdvices.split( separator: '\n\n');
  const result: { [key: string]: string[] } = {};

  categories.forEach(category : string => {
    const [title : string , ...items : string[] ] = category.split( separator: '\n');

    if (title && items.length > 0) {
      result[title] = items.filter(item : string => item.trim()).map(item : string => item.trim());
    }
  });
}

```

Рисунок 2.18 – Метод для генерування порад

```

export const cookingTipsInUkrainianPrompt = (recipeNames: string[], categoryNames: string[]) : string => {
  return `Завдання: Згенеруй по 3-4 загальні поради (без підпунктів) для приготування по таким категоріям як, методи зберігання
Вхідні страви: Борщ, Курячий суп з овочами, картопля з грибами на сковороді
Умови приготування: Без кухонного успадкування, довготривале зберігання
Відповідь:
Методи зберігання:
1. Борщ можна заморозити, розлити по порціях і зберігати в морозилці до 3 місяців.
2. Суп можна зберігати в герметичному контейнері в холодильнику до 3-5 днів або заморозити на 2-3 місяці.
3. Страву можна зберігати в холодильнику до 3 днів. Можна також заморозити, розклавши по порціях, до 2 місяців.

Гігієна:
1. Ретельно мийте руки, поверхні, посуд і інструменти перед приготуванням.
2. Використовуйте окремі дошки для сирих і готових продуктів.
3. Готуйте їжу при безпечних температурах (вище 70°C).
4. Швидко охолоджуйте і зберігайте залишки їжі в холодному місці.

Методи транспортування:
1. Перевозьте готові страви в термоконтейнерах або сумках-холодильниках, щоб зберегти температуру.
2. Для заморожених продуктів використовуйте термоізовані сумки з холодними елементами.
3. Намагайтеся мінімізувати час транспортування та уникайте зберігання страв при кімнатній температурі.
Завдання: Згенеруй поради для приготування по таким категоріям як, методи зберігання, гігієна, методи транспортування
Вхідні страви: Гречаний суп з м'ясом, печена картопля на вогнищі
Умови приготування: В польових умовах, швидке приготування
Відповідь:
Методи зберігання:
1. Гречаний суп з м'ясом можна приготувати в герметичному контейнері з термоса, що дозволить зберігати тепло і зберігати його
2. Печену картоплю можна загорнути у фольгу і зберігати в термосумці або термоконтейнері до 3-4 годин.
3. Якщо є можливість, зберігайте інгредієнти окремо і з'єднайте їх безпосередньо перед приготуванням, щоб зберегти свіжість.
4. Використовуйте термопакети з холодними елементами для зберігання продуктів до приготування, якщо температура навколишнього

```

Рисунок 2.19 – Приклад промту для отримання порад

Створюючи методи на сервері для отримання даних, часто потрібно було враховувати мовний параметр. Вони додавались до запитів до бази даних та надавали відповідний результат згідно обраною мовою додатку.

Щодо варіантів реалізації локалізацій на клієнтській частині, було взято за основу ідею використання json файлів, що включають всі статичні тексти, які відображаються користувачу. Кожен файл містить записи для відповідної мови та спільного між ними являються назви полів, які надають однакові посилання до потрібних текстів. Загальна схема роботи полягає в обранні відповідного файлу та відображення цих текстів при рендері сторінок додатку. На рисунку 2.20 зображено приклад json файлу для української мови додатку та на рисунку 2.21 зображено метод застосування на платформі.

```
{
  "introduction": {
    "skip": "Пропустити",
    "get-started": "Почати роботу",
    "already-have-an-account": "Уже маю обліковий запис",
    "slide1": {
      "title": "Military Kitchen Planner",
      "description": "Додаток допомагає військовим планувати свої трапези, враховуючи калорійність, поживність та легкість приготування",
    },
    "slide2": {
      "title": "Графік прийому їжі",
      "description": "Військові можуть створювати та переглядати свій щоденний графік прийому їжі, щоб забезпечити оптимальне харчування",
    },
    "slide3": {
      "title": "Рецепти та харчові обмеження",
      "description": "\"Military Kitchen Planner\" надає доступ до різноманітних рецептів, враховуючи військові дієтичні обмеження",
    },
    "slide4": {
      "title": "Публікування своїх рецептів",
      "description": "Додаток дозволяє військовим створювати свої рецепти та за потреби публікувати їх для розвитку своєї кулінарії",
    },
    "slide5": {
      "title": "Створення облікового запису",
      "description": "\"МКР\" надає можливість користувачам створити власний обліковий запис, щоб зберігати їхні налаштування, історію та рецепти",
    }
  },
  "registration": {
    "sign-up": "Реєстрація",
    "description": "Після заповнення даних у полях нижче, ви отримаєте доступ до всіх функцій додатку.",
    "email": "Електронна пошта",
    "password": "Пароль",
    "password-placeholder": "Введіть пароль",
    "confirm-password": "Підтвердити пароль",
    "confirm-password-placeholder": "Підтвердіть пароль",
    "continue": "Продовжити"
  },
  "sign-in": {
    "title": "Гайда заходь. Тебе не вистачало!",
    "forgot-password": "Забули пароль?",
  }
}
```

Рисунок 2.20 – Приклад json файлу для української мови додатку

```

<ScreenTitle
  title={i18n.t( scope: 'registration.sign-up')}
  description={i18n.t( scope: 'registration.description')}
/>

<View className="pt-[56px] gap-y-[12px]">
  <View>
    <FormTextInput
      name="email"
      label={i18n.t( scope: 'registration.email')}
      placeholder="e.g. bohdan@gmail.com"
      control={control}
    />
  </View>

  <View>
    <FormPasswordInput
      name="password"
      label={i18n.t( scope: 'registration.password')}
      placeholder={i18n.t( scope: 'registration.password-placeholder')}
      control={control}
    />
  </View>

```

Рисунок 2.21 – Метод застосування перекладів за допомогою i18n

Підсумовуючи, було створено діаграму компонентів мобільної-платформи для планування трапез (рис. 2.22) та визначено основні компоненти збору.

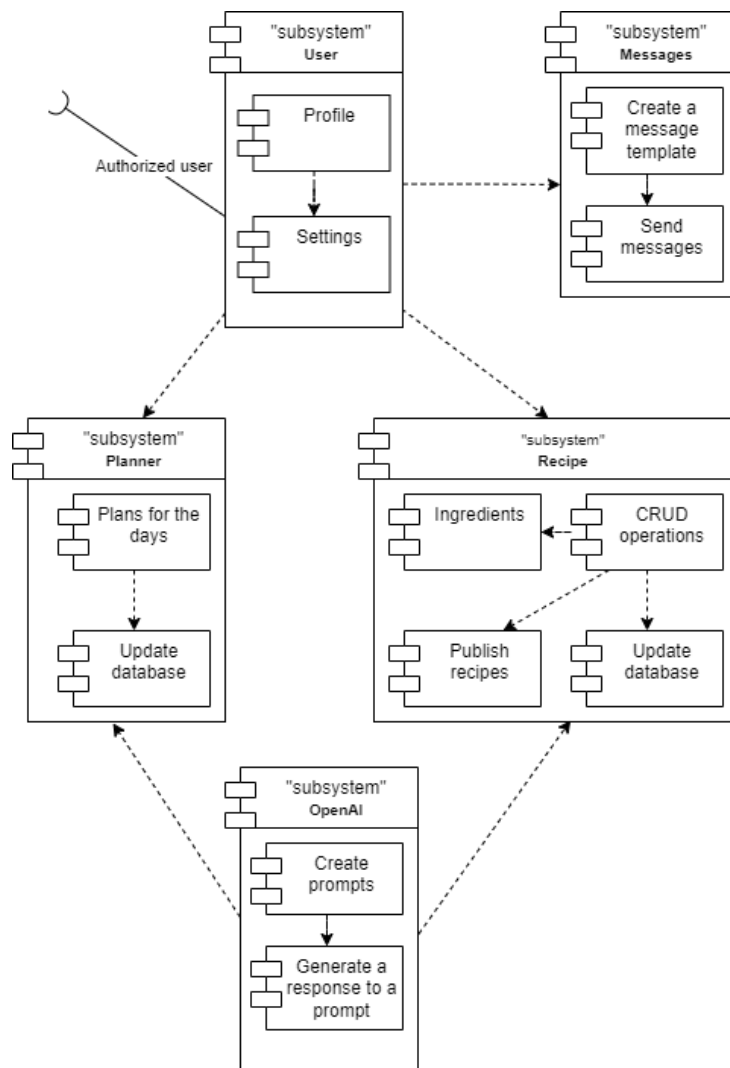


Рисунок 2.22 – Діаграма компонентів

На діаграмі визначені такі елементи:

- User – підсистема, яка відповідає за створення, редагування, видалення користувачів та їх налаштувань;
- Planner – відповідає за взаємодію з планами надаючи інструменти обробки, збереження та групування даних;
- Recipe – надає методи для збереження, оновлення рецептів та їх залежностей, також маніпулює публікаціями та перекладами записів;
- Messages – підсистема відповідає за утворення шаблонів повідомлень на електронну пошту та відправку їх користувачам;
- OpenAI – відповідає за створення промптів, генерування порад по вхідним параметрам, перекладам для рецептів та інгредієнтів.

Висновки до другого розділу

У процесі проектування мобільної платформи планування трапез були проаналізовані усі випадки використання програмного продукту відповідно до вимог програмного забезпечення. На основі отриманих результатів була спроектована структура платформи. Проектування включало моделювання структури проекту та взаємодії між їх складовими компонентами.

У другому пункті була реалізована база даних відповідно до вимог реляційної моделі, яка забезпечує зберігання даних та доступ до них в потрібний час. База даних складається з 21 таблиці, в яких може зберігатися вся необхідна інформація, для стабільної роботи мобільної платформи планування трапез.

В останньому пункті детально був розглянутий реалізований код клієнтського та серверного додатків. Основна увага була приділена функціям обробки та збереження даних, виведення їх на клієнтську частину, формування перекладів та генерування порад штучним інтелектом, варіанти авторизації, основні операції роботи з базою даних, а також функції пагінації, пошуку, фільтрації, враховуючи мовні параметри.

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З МОБІЛЬНО-ОРІЄНТОВАНОЮ СИСТЕМОЮ ПЛАНУВАННЯ ТРАПЕЗ

3.1 Порядок встановлення та налаштування параметрів системи

Діаграма розгортання є візуальним зображенням конфігурації вузлів обробки часу виконання та компонентів, які на них живуть. Це один з типів структурних діаграм, що використовуються для моделювання фізичних аспектів об'єктно-орієнтованої системи, і часто використовуються для моделювання статичного вигляду розгортання системи.

На рисунку 3.1. зображено діаграму розгортання, яка показує конфігурацію вузлів та компонентів системи.

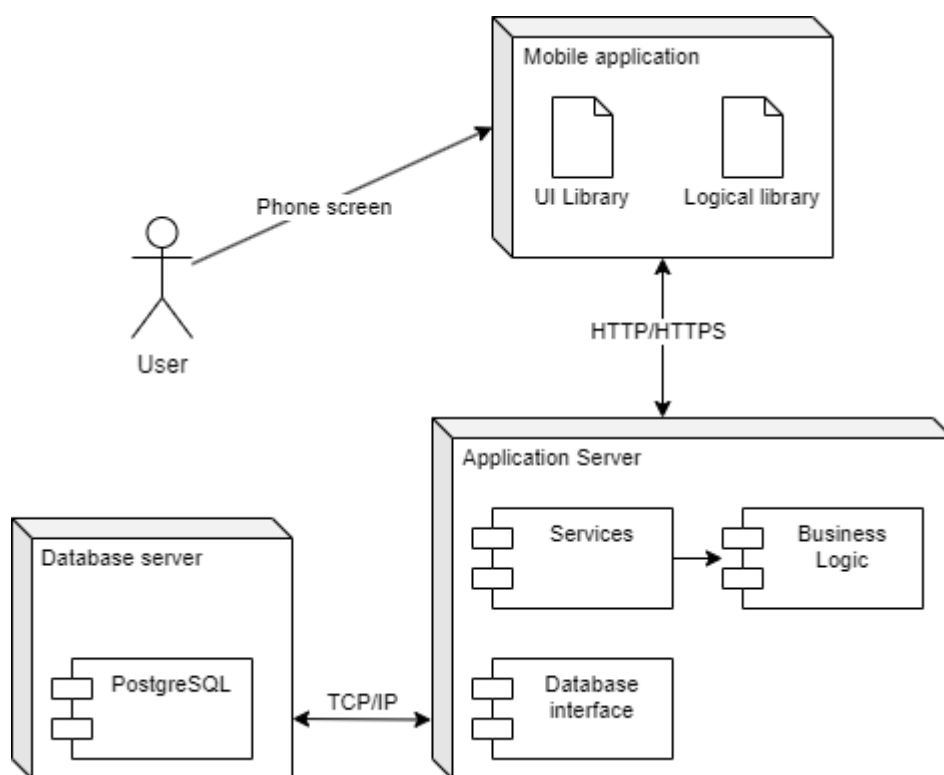


Рисунок 3.1 – Діаграма розгортання додатку

Опис процесу розгортання:

- Mobile application – розгортання клієнтської частини додатку, яка містить свою бібліотеку UI компонентів та сервіси з бізнес логікою, що об'єднує дані з серверної частини та візуалізують їх, також надають методи роботи з відправкою даних до серверу;

- Server application – розгортання серверної частини додатку, для зберігання, обробки даних та проведення CRUD операцій. Надає сервіси для відправки електронних повідомлень та взаємодії зі штучним інтелектом;
- Database server – розгортання бази даних, яка взаємодіє з сервером за допомогою TCP/IP та зберігає всі необхідні данні.

3.2 Структура інтерфейсу та порядок взаємодії системи планування трапез

Під час першого відкриття мобільного застосунку, користувача зустрічає початкова сторінка з короткими описами самого додатку, що зображено на рисунку 3.2. На цьому етапі також беруться мовні параметри зі смартфона користувача та відповідно до них відображаються тексти українською або англійською мовою.

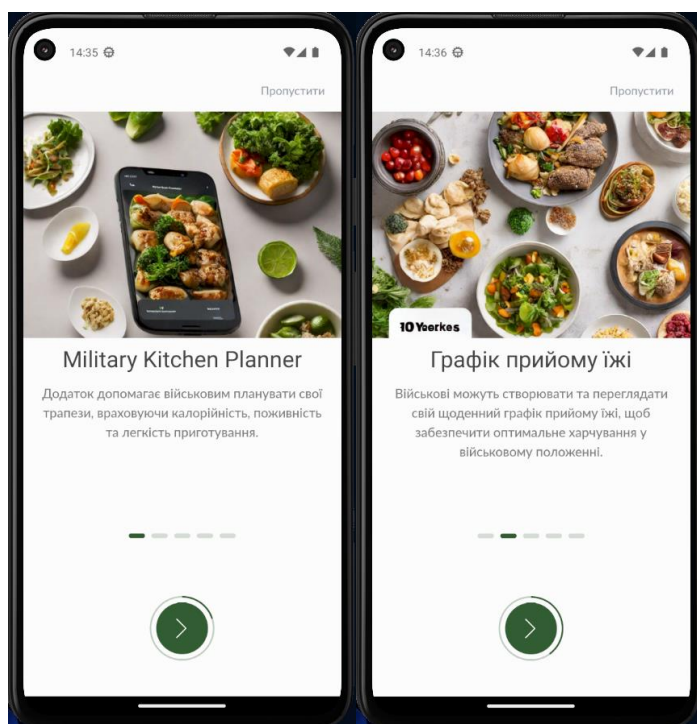


Рисунок 3.2 – Вступна сторінка додатку

На останньому слайді початкової сторінки, користувачу запропонують почати роботу, створивши свій обліковий запис, або авторизувавшись за наявності його (рис. 3.3).



Рисунок 3.3 – Сторінка початку роботи з додатком

Сторінка реєстрації (рис. 3.4) має відповідні валідації при заповненні яких, користувача перевіряють на існування його електронної адреси в системі та складність введеного паролю.

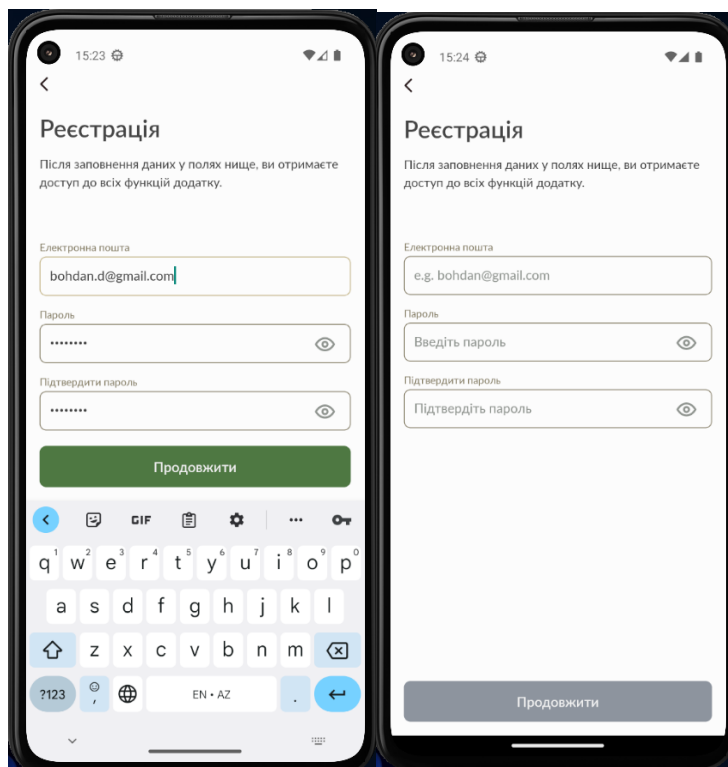


Рисунок 3.4 – Сторінка реєстрації

Якщо у користувача відсутні персональні дані у системі, додаток запропонує заповнити відповідну форму (рис. 3.5)

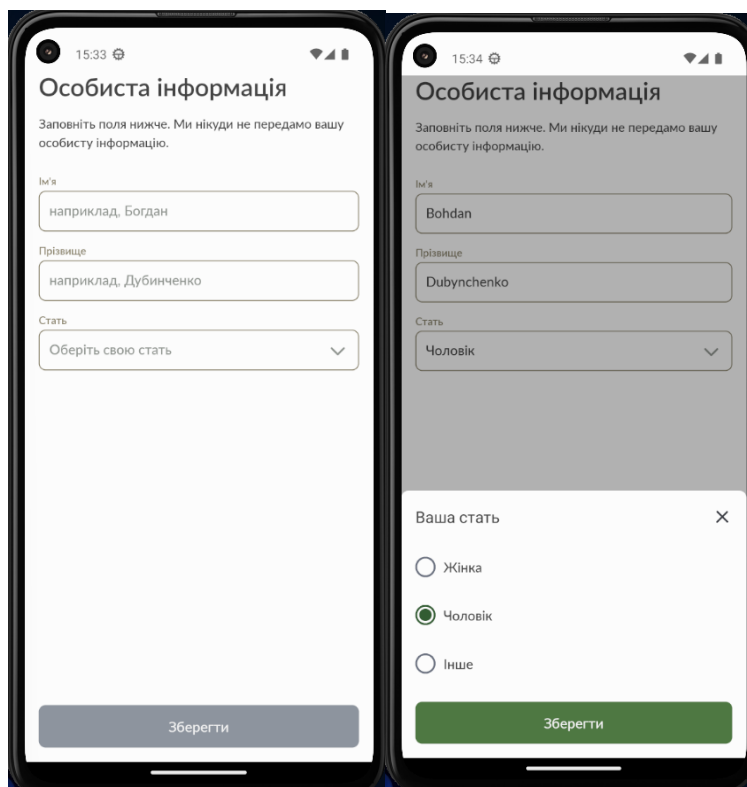


Рисунок 3.5 – Сторінка заповнення особистої інформації

Для авторизації відповідно є своя форма, на котрій користувач має можливість увійти до свого облікового запису за допомогою гугл, або стандартного введення електронної адреси та паролю (рис. 3.6). Під час першого входу через Google, користувача запитає доступ до відповідних його даних.

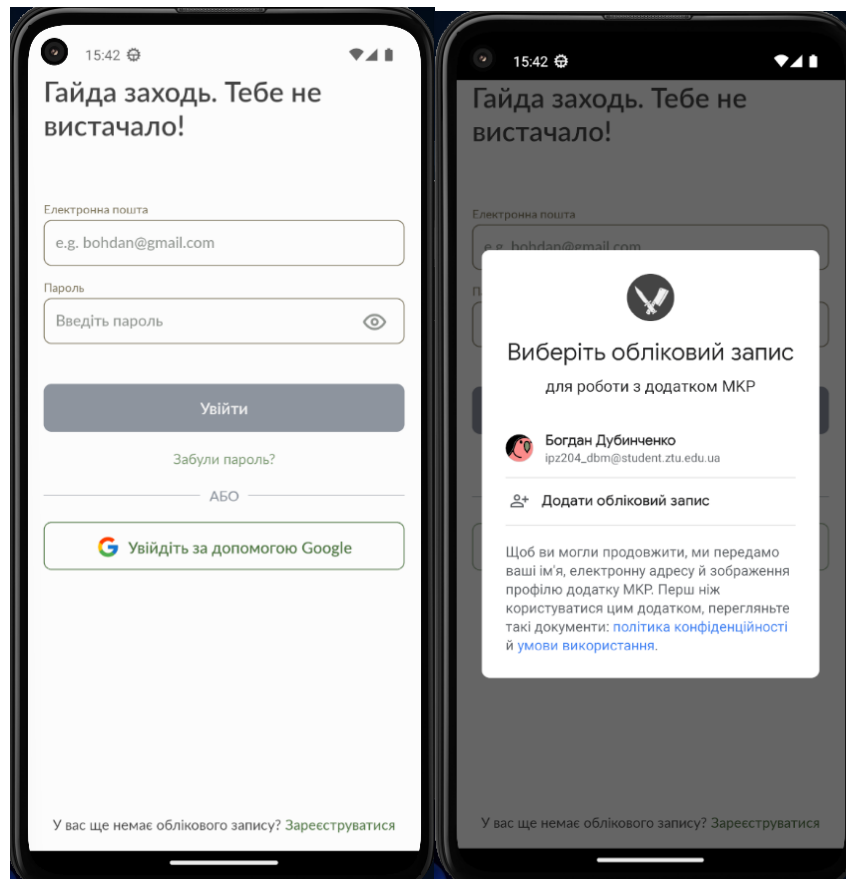


Рисунок 3.6 – Сторінка авторизації та засіб входу за допомогою Google

У разі якщо користувач не може пригадати пароль до свого облікового запису, він має змогу його відновити засобами електронної адреси та тимчасового коду доступу для цієї операції. Після введення адреси пошти, користувач отримає лист, з подальшими інструкціями, у якому він може підтвердити власність свого акаунту та отримати форму для оновлення паролю. Етапи скидання паролю та шаблон повідомлення на електронну адресу зображено на рисунку 3.8 – 3.9 відповідно.

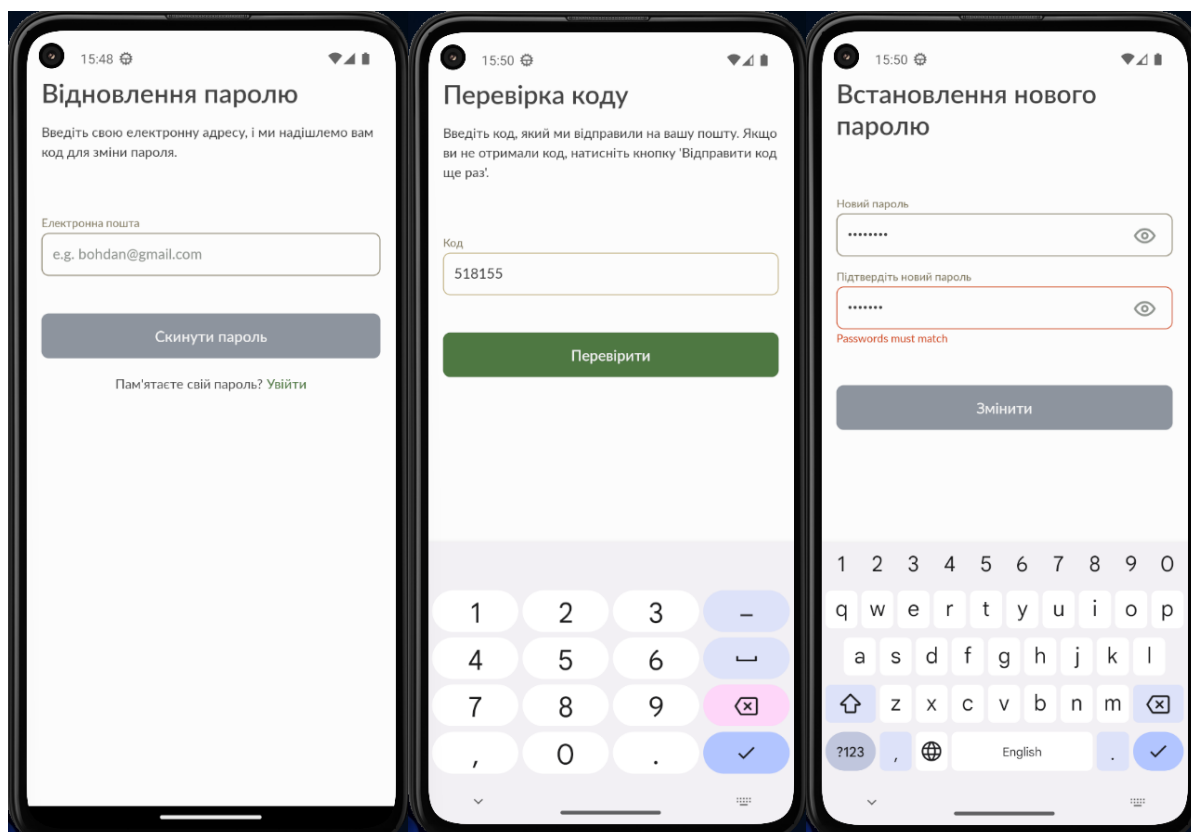


Рисунок 3.7 – Етапи скидання паролю за допомогою тимчасового коду

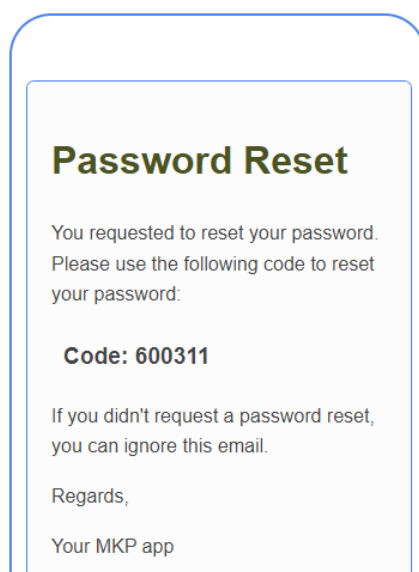


Рисунок 3.8 – Шаблон повідомлення для відправки тимчасового коду

Після успішної авторизації на платформі, користувача зустрічає домашня сторінка з рецептами, що зображена на рисунку 3.9

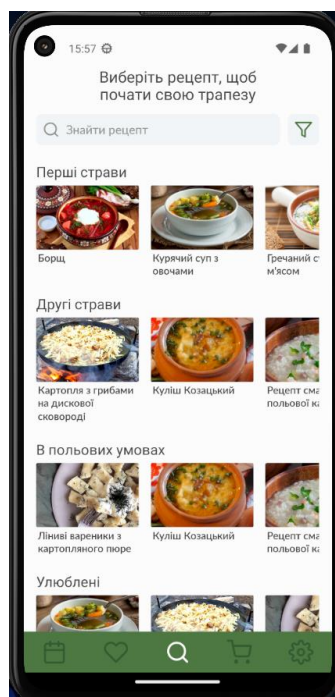


Рисунок 3.9 – Домашня сторінка з рецептам

На даній сторінці користувачу доступно перегляд різних рецептів за категоріями, список своїх вподобаних рецептів, функції пошуку та фільтрацій. З кожним рецептом присутня можливість детально ознайомитись (рис. 3.10), переглянувши його вміст та за потреби додати до розкладу.

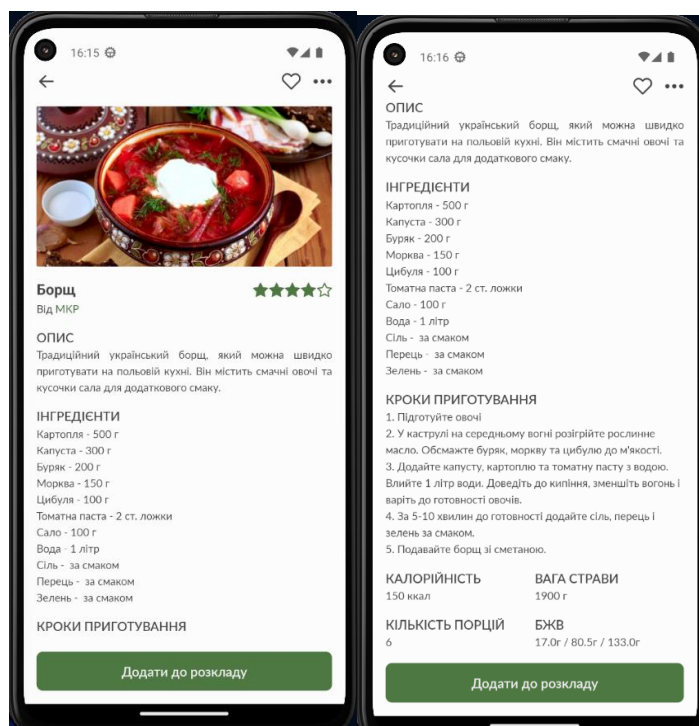


Рисунок 3.10 – Сторінка ознайомлення з обраним рецептом

Наступний розділ (рис. 3.11) призначений для перегляду та планування трапез. При першому переході користувачу відображаються його плани на поточний тиждень, на якому реалізовано редагування, створення, та видалення трапез. Кожен день містить інформацію про трапези та їх налаштування, під кожним яким знаходиться список доданих рецептів. Для перегляду минулих розкладів присутній календар, на якому виділені всі дні з збереженими даними.

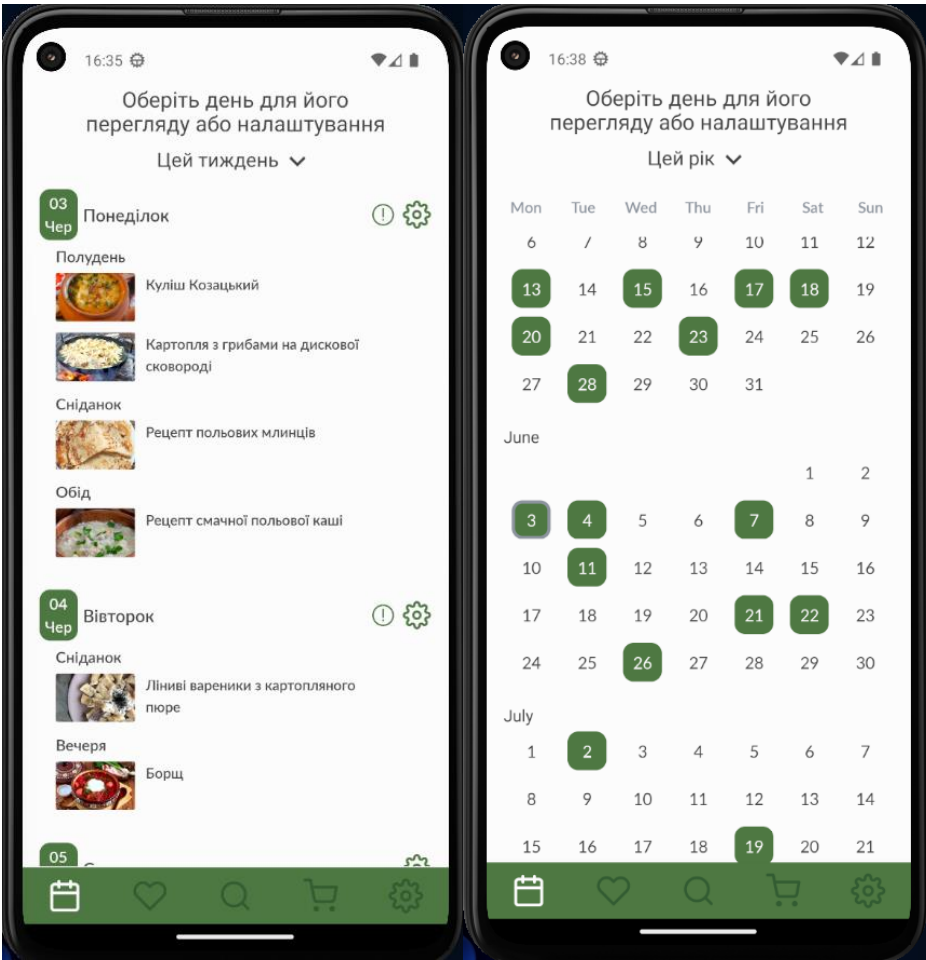


Рисунок 3.11 – Перегляд розкладу трапез по поточному тижню та року

Для налаштування нового дня потрібно натиснути на потрібну дату та користувачу буде показане вікно з внутрішніми налаштуваннями дня (рисунок 3.12). Категорії описують умови перебування кухні, які в подальшому допоможуть отримувати коректні поради. На даній сторінці доступно створення нових трапез, які не обмежені стандартними сніданками чи обідами. Користувач має змогу налаштувати план так як йому зручно.

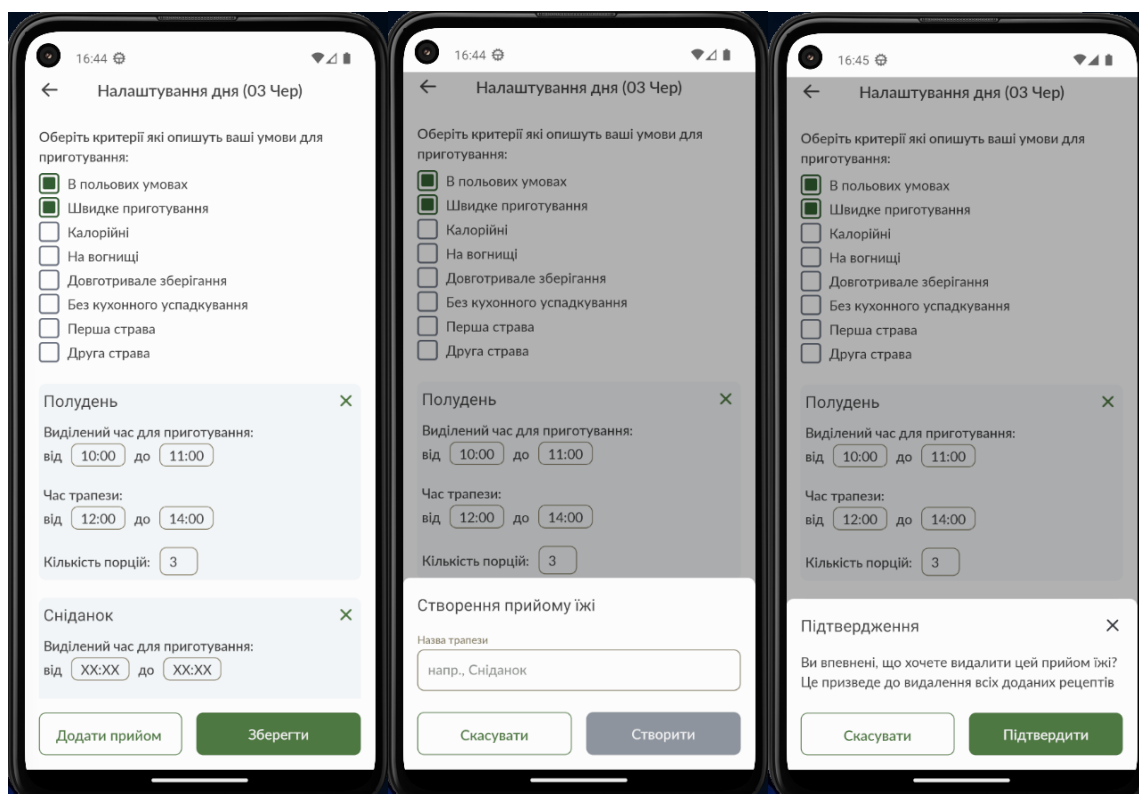


Рисунок 3.12 – Планування трапез для конкретного дня

Щоб додати рецепт до налаштованого дня та трапез, достатньо обрати потрібну страву та натиснути “Додати до розкладу”. Користувач побачить відповідні записи планів до яких він може додати вподобаний рецепт (рис. 3.13). Після чого буде оновлено список покупок враховуючи кількість порцій у плані та рецепту.

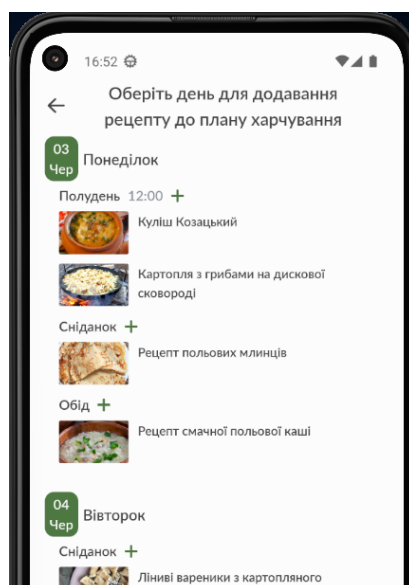


Рисунок 3.13 – Вікно додавання рецепту до плану

Користувачу доступні певні методи для знаходження потрібного рецепту. На платформі доступні різноманітні фільтри та поля для пошуку, які надають більшу можливість для вибірок потрібних записів під потреби юзера (рис. 3.14). Також щоб не шукати кожного разу одну і ту ж страву користувач має змогу вподобати рецепт (рис. 3.15) та знаходити його за окремою категорією.

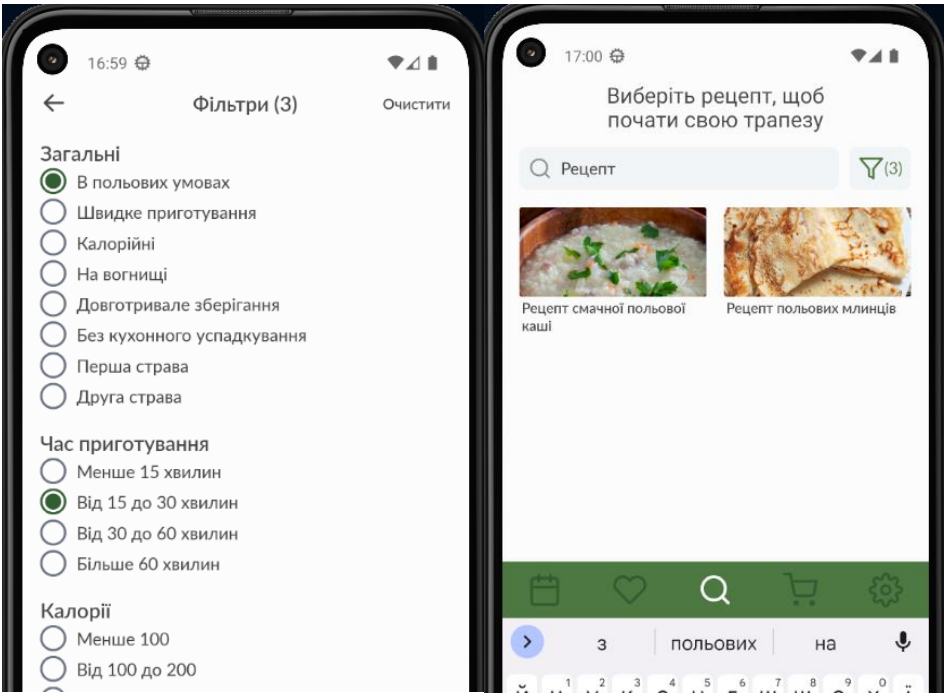


Рисунок 3.14 – Фільтри для пошуку рецептів

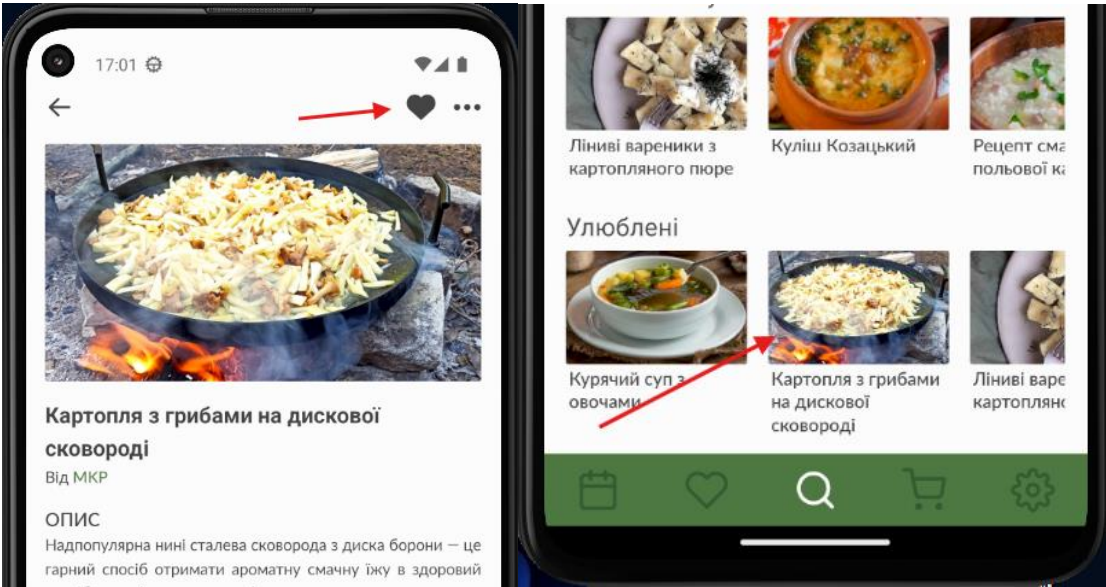


Рисунок 3.15 – Вподобання рецепту

Після додавання різних страв користувачу формуються списки продуктів, які необхідні для кожного запланованого дня (рис. 3.16). На даній сторінці позначаються вже куплені інгредієнти, або за потреби кухар має змогу додати персональні записи, для більш зручного зведеного списку покупок.

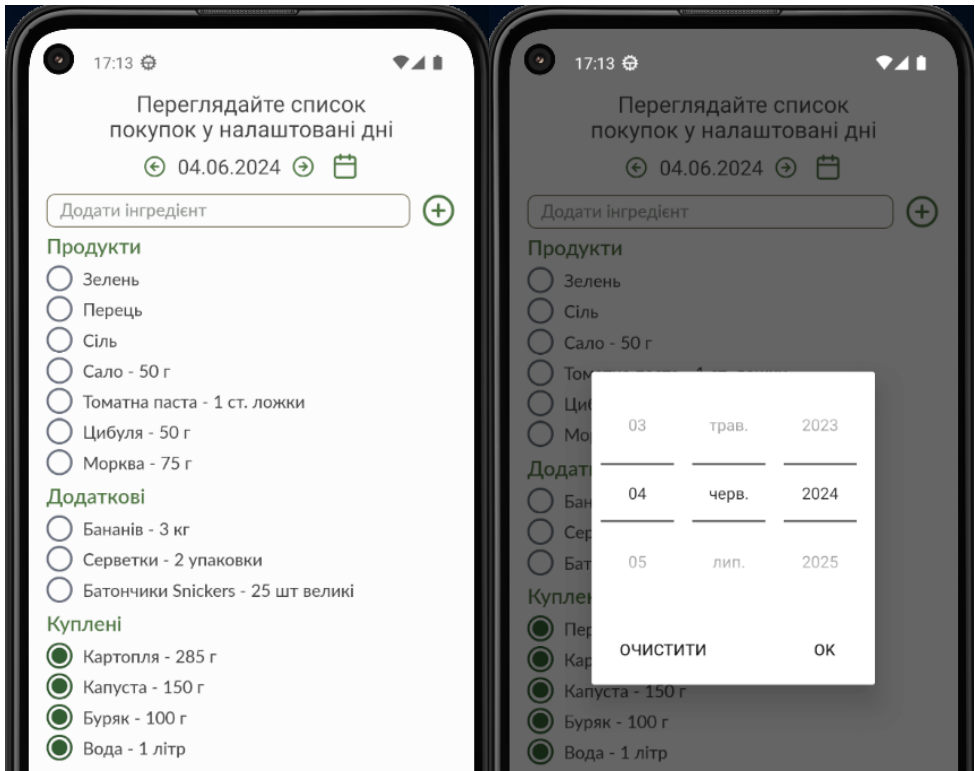


Рисунок 3.16 – Перегляд списку покупок та пошук запланованого дня

Наступна сторінка має функціонал створення, перегляд та публікування власних рецептів (рис. 3.17). Для створення рецепту користувачу потрібно заповнити відповідну форму (рис. 3.18), додавши фото яке можна відредагувати на платформі та інгредієнти які можна знаходити по схожим назвам або додаючи свої нові. Після заповнення всіх полів, форма валідується та автор може зберегти запис щоб в подальшому його використовувати для планування раціонів.

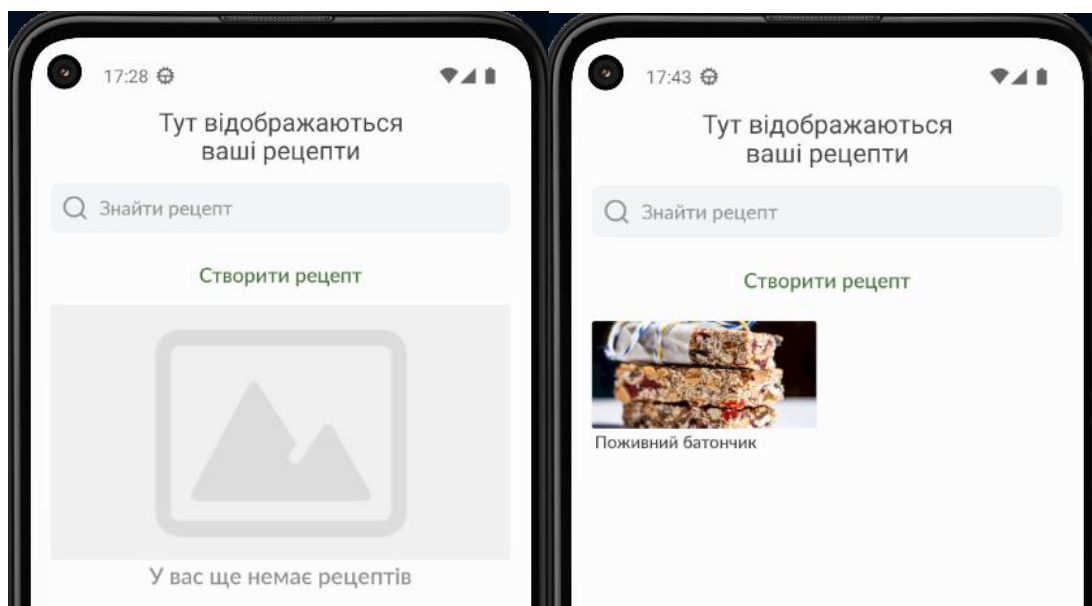


Рисунок 3.17 – Сторінка власних рецептів

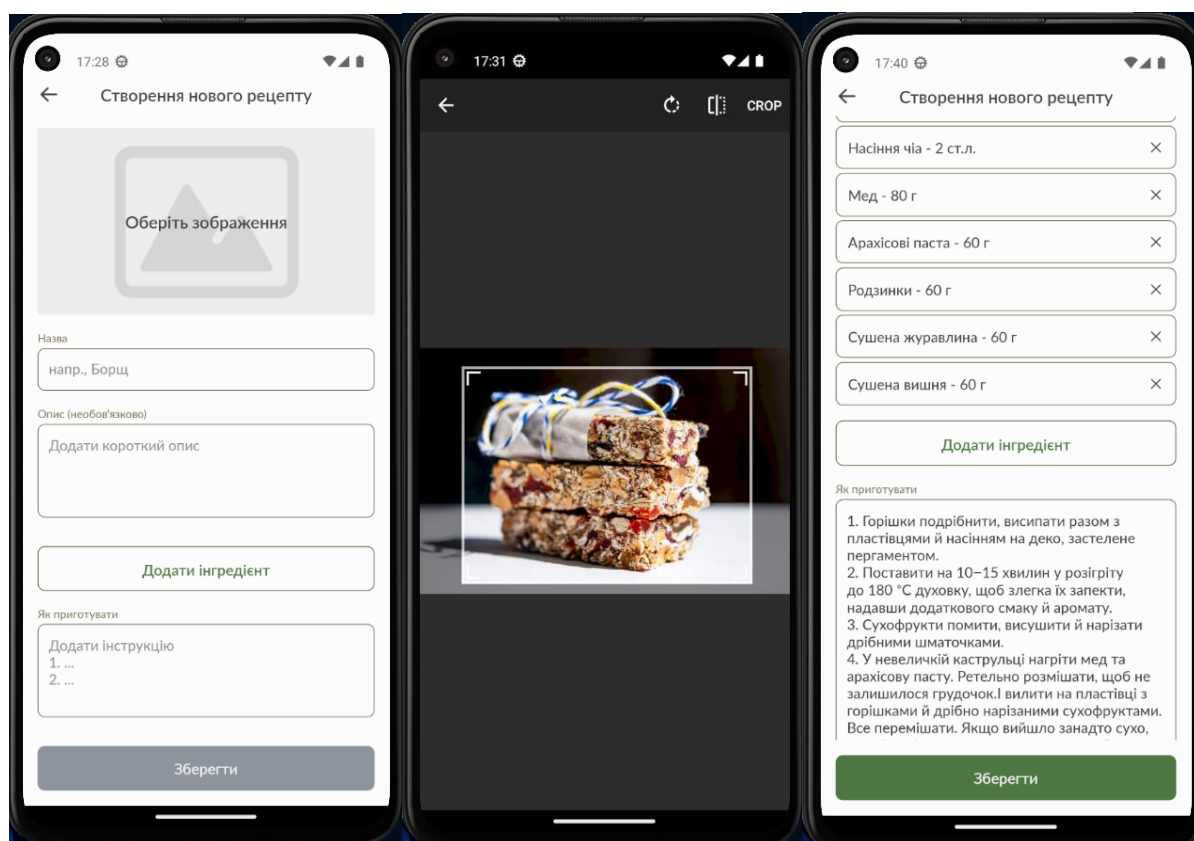


Рисунок 3.18 – Форма створення рецепту

Щоб відобразити свій рецепт іншим користувачам потрібно його опублікувати. Даний етап передбачає створення локалізації під різні мови, цей процес є автоматизований штучним інтелектом, тому користувачу залишається тільки натиснути одну кнопку для цієї дії (рис. 3.19). Та вже за

лічені секунди рецепт матиме англійську та українську локалізацію для використання у публічному просторі.

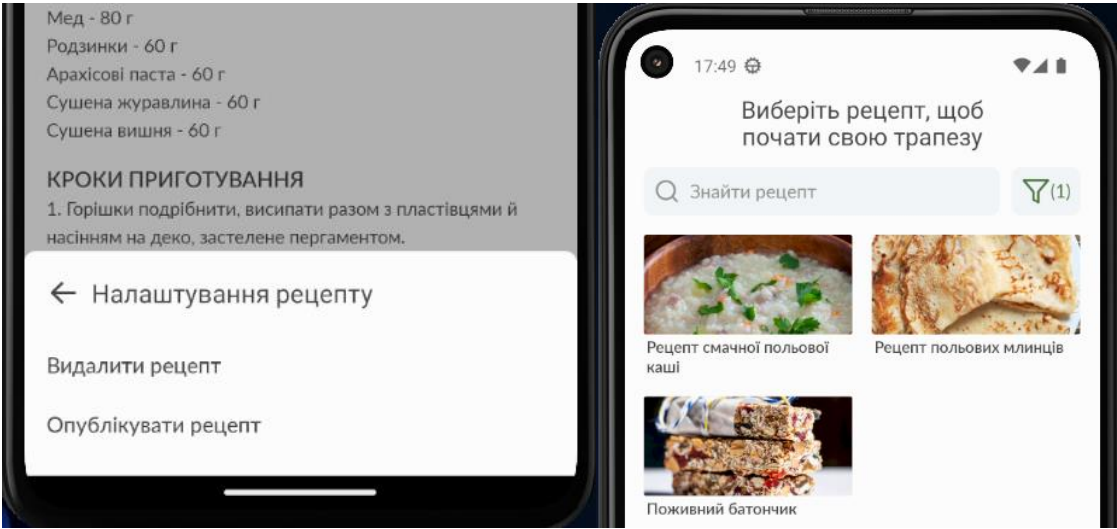


Рисунок 3.19 – Публікація рецепту

З додаткових можливостей користувач має можливість персоналізувати свій профіль, наприклад відредагувати свої данні, змінити мову додатку або ж загальну тему зі світлої на темну (рис. 3.20).

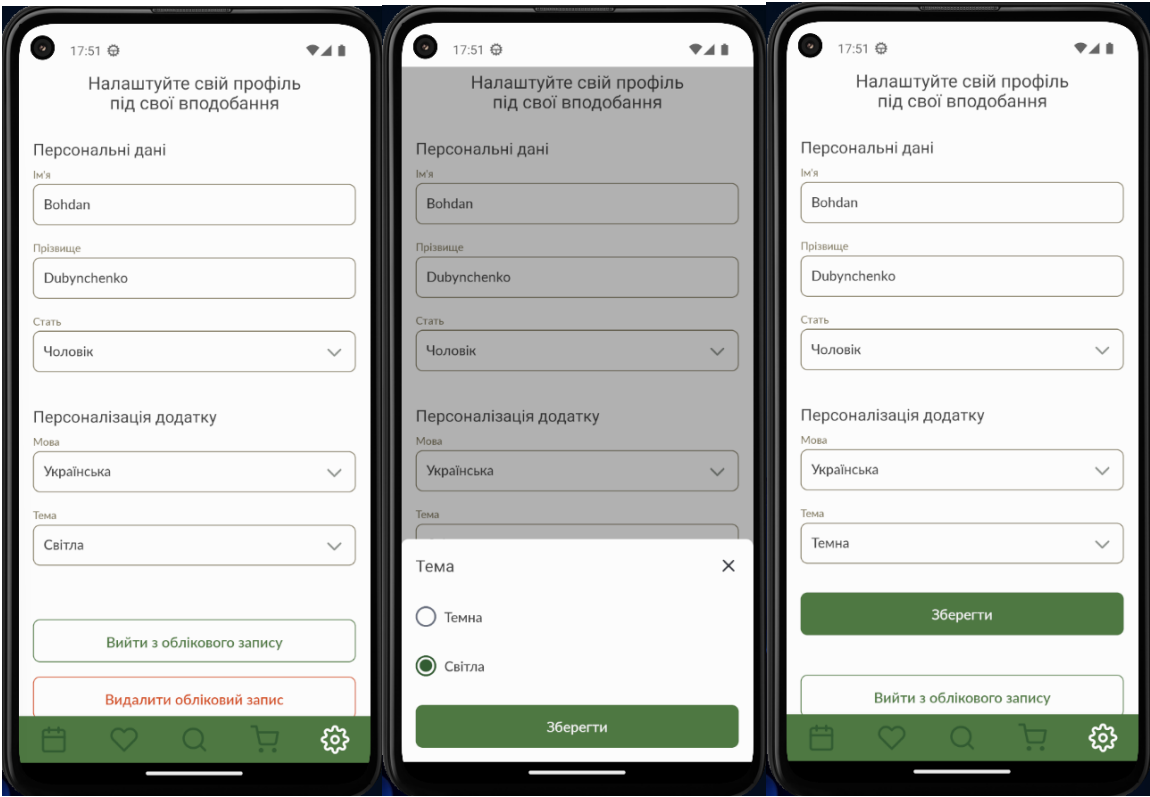


Рисунок 3.20 – Персоналізація додатку

Якщо користувач змінить тему, наприклад на темну, відповідно зміниться візуально весь інтерфейс додатку (рис. 3.21)

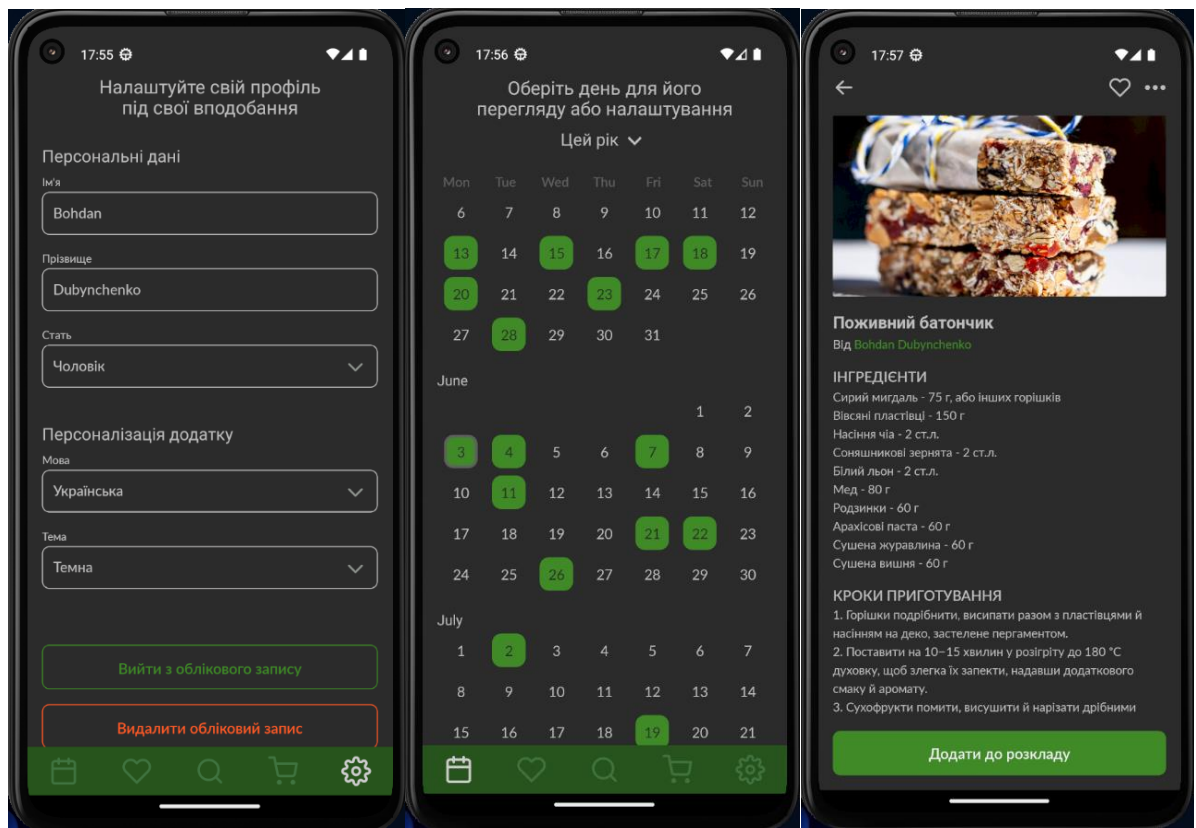


Рисунок 3.21 – Темни режим додатку

Якщо користувач змінює мову платформи, відповідно додаток перезавантажиться задля скидання всіх конфігурацій та зміни їх на іншу мову. Приклад англійської версії застосунку зображено на рисунку 3.22. Також всі дані які приходять зі серверу також матимуть обрану мову.

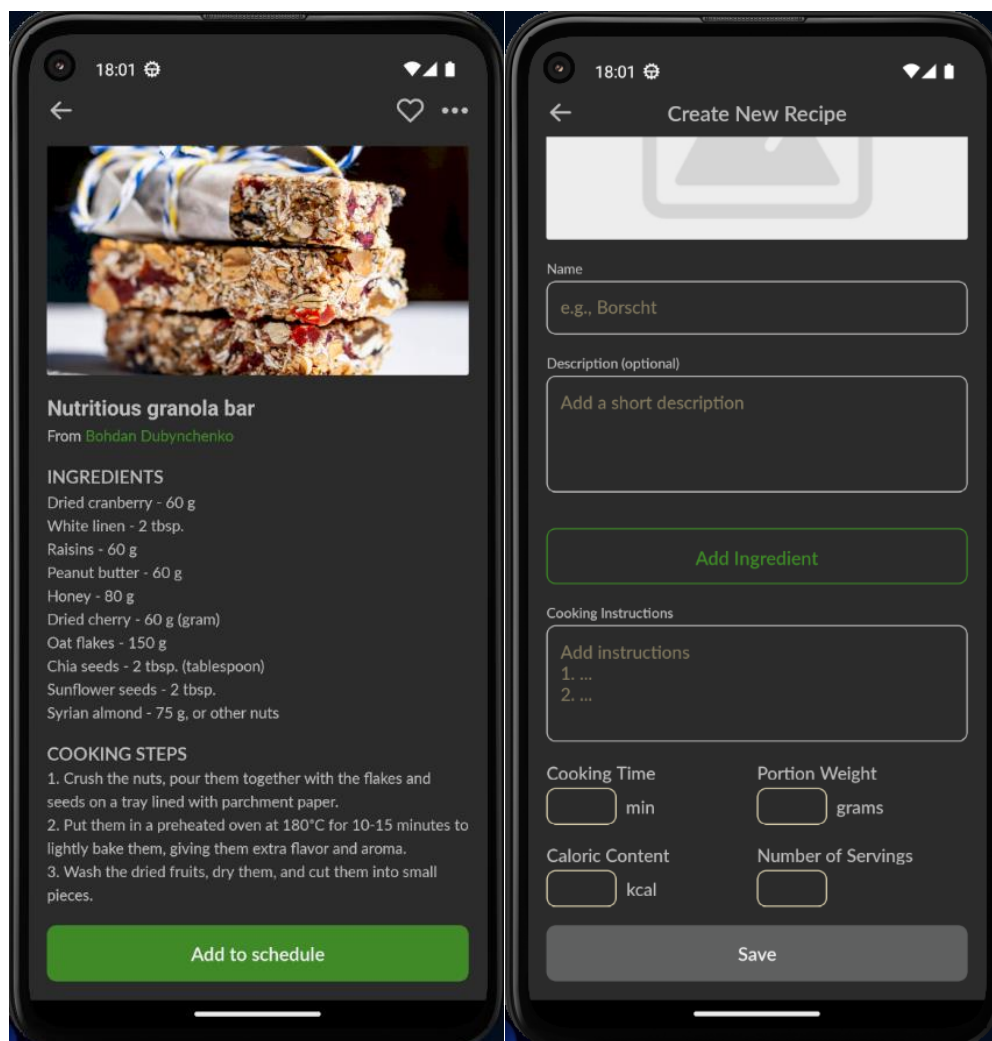


Рисунок 3.22 – Англомовна версія застосунку

Штучний інтелект був застосований не тільки для формування перекладів, а ще й для формування різних порад, які залежать від налаштувань трапез та планів, вони включають умови приготування та обрані страви, надаючи інформацію наприклад про методи зберігання готової продукції, її транспортування, або правила гігієни в відповідних умовах. Приклад згенерованих порад зображено на рисунку 3.23

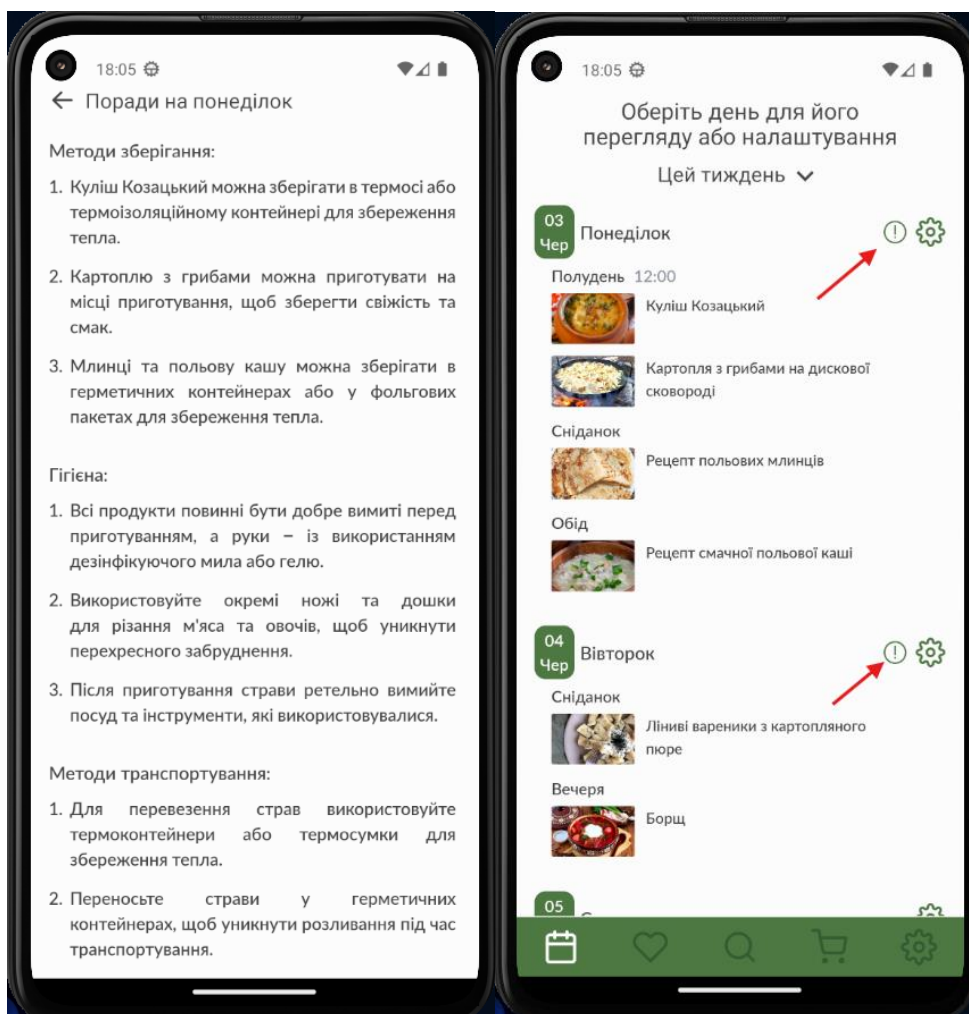


Рисунок 3.23 – Приклад генерування порад по запланованому дню

Остання функція додатку це повне видалення інформації про користувача. За потреби, видалення свого облікового запису можна здійснити на сторінці налаштувань (рис. 3.24).

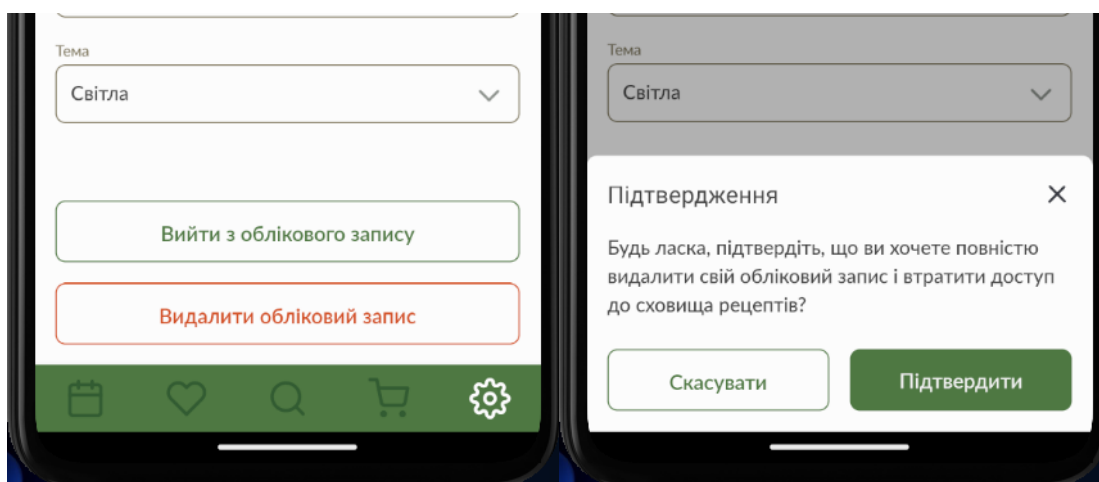


Рисунок 3.24 Видання облікового запису

3.3 Тестування роботи програмного комплексу

Розробка будь-якого програмного продукту завжди несе в собі ризик виникнення помилок. Тому тестування програмного забезпечення є критично важливим для досягнення високої якості додатку. В ході тестування перевіряються різні аспекти роботи програми, такі як правильність вхідних та вихідних даних, коректність роботи функцій та алгоритмів, обробка помилок і відновлення після збоїв і т.д. Метою тестування є виявлення дефектів та помилок в програмному продукті. Це дозволяє їх виправити ще до випуску програми на ринок або введення в експлуатацію на ринок.

Для проведення тестування спочатку створюються так звані "Test Cases". Тест-кейс – це тестовий артефакт, який передбачає виконання певних дій або умов для перевірки конкретної функціональності програмної системи, що розробляється [26].

Представимо нижче сценарії (тестові випадки) для мобільно-орієнтованої платформи у вигляді таблиць.

У таблиці 3.1 зображено приклад валідації даних при проходженні реєстрації, якщо введена електронна адреса існує в системі, користувачу відображається помилка, що щось не так, але без опису для безпеки платформи.

Таблиця 3.1

Test Case 1

Опис тестового випадку		Перевірка реєстрації з існуючим email	Статус		Успішно
Кроки сценарію	1. Перейдіть до мобільної платформи		Тестові дані	1. email = bohdan.d@gmail.com	
	2. Відкрийте форму реєстрації			2. password = bohdan&7	
	3. Заповніть всі необхідні поля			3. confirmPassword = bohdan&7	
	4. Натисніть кнопку “Продовжити”				
Очікуваний результат	Користувачу відображається помилка, що email або пароль невірні		Фактичний результат		Як і очікувалося

Наступний випадок демонструє успішне проходження авторизації (табл. 3.2), після якої успішним результатом є відображення домашньої сторінки або форми персональних даних при відсутності їх у користувача.

Таблиця 3.2

Test Case 2

Опис тестового випадку		Перевірка авторизації з існуючим email		Статус		Успішно	
Кроки сценарію	1. Перейдіть до мобільної платформи			Тестові дані	1. email = bohdan.d@gmail.com		
	2. Відкрийте форму авторизації				2. password = bohdan&7		
	3. Заповніть всі необхідні поля						
	4. Натисніть кнопку “Продовжити”						
Очікуваний результат	Користувачу відображається домашня сторінка або форма введення персональних даних			Фактичний результат			Як і очікувалося

Сценарій авторизації за допомогою Google зображено на таблиці 3.3. Вона перевіряє коректність запиту доступу до потрібних даних та успішне проходження етапу авторизації до платформи.

Таблиця 3.3

Test Case 3

Опис тестового випадку		Перевірка авторизації за допомогою Google		Статус		Успішно	
Кроки сценарію	1. Перейдіть до мобільної платформи			Тестові дані	1. Існуючий обліковий запис Google		
	2. Відкрийте форму авторизації						
	3. Натисніть кнопку “Увійдіть за допомогою Google”						
	4. Оберіть свій обліковий запис						
Очікуваний результат	Користувачу відображається запит доступу до певних даних для авторизації, після чого відображається домашня сторінка			Фактичний результат		Як і очікувалося	

Для скидання паролю користувачу потрібно підтвердити свою електронну адресу за допомогою тимчасового коду, який відправляється на відповідну пошту. Після успішного введення коду відновлення, користувачу надається можливість змінити пароль до свого облікового запису. Опис сценарію описаний в таблиці 3.4

Таблиця 3.4

Test Case 4

Опис тестового випадку		Перевірка відновлення паролю	Статус	Успішно
Кроки сценарію	1. Перейдіть до мобільної платформи	Тестові дані	1. email = bohdan.d@gmail.com	
	2. Відкрийте форму відновлення паролю		2. code = 324545	
	3. Заповніть поле “email”		2. password = bohdan&7	
	4. Отримайте код для відновлення на вказану адресу		3. confirmPassword = bohdan&7	
	5. Введіть тимчасовий код до форми			
	6. Заповніть поля для збереження нового паролю			
Очікуваний результат	Користувач при скиданні паролю отримує повідомлення з тимчасовим кодом, після підтвердження його, успішно змінює пароль		Фактичний результат	Як і очікувалося

Додавання рецептів до планів супроводжується рядом змін, наприклад оновлення списку покупок, розрахування кількості інгредієнтів під вказану кількість порцій та відповідно відображення доданого рецепту у відповідному плані харчування. Етапи проходження цього функціоналу зображені на таблиці 3.5.

Таблиця 3.5

Test Case 5

Опис тестового випадку	Перевірка додавання рецепту до планів	Статус	Успішно
------------------------	---------------------------------------	--------	---------

Продовження таблиці 3.5

Кроки сценарію	1. Перейдіть до мобільної платформи	Тестові дані	1. email = bohdan.d@gmail.com	
	2. Авторизуйтеся в додатку		2. password = bohdan&7	
	3. Оберіть та натисніть на будь-який рецепт		3. Запланований день з трапезою	
	4. Натисніть кнопку “Додати до розкладу”			
	5. У відображеному вікні оберіть трапезу до якого потрібно додати рецепт			
	6. Перейдіть до сторінки “Мої плани”			
Очікуваний результат	Після успішного додавання рецепту до трапези, оновлюється план та список інгредієнтів.	Фактичний результат		Як і очікувалося

Для публікації рецептів потрібно перевірити правильність генерувань перекладів та успішне відображення запису іншим користувачам. Сценарій цього функціоналу зображений на таблиці 3.6.

Таблиця 3.6

Test Case 6

Опис тестового випадку		Перевірка публікації рецепту	Статус	Успішно
Кроки сценарію	1. Перейдіть до сторінки “Мої рецепти”	Предумови	1. Авторизований обліковий запис	
	2. Натисніть на неопублікований рецепт		2. Власний рецепт	
	3. Натисніть іконку “Налаштування”			
	4. Натисніть кнопку “Опублікувати”			
	5. Після успішної виконання операції перейдіть до сторінки “Пошук рецептів”			
	6. Знайдіть за назвою свій рецепт			
	7. Змініть мову додатку для перевірки локалізації свого рецепту			

Очікуваний результат	Після успішної публікації рецепт має локалізації та відображається іншим	Фактичний результат	Як і очікувалося
----------------------	--	---------------------	------------------

Висновки до третього розділу

В даному розділі була наведена детальна схема розгортання платформи. Кожен структурний елемент системи був описаний, а також було продемонстровано його взаємодію з іншими частинами платформи. Це дозволило зрозуміти архітектуру системи та взаємозв'язки між її компонентами.

Далі було розписано головні блоки інтерфейсу розробленого програмного продукту. Були розглянуті всі функціональні можливості додатку. Кожна сторінка клієнтського застосунку отримала свої унікальні функції та можливості в системі. Це допоможе кожному користувачеві зорієнтуватись у функціоналі додатку та використовувати його відповідно до своїх потреб.

Проведено тестування клієнтської та серверної частини додатку за допомогою описаних та виконаних Test Cases випадків. У результаті було остаточно перевірено правильність роботи мобільної-орієнтованої платформи.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи бакалавра було спроектовано та реалізовано мобільно-орієнтовану платформу для планування трапез, проведено аналіз предметної області та порівняно різні платформи, що працюють на основі зберігання рецептів та маніпуляцій над ними. На основі цього аналізу були визначені основні вимоги до платформи. В процесі дослідження була розроблена архітектура, підходи та основні технології реалізації мобільного додатку.

Проектування платформи включало аналіз варіантів використання програмного продукту та проектування структури, включаючи моделювання взаємодії між складовими частинами. Також було спроектовано базу даних для зберігання нової інформації від користувачів.

Описано алгоритм роботи з додатком та послідовні пункти варіантів його використання. Розглянуто основний функціонал, включаючи створення рецептів та їх публікація, планування трапез, генерування перекладів та порад штучним інтелектом, загальну взаємодію клієнта з платформою.

Також була надана детальна схема розгортання мобільно-орієнтованої платформи, описано взаємодію структурних елементів системи та наведено опис інтерфейсу розробленого програмного продукту для різних типів користувачів.

Додаток має інтуїтивний інтерфейс, який дозволяє користувачам легко відшуковувати потрібні рецепти, маніпулювати планами та їх переглядом, отримувати новий досвід з кожним днем використанням.

Результати роботи являють собою готовий до використання програмний комплекс, який допоможе поліпшити управління логістикою та планування харчових раціонів у військових підрозділах. У подальшому розвитку системи можливо впровадження засобів доступу без підключення до мережі. Це вдосконалення дозволить покращити функціональність та ефективність системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Дубинченко Б.М., Савіцький Р.С. ПЕРЕВАГИ ТА НЕДОЛІКИ РОЗРОБКИ МОБІЛЬНИХ ДОДАТКІВ. Тези XV міжнародної науково-практичної конференції "Комп'ютерні системи та мережні технології", 25-26 квітня 2024. Київ "НАУ", 2024. Подано до друку
2. Документація Node.JS [Електронний ресурс] – Режим доступу до ресурсу: <https://nodejs.org/docs/latest/api/>.
3. PostgreSQL [Електронний ресурс] – Режим доступу до ресурсу: <https://www.postgresql.org/>.
4. Документація React Native [Електронний ресурс] – Режим доступу до ресурсу: <https://reactnative.dev/docs/getting-started>.
5. Документація JavaScript [Електронний ресурс] – Режим доступу до ресурсу: <https://devdocs.io/javascript/>.
6. Best practices of designing REST API [Електронний ресурс] – Режим доступу до ресурсу: <https://jazzteam.org/technical-articles/restful-services-manual/>.
7. Документація Nest.JS [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.nestjs.com/>.
8. Документація Expo Router [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.expo.dev/router/introduction/>.
9. RealPlans [Електронний ресурс] – Режим доступу до ресурсу: <https://realplans.com/>.
10. Emeals [Електронний ресурс] – Режим доступу до ресурсу: <https://emeals.com/>.
11. Mealpy [Електронний ресурс] – Режим доступу до ресурсу: <https://play.google.com/store/apps/details?id=com.mk.mealpy>.

12. Визначення терміну “Архітектура” [Електронний ресурс] – Режим доступу до ресурсу: <https://www.geeksforgeeks.org/architecture-of-a-system/>.

13. Building Android Java/ JavaScript Bridges [Електронний ресурс] – Режим доступу до ресурсу: <https://labs.withsecure.com/publications/building-android-javajavascript-bridges>.

14. MVC Design Pattern [Електронний ресурс] – Режим доступу до ресурсу: <https://www.geeksforgeeks.org/mvc-design-pattern/>.

15. Single Page Application [Електронний ресурс] – Режим доступу до ресурсу: <https://www.geeksforgeeks.org/what-is-single-page-application/>.

16. JavaScript Dependency Injection [Електронний ресурс] – Режим доступу до ресурсу: <https://www.memcyco.com/home/what-is-javascript-dependency-injection/>.

17. TypeScript Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://www.typescriptlang.org/docs/>.

18. Документація Expo [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.expo.dev/>.

19. Документація pgAdmin [Електронний ресурс] – Режим доступу до ресурсу: <https://www.pgadmin.org/docs/>.

20. Документація Android Studio [Електронний ресурс] – Режим доступу до ресурсу: <https://developer.android.com/studio/intro>.

21. Документація PassportJS [Електронний ресурс] – Режим доступу до ресурсу: <https://www.passportjs.org/docs/>.

22. Документація Google Auth [Електронний ресурс] – Режим доступу до ресурсу: <https://developers.google.com/identity/account-linking>.

23. Документація Nodemailer [Електронний ресурс] – Режим доступу до ресурсу: <https://nodemailer.com/>.

24. Документація Pug [Електронний ресурс] – Режим доступу до ресурсу: <https://pugjs.org/api/getting-started.html>.
25. Документація Cloudinary [Електронний ресурс]. Режим доступу до ресурсу: https://cloudinary.com/documentation/node_integration.
26. Стаття про Generate knowledge prompting [Електронний ресурс] – Режим доступу до ресурсу: <https://arxiv.org/abs/2110.08387>.
27. Test Cases [Електронний ресурс] – Режим доступу до ресурсу: <https://luxequality.com/blog/how-to-write-test-cases-for-software-testing/>.
28. Документація i18n [Електронний ресурс] – Режим доступу до ресурсу: <https://react.i18next.com/>.
29. Стаття про Zero-Shot prompting [Електронний ресурс] – Режим доступу до ресурсу: <https://arxiv.org/abs/2110.08207>.
30. Документація OpenAI API [Електронний ресурс] – Режим доступу до ресурсу: <https://platform.openai.com/docs/introduction>.
31. Документація NativeWind [Електронний ресурс]. Режим доступу до ресурсу: <https://www.nativewind.dev/quick-starts/expo>.
32. React useContext [Електронний ресурс] - Режим доступу до ресурсу: <https://react.dev/reference/react/useContext>.
33. Документація Zustand [Електронний ресурс] - Режим доступу до ресурсу: <https://docs.pmnd.rs/zustand/getting-started/introduction>.
34. Документація React-hook-form [Електронний ресурс] - Режим доступу до ресурсу: <https://react-hook-form.com/docs>.
35. Entity Relationship (ER) [Електронний ресурс] – Режим доступу до ресурсу: <https://www.guru99.com/er-diagram-tutorial-dbms.html>.

ДОДАТКИ

					ПЗ.КР.Б – 121 – 24 – ПЗ	
						85

```

@Controller('plan')
export class PlanController {
  constructor(private readonly planService: PlanService) {}

  @UseGuards(JwtAuthGuard)
  @Get('all')
  async findAll(@Req() req: Request & { user: Omit<UserEntity, 'password'> })
  {
    const userId = req.user.id;
    return await this.planService.findAllByUserId(userId);
  }

  @UseGuards(JwtAuthGuard)
  @Get('getForCurrentYear')
  async findForTheYear(@Req() req: Request & { user: Omit<UserEntity,
'password'> }) {
    const userId = req.user.id;
    return await this.planService.findByCurrentYear(userId);
  }

  @UseGuards(JwtAuthGuard)
  @Get('getForCurrentWeek')
  async findForTheWeek(
    @Req() req: Request & { user: Omit<UserEntity, 'password'> },
    @Query() { lang }: Partial<LangParam>,
  ) {
    const userId = req.user.id;
    return await this.planService.findByCurrentWeek(userId, { lang:
getExistingLangCode(lang) });
  }

  @UseGuards(JwtAuthGuard)
  @Get('getByDate/:date')
  async findByDate(
    @Req() req: Request & { user: Omit<UserEntity, 'password'> },
    @Param('date') date: string,
    @Query() { lang }: Partial<LangParam>,
  ) {
    const userId = req.user.id;
    return await this.planService.findByDate(userId, date, { lang:
getExistingLangCode(lang) });
  }

  @UseGuards(JwtAuthGuard)
  @Get('generateAdvicesByPlanId/:id')
  async generateAdvicesByPlanId(
    @Req() req: Request & { user: Omit<UserEntity, 'password'> },
    @Param('id') id: string,
    @Query() { lang }: Partial<LangParam>,
  ) {
    const userId = req.user.id;

    return await this.planService.generateAdvicesByPlanId(userId, +id, {
      lang: getExistingLangCode(lang),
    });
  }

  @UseGuards(JwtAuthGuard)
  @Post()

```

```

    async create(
        @Req() req: Request & { user: Omit<UserEntity, 'password'> },
        @Body() createPlanDto: CreatePlanInput,
    ) {
        const userId = req.user.id;
        return this.planService.createPlan(userId, createPlanDto);
    }

    @UseGuards(JwtAuthGuard)
    @Post('createWithMealPlans')
    async createWithMealPlans(
        @Req() req: Request & { user: Omit<UserEntity, 'password'> },
        @Body() createPlanWithMealPlans: CreatePlanWithMealPlans,
    ) {
        const userId = req.user.id;
        return this.planService.createPlanWithMealPlans(userId,
createPlanWithMealPlans);
    }

    @UseGuards(JwtAuthGuard)
    @Delete(':id')
    remove(@Req() req: Request & { user: Omit<UserEntity, 'password'> },
    @Param('id') id: string) {
        const userId = req.user.id;

        return this.planService.deletePlanById(+id, userId);
    }

    @UseGuards(JwtAuthGuard)
    @Patch(':id')
    async update(
        @Req() req: Request & { user: Omit<UserEntity, 'password'> },
        @Param('id') id: string,
        @Body() updatePlanDto: UpdatePlanWithMealPlans,
    ) {
        const userId = req.user.id;
        return this.planService.updatePlanWithMealPlans(userId, +id,
updatePlanDto);
    }
}

```

```

@Injectable()
export class RecipeService {
  constructor(
    private readonly recipeMapper: RecipeMapper,
    private readonly recipeRepository: RecipeRepository,
    private readonly categoryService: CategoryService,
    private readonly userService: UsersService,
    private readonly ratingService: RatingService,
    private readonly openAiService: OpenAiService,
    private readonly ingredientService: IngredientService,
    @InjectRepository(RecipeEntity)
    private readonly recipeRepositoryEntity: Repository<RecipeEntity>,
    @InjectRepository(RecipeTranslationEntity)
    private readonly recipeTranslationRepository:
Repository<RecipeTranslationEntity>,
  ) {}

  async getMyRecipes(
    userId: number,
    { lang, searchQuery }: Partial<LangParam & SearchParam>,
  ): Promise<ClientPreviewRecipeDTO[]> {
    const myRecipes = await this.recipeRepository.getMyRecipes({ userId,
lang, searchQuery });

    return this.recipeMapper.entityToClientPreviewRecipesDTO(myRecipes);
  }

  async getPreviewRecipesByCategoryId(
    categoryId: string,
    parameters: Partial<PaginationParams & LangParam>,
  ): Promise<ClientPreviewRecipeDTO[]> {
    const previewRecipesByCategoryId = await
this.recipeRepository.previewRecipesByCategoryId({
    categoryId,
    ...parameters,
  });

    return
this.recipeMapper.entityToClientPreviewRecipesDTO(previewRecipesByCategoryId)
;
  }

  async getRecipeById(recipeId: number, { lang }: LangParam):
Promise<ClientRecipeDTO> {
    const recipeById = await this.recipeRepository.getRecipeById({ recipeId,
lang });

    return this.recipeMapper.entityToClientRecipeDTO(recipeById);
  }

  async getFilteredPreviewRecipes(
    parameters: Partial<PaginationParams & LangParam & SearchParam &
FilterRecipeParams>,
  ): Promise<ClientPreviewRecipeDTO[]> {
    const filteredPreviewRecipes =
    await this.recipeRepository.getFilteredPreviewRecipes(parameters);
  }

```

```

        return
        this.recipeMapper.entityToClientPreviewRecipesDTO(filteredPreviewRecipes);
    }

    async publishRecipe(userId: number, recipeId: number, publishDto:
PublishRecipeDto) {
        const recipe = await this.recipeRepository.findOneByIdAndUserId({
recipeId, userId });

        if (!recipe) {
            throw new NotFoundException('Recipe not found');
        }

        recipe.isPublished = publishDto.isPublished;

        if (publishDto.isPublished) {
            await this.translateRecipe(recipeId, LangEnum.EN);
        }

        return await this.recipeRepositoryEntity.save(recipe);
    }

    async translateRecipe(recipeId: number, translateTo: LangEnum) {
        const recipe = await this.recipeRepository.getRecipeById({ recipeId,
lang: LangEnum.UK });

        if (!recipe) {
            throw new NotFoundException('Recipe not found');
        }

        const recipeContext = recipe.translations[0];
        const recipeTranslation = this.recipeTranslationRepository.create({
            languageCode: translateTo,
        });

        recipeTranslation.title = await
this.openAiService.translateTextFromUaToEn(recipeContext.title);

        recipeTranslation.cookingInstructions = await
this.openAiService.translateTextFromUaToEn(
            recipeContext.cookingInstructions,
            2000,
        );

        if (!!recipeContext.description) {
            recipeTranslation.description = await
this.openAiService.translateTextFromUaToEn(
                recipeContext.description,
            );
        }
        recipeTranslation.recipe = recipe;

        await this.recipeTranslationRepository.save(recipeTranslation);

        await
this.ingredientService.translateIngredients(recipe.ingredientToRecipes,
translateTo);
    }

```

```

    async addFavorite(userId: number, recipeId: number): Promise<void> {
        const recipe = await this.recipeRepository.getRecipeWithUserById({
            recipeId });

        if (!recipe) {
            throw new NotFoundException('Recipe not found');
        }

        if (recipe.user.id === userId) {
            throw new Error('You can not add your own recipe to favorites');
        }

        await this.recipeRepository.saveFavoriteRecipeByUserId({ userId, recipe
    });
    }

    async removeFavorite(userId: number, recipeId: number): Promise<void> {
        await this.recipeRepository.deleteFavoriteRecipe({ userId, recipeId });
    }

    async getFavoriteRecipes(
        userId: number,
        parameters: Partial<LangParam & PaginationParams>,
    ): Promise<ClientPreviewRecipeDTO[]> {
        const favoriteRecipes = await
            this.recipeRepository.getFavoritePreviewRecipesByUserId({
                userId,
                ...parameters,
            });

        return
            this.recipeMapper.entityToClientPreviewRecipesDTO(favoriteRecipes);
    }

    async isFavoriteRecipe(userId: number, recipeId: number): Promise<boolean>
    {
        return this.recipeRepository.isFavoriteRecipe(userId, recipeId);
    }

    async createRecipe({
        userId,
        lang,
        ingredients,
        cookingInstructions,
        title,
        description,
        categoryIds,
        ...recipeData
    }): CreateRecipeDto & LangParam & { imageUrl: string; userId: number }):
    Promise<RecipeEntity> {
        const user = await this.userService.findUserById(userId);

        const newRecipe = await this.recipeRepositoryEntity.save({
            user,
            ...recipeData,
        });

        await this.recipeTranslationRepository.save({
            title,
            description,
            cookingInstructions,

```

```

        languageCode: lang,
        recipe: newRecipe,
    });

    await this.ratingService.createRatingWithDefaults(newRecipe);

    await Promise.all(
        ingredients.map(
            async ingredient =>
                await this.ingredientService.createIngredientToRecipe({
                    ...ingredient,
                    recipe: newRecipe,
                    lang,
                })
        ),
    );

    if (categoryIds.length > 0) {
        await Promise.all(
            categoryIds.map(async categoryId => {
                await this.categoryService.createCategoryToRecipe(categoryId,
                    newRecipe.id);
            })
        );
    }

    return newRecipe;
}

```