**ПОЛТАВСЬКИЙ ДЕРЖАВНИЙ АГРАРНИЙ УНІВЕРСИТЕТ**
**НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЕКОНОМІКИ, УПРАВЛІННЯ, ПРАВА ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**
**КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ**

# Пояснювальна записка

до кваліфікаційної роботи на здобуття ступеня вищої освіти бакалавр

на тему: «Проєктування мобільного додатку для вибору продуктів харчування»

Виконав: здобувач вищої освіти
за освітньо-професійною програмою
Інформаційні управляючі системи
спеціальності 126 Інформаційні
системи та технології
освітнього ступеня бакалавр
групи 126ІСТ_бд_2022[1](стн)
Бзот М.О.
Керівник: Протас Н.М.
Рецензент: Муравльов В.В.

**Полтава – 2024 року**

# ВСТУП

*Актуальність.* На сьогодні існує багато видів обмежень у харчуванні. Причинами таких обмежень можуть бути, зокрема, релігійні переконання, етичні міркування чи медичні протипоказання. До обмежень раціону з етичних причин належать такі режими харчування, як вегетаріанство та веганство. За оцінками BusinesStat, продажі продуктів для веганів до 2018 року зростали темпами 18,6 – 32,2% на рік: у 2014-2018 рр. пропозиція товарів для веганів на світовому ринку збільшилася у 2,5 рази [1]. У період 2019-2023 років пропозиція товарів для веганів на світовому ринку продовжить зростати на 27,2-30,5% на рік [2]. Це свідчить про зростання попиту на продукти рослинного походження. За різними даними, до 4% людей є вегетаріанцями та близько 2,2% – веганами [3].

Крім етичних міркувань, причиною обмеження раціону може бути віросповідання. Найбільш поширені релігійні обмеження – пост (у православних християн), халяль (у мусульман) та кашрут (у юдеїв) [4]. За даними опитування, близько 65-68% опитаних сповідують православ'я, близько 7% – іслам, і близько 1% – іудаїзм [5]. При цьому, близько 25% опитаних дотримуються православних постів, і близько 50% з них відмовляються від продуктів тваринного походження на час посту [6].

Іншою поширеною причиною обмеження раціону є різні непереносимості. Наприклад, 16% населення мають непереносимість лактози [7]. Крім лактози, зустрічаються непереносимості глютену, кофеїну, саліцилатів, амінів та інших компонентів. Проте найсерйознішою причиною обмеження раціону є харчова алергія. Вживання продуктів, що містять алерген, може спричинити серйозні наслідки для алергіка. Найчастіше алергенами виступають горіхи, морепродукти, яйця, молоко, бобові, злаки, цитрусові та мед. За даними ВООЗ, за останні десять років захворюваність на алергію зросла на 20% [8].

Отже, існує велика кількість людей, які з тих чи інших причин відмовляються від вживання певних продуктів (періодично чи постійно). Однак сучасний ритм життя не завжди дозволяє витрачати час на ретельний аналіз

складу продуктів. У поспіху можна недостатньо уважно прочитати склад продукту. Ця робота спрямована на створення мобільного додатка, що забезпечує персональний підбір продуктів харчування залежно від обмежень у раціоні. Програма пропонує користувачеві персональний список продуктів, розділених на категорії. Відмінною особливістю цієї програми є функція винесення вердикту за штрих-кодом продукту: користувач сканує штрих-код продукту, а система надає короткий вердикт щодо його придатності залежно від обмежень користувача.

*Метою кваліфікаційної роботи* є розробка мобільного додатка на платформі Android, призначеного для персонального вибору продуктів харчування.

Відповідно до мети роботи необхідно вирішити наступні завдання:

1) провести аналіз предметної галузі;

2) виконати огляд існуючих рішень на ринку мобільних програм;

3) визначити вимоги до системи, що розробляється, і розробити варіанти її використання;

4) розробити архітектуру мобільного додатку;

5) розробити архітектуру бази даних для зберігання відомостей про продукти;

6) спроєктувати інтерфейс мобільного додатку на платформі Android;

7) спроєктувати та розробити REST-сервіс для віддаленого доступу до бази даних;

8) розробити план тестування системи.

*Об'єкт дослідження* – процеси проєктування мобільних додатків з клієнт-сервісною архітектурою.

*Предмет дослідження* – проєктування та розробка мобільного додатку з використанням REST-сервісу для віддаленого доступу до бази даних.

*Методи досліджень* – проведені в роботі дослідження базуються методах програмної інженерії, розробки мобільних додатків з клієнт-серверною архітектурою, проєктування та нормалізації баз даних.

*Інформаційна база* – Інтернет-ресурси та друковані видання, що місять інформацію про проєктування мобільних додатків, стандарти з розробки програмного забезпечення, довідкова інформація про використані програмні компоненти архітектури.

*Практична значущість* – на основі запропонованих архітектури, схеми бази даних, моделей і патернів та інтерфейсу, а також планів тестування спрощено реалізацію мобільного додатка, що виконує корисні функції вибору продуктів харчування.

Результати роботи апробовані в рамках наукової конференції здобувачів вищої освіти за результатами науково-дослідної роботи у 2023-2024 роках.

Структура кваліфікаційної роботи логічно пов’язана з задачами досліджень і містить перелік умовних позначень, вступ, три розділи основної частини, висновки, список використаних джерел. Загальний обсяг текстової частини кваліфікаційної роботи складає 42 сторінки формату А4. Вона містить 18 рисунків і 3 таблиці.

# РОЗДІЛ 1
# АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ, ЗАДАЧ, СТРУКТУРИ ТА ВИМОГ ДО МОБІЛЬНИХ ДОДАТКІВ

## 1.1 Аналіз стандартів та систем позначення товарів

На сьогоднішній день більшість вироблених товарів у світі реєструється в міжнародній системі товарної нумерації та штрихового кодування GS1 (колишній EAN/UCC) [9]. Система GS1 керується міжнародною асоціацією GS1. GS1 – це міжнародна організація, що займається стандартизацією обліку та штрихового кодування логістичних одиниць.

GTIN – це ідентифікаційний номер GS1 для одиничного або групового пакування товару. Він наноситься у вигляді штрих-коду на упаковку товару у формі коду EAN-13 або EAN-8 і є унікальним ідентифікатором товару [10].

На рис.1.1 показані приклади штрих-кодів у форматах EAN-13 (зверху) та EAN-8 (знизу). Зображення взято з офіційного сайту GS1.

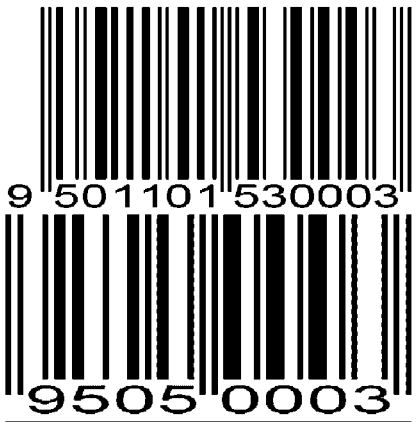


Рисунок 1.1 – Приклади штрих-кодів EAN-13 та EAN-8

Відповідно до стандартів GS1, зміна складу товару, а також поява відмінностей, які будуть видно споживачеві, вимагають присвоєння нового GTIN (якщо виконується обидва умови) [11]. Наприклад, додавання нового

інгредієнта, що є алергеном, до складу продукту є обов'язковим для декларування, а це в свою чергу, вимагає присвоєння нового GTIN.

Окрім штрих-кодів EAN, для маркування товарів також широко використовуються QR-коди (Quick Response Code). QR-коди – це двовимірні штрих-коди, які можуть містити значно більше інформації, ніж EAN-коди. Їх можна сканувати за допомогою смартфонів або спеціальних сканерів, що робить їх зручним інструментом для надання споживачам додаткової інформації про продукт. QR-коди на упаковці товару можуть містити:

– детальну інформацію про продукт: опис, склад, інструкції з використання, термін придатності тощо;

– посилання на веб-сайти: сторінки виробника, офіційні ресурси з інформацією про продукт, маркетингові кампанії тощо;

– інтерактивний контент: відео, аудіо, 3d-моделі, опитування тощо;

– соціальні мережі: посилання на сторінки продукту або бренду в соціальних мережах.

QR-коди роблять упаковку товару більш інтерактивною та інформативною, що може підвищити зацікавленість споживачів продуктом та покращити його сприйняття [12].

Окрім QR-кодів та EAN, існують й інші типи кодування, які використовуються для маркування товарів. Деякі з них:

1. Data Matrix: двовимірний код, схожий на QR-код, але з більшою щільністю даних [13].

2. UPC: одно- або двовимірний код, що використовується в США та Канаді [14].

3. Code 128: одномірний код, який може містити алфанумерні символи [15].

4. GS1 DataMatrix: двовимірний код, розроблений GS1 для маркування логістичних одиниць [13].

Вибір типу кодування для маркування товару залежить від декількох факторів, таких як кількість інформації, яку потрібно закодувати; тип інформації, яку потрібно закодувати; можливостей сканування; вартості кодування.

Переваги використання QR-кодів та інших видів кодування:

– надання споживачам додаткової інформації про продукт;
– збільшення зацікавленості споживачів продуктом;
– покращення сприйняття продукту споживачами;
– зменшення кількості фальсифікату;
– автоматизація процесів відстеження та логістики.

QR-коди та інші види кодування є цінними інструментами для маркування товарів, які можуть допомогти виробникам покращити комунікацію зі споживачами, підвищити прозорість ланцюжка постачання та стимулювати продажі.

## 1.2 Аналіз наявних на ринку додатків для розпізнавання штрих-кодів

На сучасному ринку мобільних додатків для розпізнавання штрих-кодів представлено багато варіантів, кожен з яких має свої особливості, переваги та недоліки. Вони дозволяють користувачам швидко та зручно отримувати інформацію про продукти, що особливо актуально у контексті зростання попиту на здоровий спосіб життя та контроль за споживаними товарами.Firebase – це кросплатформний SDK, розроблений компанією Google [10]. Ця платформа надає розробникам широкий вибір інструментів для створення додатків для Android. Одним з модулів Firebase є ML Kit – мобільне SDK, яке пропонує можливості машинного навчання [11]. ML Kit містить готові до використання API, включаючи API для сканування та розпізнавання штрих-кодів (включаючи формати EAN-8 та EAN-13).

Перевагою використання Firebase ML Kit для розпізнавання штрих-кодів є його підтримка Google, що може забезпечити стабільність та оновлення [12]. Проте варто зауважити, що ML Kit може бути більш важким рішенням порівняно

з іншими інструментами, наприклад, з ZXing. Додатково можна відзначити, що Firebase ML Kit також пропонує інші можливості машинного навчання, такі як виявлення облич, визначення тексту та інші.

ZXing – це відкрита бібліотека з вихідним кодом, яка надає можливість розпізнавати штрих-коди на платформі Android [18]. Бібліотека реалізована мовою Java з можливістю портування на інші мови програмування. Вона підтримує різноманітні формати, включаючи EAN-8 та EAN-13, які використовуються для маркування продуктів харчування. Проект ZXing надає розробникам Android можливість інтегрувати функцію сканування штрих-кодів у свої програми за допомогою стороннього додатка «Barcode Scanner», який є продуктом ZXing. Також існує можливість розпізнавання штрих-кодів без використання стороннього додатка, використовуючи класи, що надаються у комплекті з бібліотекою. Наразі проект знаходиться у режимі технічного обслуговування. Розробники не планують активно розвивати проект, і програма «Barcode Scanner» не буде отримувати оновлення.

Для розпізнавання QR-кодів найбільш популярними додатками є Google Lens та QR & Barcode Scanner. Google Lens це безкоштовний додаток від Google, який може сканувати QR-коди, а також штрих-коди, візитки, текст, книги та інші об'єкти. Додаток пропонує переклад, пошук інформації, можливість ділитися знайденою інформацією та інші функції.

Рисунок 1.2 – Google Lens app icon

QR & Barcode Scanner це безкоштовний додаток, який може сканувати QR-коди та штрих-коди, а також пропонує історію сканування, можливість створювати власні QR-коди та інші функції.

Інші додатки, такі як MyFitnessPal та Yuka, також здобули популярність серед користувачів. MyFitnessPal орієнтований на тих, хто хоче відстежувати калорійність та поживні речовини споживаних продуктів. Він пропонує велику базу даних продуктів і можливість інтеграції з фітнес-трекерами. Однак, користувачі часто скаржаться на необхідність підписки для доступу до всіх функцій додатка [16].

Yuka, своєю чергою, фокусується на оцінюванні корисності продуктів для здоров'я. Додаток надає оцінку продуктів на основі їх складу, що допомагає користувачам обирати більш здорові варіанти. Основною перевагою Yuka є її зрозумілий та зручний інтерфейс, однак база даних не велика, що може обмежувати користувачів у деяких випадках.

Також варто згадати додаток ShopSavvy, який спеціалізується на порівнянні цін на продукти в різних магазинах. Він дозволяє користувачам знайти найвигідніші пропозиції, але не надає детальної інформації про харчову цінність чи склад продуктів, що робить його менш корисним для тих, хто слідкує за своїм харчуванням [17].

На сьогоднішній день на ринку мобільних додатків для операційної системи Android не існує рішень, які повністю відтворюють функціонал системи, що розробляється. Нижче наведено аналіз додатків, які є частковим аналогом розроблюваної системи. Розглядалися програми, що мають рейтинг не нижче 2 зірок у Google Play [18].

Один із таких додатків – Food Scanner. Це мобільний додаток, який за допомогою штрих-коду товару визначає його категорію (халяль, веган, без глютену, кошерна їжа) [12]. У Google Play цей додаток отримав 111 відгуків і має рейтинг 4.1 зірки. Food Scanner дає користувачеві можливість вказати, які категорії товарів він віддає перевагу, а також які інгредієнти він уникає.

Додаток пропонує такі функції користувачеві:

– сканувати штрих-код товару для визначення придатності продукту для користувача;

– додавати нові продукти до бази даних;

– виправляти товари у базі;

– шукати продукти за назвою;

– детально переглядати інформацію про продукт у базі;

– переглядати історію останніх сканувань.

На рис.1.3 наведені знімки екрану роботи цього додатка.

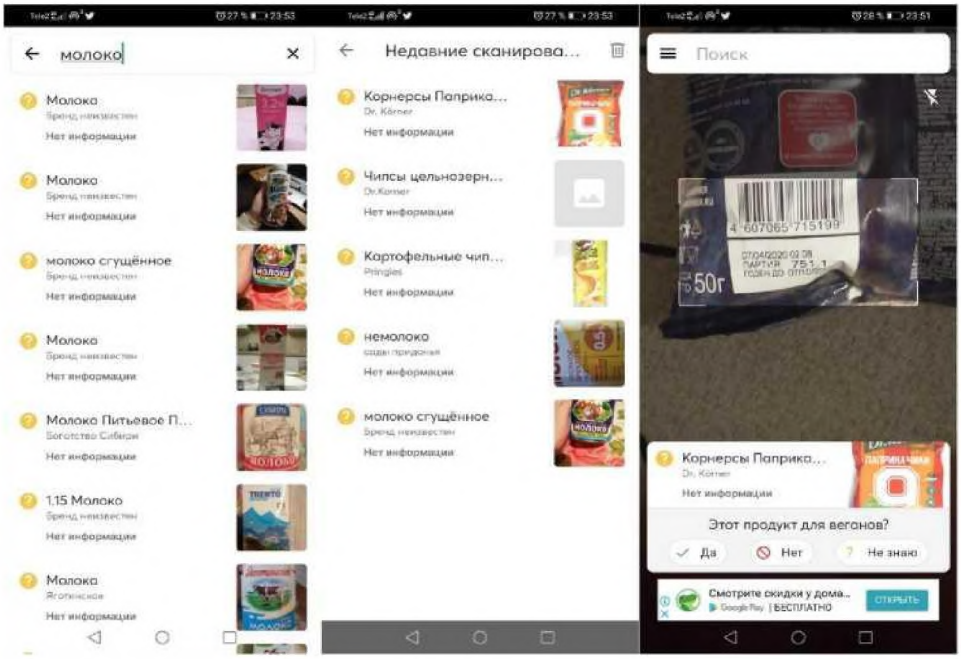


Рисунок 1.3 – Скріншоти екрану додатку Food Scanner

До переваг додатку Food Scanner можна віднести можливість пошуку товарів за базою та можливість вибору бажаної категорії.

Недоліки додатку Food Scanner наступні:

1. Користувачі самі вносять товари в основу без подальшої модерації. Це може призвести до зловмисного внесення неправдивої інформації.

2. Погана робота сканера.

3. Користувачі можуть редагувати наявні записи. Це може призвести до псування даних у базі.

VegCode – це мобільний додаток, призначений для веганів, який визначає за штрих-кодом товару, чи він є етичним [16]. У Google Play цей додаток має 52 оцінки та рейтинг 4.2 зірки. У VegCode немає можливості вручну встановлювати

обмеження у раціоні, він призначений виключно для автоматичного сканування штрих-кодів товарів. Крім того, в додатку відсутня функція перегляду товарів без попереднього сканування штрих-коду.

Цей додаток надає можливість користувачам сканувати штрих-коди товарів для перевірки наявності продуктів тваринного походження, додавати нові товари до бази даних та вносити пропозиції щодо виправлення інформації про вже наявні товари.

На рис. 1.4 наведені знімки екрану роботи додатка VegCode.

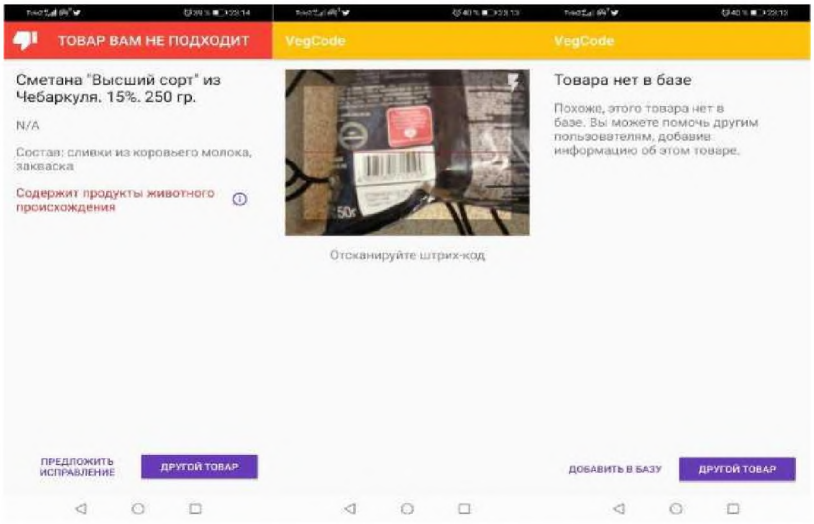


Рисунок 1.4 – Скріншоти екрану додатку VegCode

Щодо переваг, додаток відзначається гарною роботою сканера та підтримкою від розробників, але його основа продуктів є досить обмеженою. Серед недоліків варто відзначити нездатність сканера розпізнавати штрих-коди у форматі EAN-8.

Одним з найпопулярніших додатків у цій категорії є OpenFoodFacts. Він пропонує користувачам можливість сканування штрих-кодів для отримання детальної інформації про склад, харчову цінність та наявність добавок у продуктах. Цей додаток відрізняється великою базою даних, яка налічує тисячі продуктів, і постійно оновлюється завдяки внескам користувачів. OpenFoodFacts – це мобільний додаток, який дозволяє користувачам сканувати продукти харчування або їх штрих-коди і отримувати інформацію про склад, харчову

цінність і добавки, що містяться в продуктах. У Google Play цей додаток має 2178 оцінок і рейтинг 3,8 зірки, будучи найпопулярнішим серед розглянутих. Додаток OpenFoodFacts пропонує можливість вказати алергени, щоб попереджати користувача про їхню присутність у продуктах. Він надає такі функції, як сканування штрих-коду товару для визначення його придатності, додавання нових продуктів до бази даних, виправлення товарів у базі, пошук продуктів за назвою, детальний перегляд інформації про продукт, перегляд останніх сканувань, порівняння продуктів між собою, створення списків продуктів та перегляд списку категорій. На рис. 1.5 представлені скріншоти роботи цієї програми.

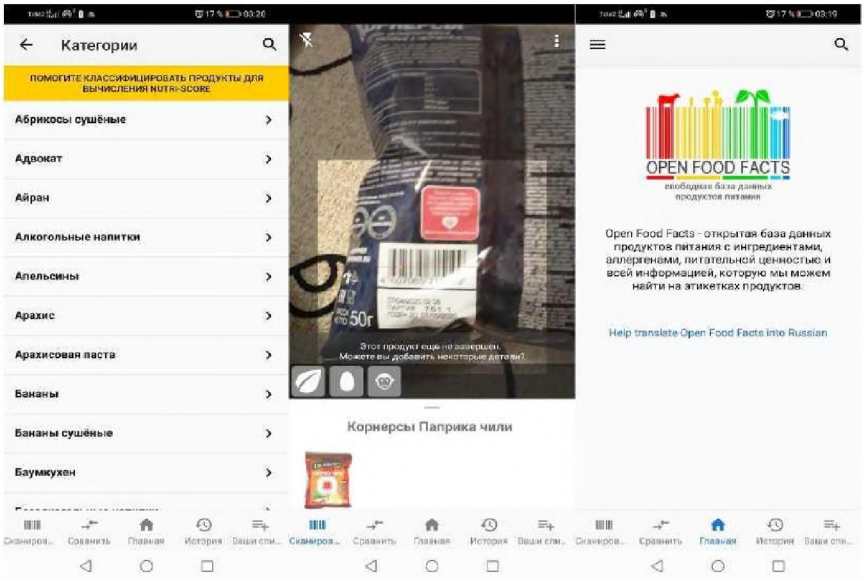


Рисунок 1.5 – Скріншоти екрану додатку OpenFoodFacts

Серед переваг додатку можна відзначити велику базу продуктів харчування, можливість вказати алергени, переглядати список категорій та створювати власні списки.

До недоліків можна віднести нестабільну роботу сканера, який інколи сканує один штрих-код кілька разів поспіль. Крім того, користувачі самостійно додають товари до бази даних без подальшої модерації, що може призвести до внесення неправдивої інформації. Також можливість редагування наявних записів користувачами може спричинити псування даних у базі.

## 1.3 Аналіз вимог до мобільного додатку для вибору продуктів харчування

Проєктована система буде мобільним додатком і веб-сервісом, що надає доступ до бази даних. Користувач зможе користуватися програмою без постійного підключення до Інтернету. Завдяки мобільному додатку користувачі зможуть переглядати розділений на категорії список продуктів, які відповідають їхнім заданим обмеженням; додавати цікаві продукти до обраного; сканувати штрих-коди продуктів за допомогою камери телефону, щоб отримати висновок про можливість використання цього продукту. Якщо інформація про продукт відсутня в базі даних, користувач зможе надіслати пропозицію щодо його додавання. Модератори системи матимуть можливість додавати нові продукти до бази даних.

Для користування програмою користувачам потрібен пристрій на операційній системі Android версії 5.0 або вище. Для завантаження та оновлення бази продуктів необхідне підключення до Інтернету. Для використання функції сканування штрих-кодів пристрій повинен бути оснащений камерою.

Проаналізуємо функціональні вимоги до системи через варіанти використання. На рис.1.6 представлена діаграма варіантів використання системи вибору продуктів харчування.

З системою вибору продуктів харчування взаємодіють три актори:

1. Модератор – користувач системи, якому доступна можливість додавання записів до бази даних.

2. Авторизований користувач – користувач системи, якому доступний весь функціонал програми.

3. Неавторизований користувач – користувач системи, якому доступні функції авторизації та реєстрації.

Модератор може здійснювати такі дії:

– переглядати запити на додавання продуктів до бази даних;

– додавати відомості про нові продукти до бази.

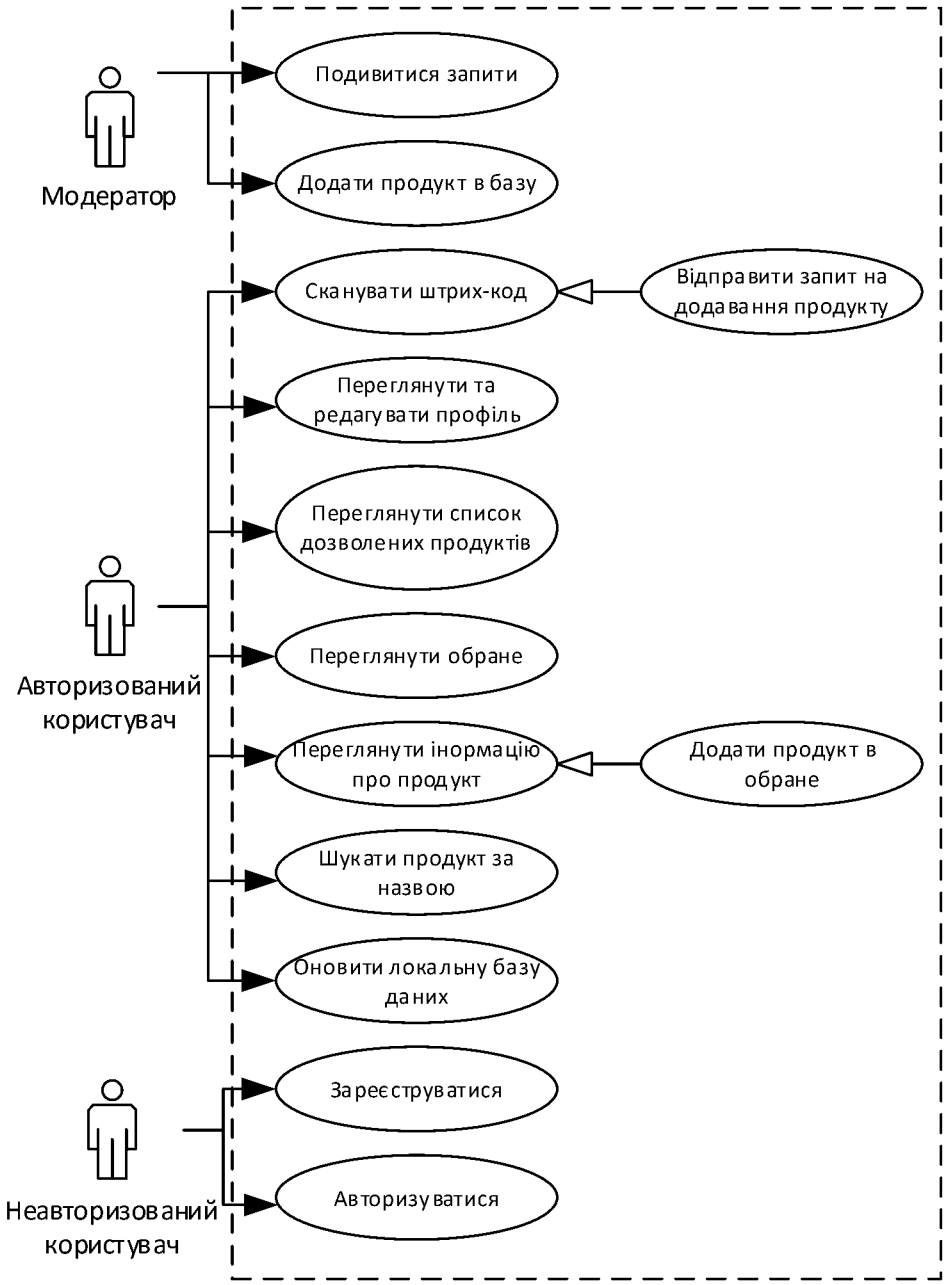


Рисунок 1.6 – Діаграма варіантів використання

Авторизований користувач може виконувати такі дії:

– змінювати обмеження в раціоні харчування в особистому кабінеті користувача;

– переглядати категоризований список дозволених продуктів харчування;

– оновлювати локальну базу даних (додаючи нові продукти з серверної бази до каталогу на пристрої);

– шукати продукт за назвою;

– переглядати детальну інформацію про знайдений або відсканований продукт, а також про продукт зі списку;

– додавати продукт у обране;

– сканувати штрих-коди за допомогою камери мобільного телефону;

– надсилати модератору запит на додавання продукту до бази за відсутності цього товару;

– виходити з системи.

Неавторизований користувач може виконувати такі дії:

– авторизуватися в системі за допомогою логіна та пароля;

– зареєструватися в системі за допомогою електронної пошти.

До нефункціональних вимог слід віднести наступні:

1. Система повинна забезпечувати можливість використання без підключення до Інтернету.

2. Мобільний додаток має бути розроблений мовою Java з використанням Android SDK [20].

3. Веб-сервіс має бути розроблений мовою Python з використанням фреймворку Django [21].

4. Система повинна використовувати СУБД MySQL [22].

5. Система повинна бути доступна на пристроях з операційною системою Android версії 5.0 і вище [23].

Мобільний додаток для вибору продуктів харчування дозволить користувачам переглядати категоризовані списки продуктів, шукати їх за назвою, сканувати штрих-коди, додавати продукти до обраного та надсилати запити на додавання нових продуктів. Додаток буде доступний без підключення до Інтернету та розроблений на Android 5.0 і вище. Цей інструмент допоможе людям обирати їжу для здорового харчування та контролювати свій раціон.

# РОЗДІЛ 2
# ПРОЄКТУВАННЯ АРХІТЕКТУРИ МОБІЛЬНОГО ДОДАТКА ТА БАЗИ ДАНИХ, ЯКУ ВІН ВИКОРИСТОВУЄ

## 2.1 Проєктування архітектури мобільного додатка

Архітектура мобільного додатка побудована на основі схеми MVC (Model/View/Controller) [24]. MVC складається з трьох основних компонентів. Модель представляє дані та бізнес-логіку програми, вигляд відповідає за екранне відображення даних, а контролер визначає, як інтерфейс реагує на взаємодію з користувачем. На рис. 2.1 представлена модель взаємодії між компонентами у схемі MVC.

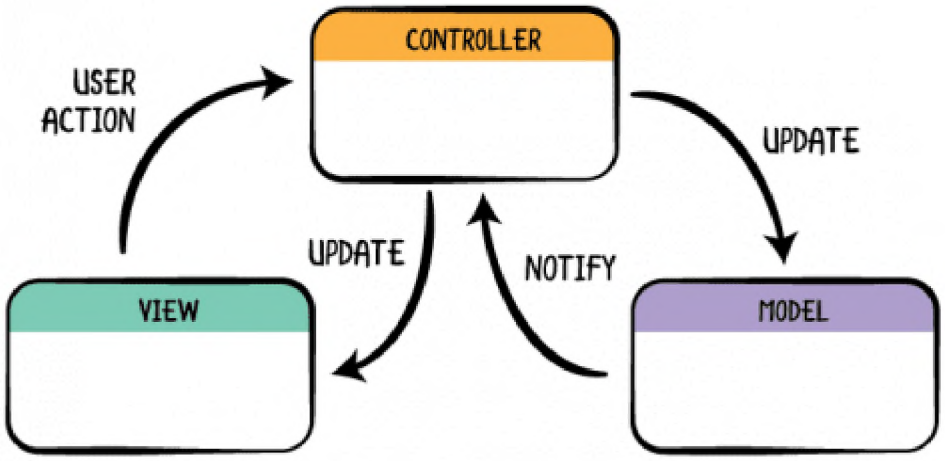


Рисунок 2.1 – Архітектура MVC

Система вибору продуктів харчування складається з двох компонентів: мобільного додатка та сервера. Для розробки діаграми компонентів необхідно визначити, як здійснюється взаємодія клієнта (мобільного додатка) з сервером.

Архітектура системи базується на клієнт-серверній моделі, що забезпечує ефективну взаємодію між мобільним додатком і сервером. Клієнтська частина (мобільний додаток) буде реалізована за допомогою Java та Android SDK, забезпечує зручний інтерфейс для користувачів. Клієнтська частина

дотримується принципів MVC, де модель відповідає за обробку даних, вигляд за відображення, а контролер за взаємодію з користувачем.

Локальна база даних використовується для збереження даних на пристрої, що дозволяє додатку працювати в автономному режимі без постійного підключення до Інтернету.

Серверна частина реалізована на Python з використанням фреймворку Django, забезпечує обробку запитів від мобільного додатка і управління даними.

СУБД MySQL використовується для зберігання основної бази даних продуктів та інформації про користувачів.

Веб-сервіс надає RESTful API, яке дозволяє мобільному додатку здійснювати запити до сервера для отримання та оновлення даних [25].

Ця архітектура забезпечує гнучкість, масштабованість і надійність системи вибору продуктів харчування, дозволяючи їй ефективно обробляти запити користувачів і працювати в режимі офлайн.

## 2.2 Розробка послідовності взаємодії між клієнтом та сервером

Для взаємодії між клієнтом та сервером було обрано стандарт REST. Використання REST обумовлено популярністю цього рішення, а також наявністю досвіду роботи з цим підходом. Передача стану подання (REST) – це підхід до розробки веб-служби, який дозволяє користувачеві отримувати та оновлювати інформацію про ресурси. Він став домінуючим при розробці API для використання в Інтернеті [26].

REST слідує чотирьом базовим принципам проектування [27]:

1) явне використання HTTP -методів;

2) незбереження стану;

3) надання URI, аналогічних структурі каталогів;

4) передача даних у XML, JavaScript Object Notation (JSON) або обох форматах.

У системі, що розробляється, передача даних буде здійснюватися у форматі JSON [28].

На рис. 2.2 представлена діаграма компонентів системи вибору продуктів харчування.

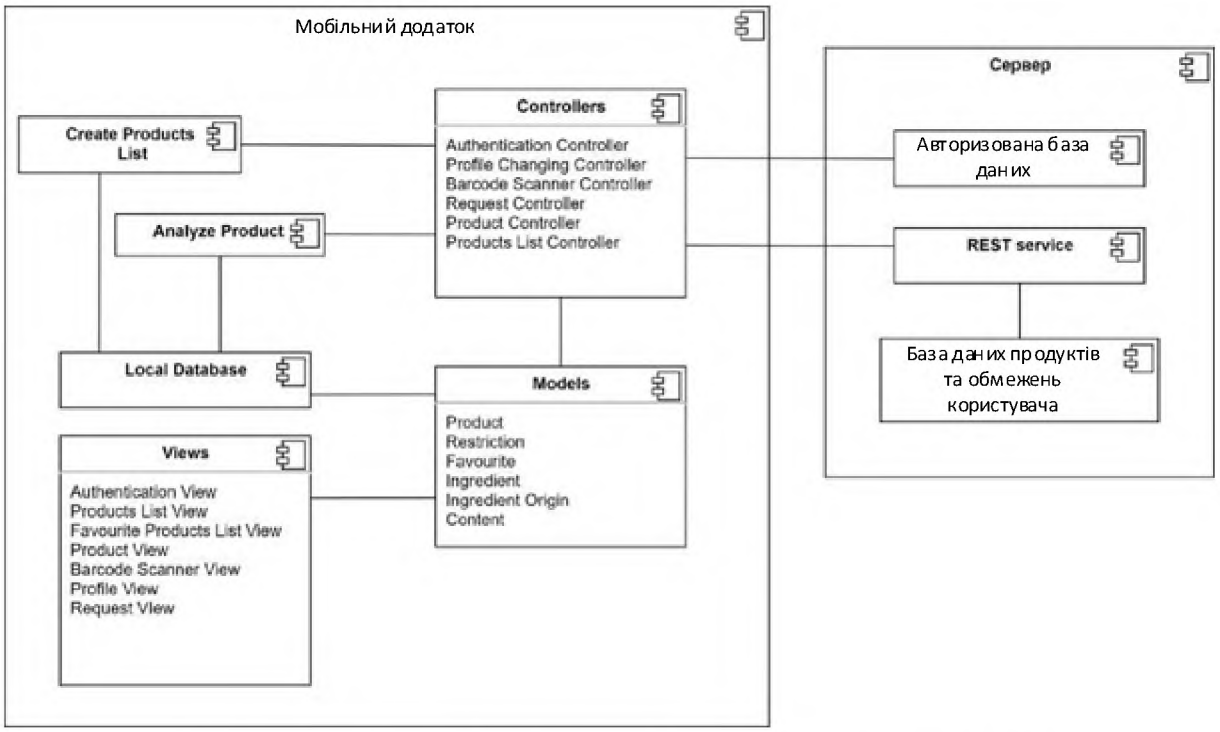


Рисунок 2.2 – Діаграма взаємодії компонентів системи

Мобільний додаток побудований на основі схеми MVC. Крім елементів даної схеми, додаток містить локальну базу даних, функцію створення списку продуктів та функцію аналізу складу продукту. Розглянемо докладніше елементи програми.

Було виділено 6 основних моделей (models) програми, які частково повторюють сутність бази даних продуктів та обмежень користувачів.

1. Product – Модель для зберігання та обробки інформації про продукти.

2. Restriction – Модель для зберігання та обробки інформації про обмеження користувача.

3. Favourite – Модель для зберігання та обробки інформації про обрані продукти користувача.

4. Ingredient – модель для зберігання та обробки інформації про інгредієнти.

5. Ingredient Origin – модель для зберігання та обробки інформації про походження (рослинна, тварина тощо) інгредієнтів.

6. Content – Модель для зберігання та обробки інформації про склад продуктів.

Було виділено 7 подань (views) для програми.

1. Authentication View – екран програми, що відповідає за автентифікацію та реєстрацію користувачеві.

2. Products List View – екран програми, що відповідає за категорований список продуктів.

3. Favourite Products List View – екран програми, що відповідає за вибрані продукти користувача.

4. Product View – Екран програми, що відповідає за деталізовану інформацію про продукт.

5. Barcode Scanner View – екран програми, що відповідає за сканер штрих-кодів.

6. Profile View – Екран програми, відповідальний за інформацію про користувача.

7. Request View – Екран програми, призначений для відправки запиту на додавання продукту в базу за його відсутності.

Було виділено 6 контролерів (controllers) для програми.

1. Authentication Controller – контролер, який відповідає за аутентифікацію та реєстрацію користувачеві.

2. Products List Controller – Контролер, який відповідає за роботу користувача зі списками продуктів.

3. Product Controller – Контролер, що відповідає за роботу користувача на екрані докладного опису продукту.

4. Barcode Scanner Controller – контролер, який відповідає за сканування штрих-коду.

5. Profile Changing Controller – контролер, який відповідає за зміна інформації у профілі користувача.

6. Request Controller – контролер, відповідальний надсилання запиту додавання товару у базу.

Інші компоненти системи включають:

Local Database – Локальна база даних. У ній зберігаються дані користувача, а також дані про продукти, які завантажуються з сервера.

Analyze Product – компонент, відповідальний аналіз складу товарів визначення наявності у ньому інгредієнтів, які під обмеження пользователя.

Create Products List – компонент, відповідальний формування категорованого списку продуктів з урахуванням обмежень користувача.

Сервер системи складається з REST -сервісу, авторизаційної бази даних (Firebase Authentication DB) та бази даних, що зберігає відомості про продукти та обмеження користувачів. За допомогою REST -сервісу забезпечується взаємодія між клієнтом та базою даних продуктів та переваг користувачів.

## 2.3 Проєктування бази даних мобільного додатку

На рис. 2.3 представлена діаграма сутностей бази даних продуктів та обмежень користувачів системи вибору продуктів харчування. На схемі позначеннями виділено первинні ключі, зовнішні ключі зображені зв‘язками.

База даних системи складається із 9 таблиць. Розглянемо докладніше кожну їх.

1. Таблиця User складається з єдиного атрибуту ID, оскільки для реєстрації в системі використовується компонент авторизаційна база даних. Цей компонент має власну базу даних, в якій зберігаються електронна пошта та пароль користувача.

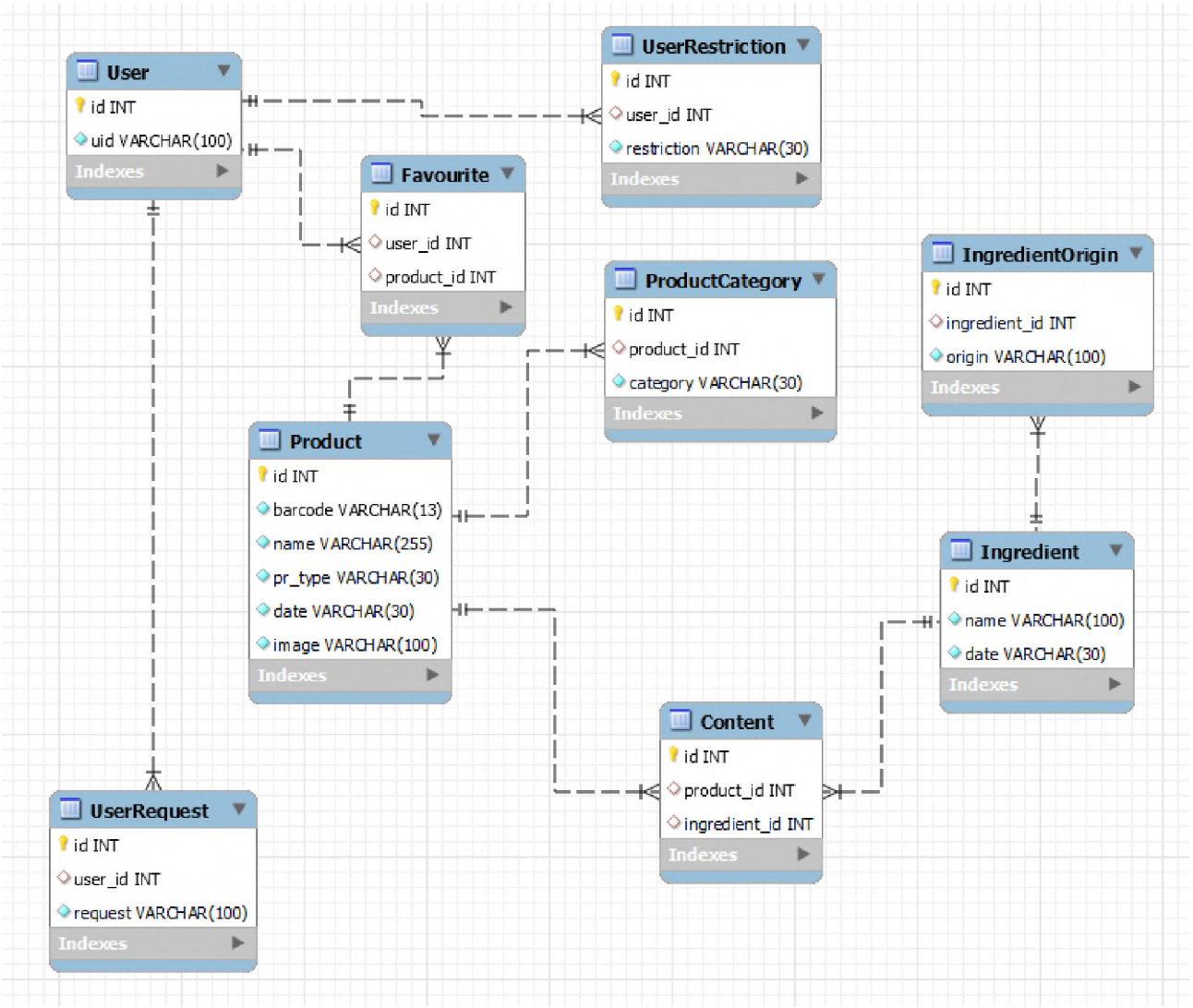


Рисунок 2.3 – Діаграма сутностей бази даних продуктів та обмежень користувачів

2. Таблиця User restrictions складається із трьох атрибутів: первинний ключ id, зовнішній ключ uid (id користувача) та атрибут Restriction. Ця таблиця зберігає обмеження користувачів.

3. Таблиця UserRequest складається із трьох атрибутів: первинний ключ id, зовнішній ключ uid (id користувача) та атрибут Request. Ця таблиця зберігає запити користувачів додавання продуктів у базу.

4. Таблиця Product складається з шести атрибутів: первинний ключ id, Name (назва продукту), Barcode (штрих-код продукту), Type (тип продукту, наприклад, кондитерський виріб), Image (зображення продукту) та Date (дата додавання продукту до бази). Таблиця призначена для зберігання інформації про

продукти. Атрибут Date служить для того, щоб користувач міг оновлювати локальну базу даних, не завантажуючи кожен раз усі дані із сервера. При оновленні будуть завантажені лише дані, які були додані після останнього оновлення локальної бази даних.

5. Таблиця ProductCategory складається з трьох атрибутів: первинний ключ id, зовнішній ключі pid (id продукту) та атрибут Category. Ця таблиця призначена для зберігання категорій товарів (веган, халяль, тощо.)

6. Таблиця Favourites складається із трьох атрибутів: первинний ключ id, зовнішній ключ pid (id продукту) та зовнішній ключі uid (id користувача). Таблиця призначена для зберігання інформації про вибрані продукти користувачів.

7. Таблиця Ingredients складається з трьох атрибутів: первинний ключ id, атрибут Name (назва інгредієнта) та атрибут Date (дата додавання інгредієнта до бази).

8. Таблиця IngredientsOrigin складається з трьох атрибутів: первинний ключ id, зовнішній ключ iid (id інгредієнта) та Origin (походження інгредієнта). Як атрибут Origin може бути зазначено, з чого зроблений даний інгредієнт (наприклад, мигдаль), або шлях походження інгредієнта (рослинний, тваринний).

9. Таблиця Content складається із трьох атрибутів: первинний ключ id, зовнішній ключ pid (id продукту) та зовнішній ключ iid (id інгредієнта). Таблиця призначена для зберігання складу товарів.

Для таблиць бази даних також розроблено зв'язки. Їх особливості наведено нижче.

Таблиця User (Користувачі): має первинний ключ id та унікальне поле uid, що ідентифікує користувачів.

Таблиця Product (Продукти): має первинний ключ id та унікальне поле barcode, а також поля для збереження назви продукту, типу продукту, дати створення та зображення продукту.

Таблиця ProductCategory (Категорії продуктів): має первинний ключ id та зовнішній ключ product_id, який посилається на таблицю Product. Зв'язок FOREIGN KEY (product_id) REFERENCES Product(id) ON DELETE CASCADE означає, що при видаленні запису з таблиці Product відповідні записи у таблиці ProductCategory також будуть видалені.

Таблиця Ingredient (Інгредієнти): має первинний ключ id та поля name (назва інгредієнта) і date (дата створення).

Таблиця IngredientOrigin (Походження інгредієнтів): має первинний ключ id та зовнішній ключ ingredient_id, який посилається на таблицю Ingredient. Зв'язок FOREIGN KEY (ingredient_id) REFERENCES Ingredient(id) ON DELETE CASCADE означає, що при видаленні запису з таблиці Ingredient відповідні записи у таблиці IngredientOrigin також будуть видалені.

Таблиця Content (Вміст продуктів): має первинний ключ id та два зовнішні ключі: product_id, який посилається на таблицю Product, та ingredient_id, який посилається на таблицю Ingredient. Зв'язки FOREIGN KEY (product_id) REFERENCES Product(id) ON DELETE CASCADE і FOREIGN KEY (ingredient_id) REFERENCES Ingredient(id) ON DELETE CASCADE означають, що при видаленні запису з таблиць Product або Ingredient відповідні записи у таблиці Content також будуть видалені.

Таблиця UserRestriction (Обмеження користувачів): має первинний ключ id та зовнішній ключ user_id, який посилається на таблицю User. Зв'язок FOREIGN KEY (user_id) REFERENCES User(id) ON DELETE CASCADE означає, що при видаленні запису з таблиці User відповідні записи у таблиці UserRestriction також будуть видалені.

Таблиця Favourite (Улюблені продукти користувачів): має первинний ключ id та два зовнішні ключі: user_id, який посилається на таблицю User, та product_id, який посилається на таблицю Product. Зв'язки FOREIGN KEY (user_id) REFERENCES User(id) ON DELETE CASCADE і FOREIGN KEY (product_id) REFERENCES Product(id) ON DELETE CASCADE означають, що

при видаленні запису з таблиць User або Product відповідні записи у таблиці Favourite також будуть видалені.

Таблиця UserRequest (Запити користувачів): має первинний ключ id та зовнішній ключ user_id, який посилається на таблицю User. Зв'язок FOREIGN KEY (user_id) REFERENCES User(id) ON DELETE CASCADE означає, що при видаленні запису з таблиці User відповідні записи у таблиці UserRequest також будуть видалені.

Отже, основною особливістю зв‘язків між таблицями є використання зовнішніх ключів з правилом ON DELETE CASCADE, яке забезпечує автоматичне видалення пов'язаних записів з підлеглих таблиць при видаленні запису з головної таблиці. Це допомагає підтримувати цілісність даних у базі даних.

Таким чином, на основі виявлених вимог було спроектовано архітектуру системи. Були виявлені компоненти системи та її основні елементи. Було спроектовано архітектуру бази даних та визначено принцип взаємодії компонентів системи. Подальша реалізація системи спиратиметься на розроблену архітектуру.

# РОЗДІЛ 3
# РОЗРОБКА ТА ТЕСТУВАННЯ МОБІЛЬНОГО ДОДАТКУ ДЛЯ ВИБОРУ ПРОДУКТІВ ХАРЧУВАННЯ

## 3.1 Реалізація компонентів сервера

Серверна частина системи є авторизованою базою даних, базою даних продуктів і обмежень користувачів і включає REST-сервіс. Для реалізації авторизованої бази даних було обрано інструмент phpMyAdmin [29]. Цей інструмент є готовим рішенням, що надає базу даних для зберігання ідентифікаторів користувачів, а також бібліотеки для автентифікації користувачів у програмі. phpMyAdmin надає можливість автентифікації номера телефону або адреси електронної пошти при реєстрації користувача та можливість скидання пароля. Крім цього, phpMyAdmin має зручну консоль, що дозволяє керувати доступом користувачів до системи: додати користувача, скинути пароль, вимкнути або видалити обліковий запис. Також консоль дозволяє відстежувати час створення облікового запису та останній вхід до системи. На рис. 3.1 представлена консоль phpMyAdmin.

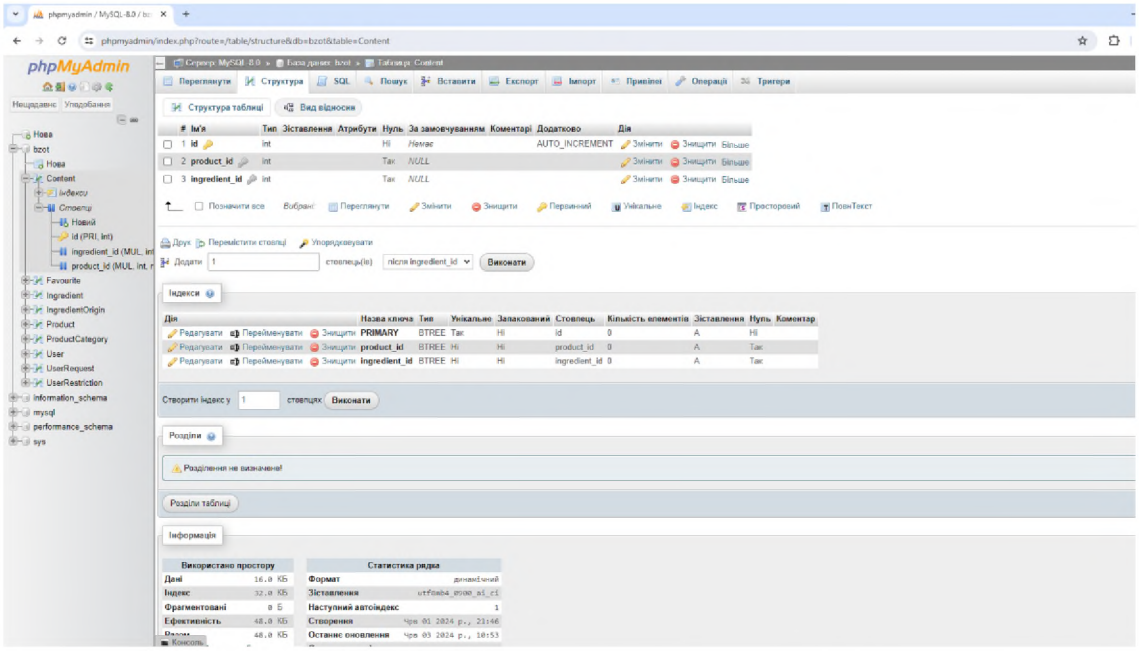


Рисунок 3.1 – Консоль phpMyAdmin

Для реалізації бази даних товарів та обмежень було обрано СУБД MySQL, оскільки вона є однією з найпоширеніших на сьогоднішній день [21], що в майбутньому дозволить полегшити розширення команди розробників. Для управління базою даних та візуалізації її структури використано MySQL Workbench (рис. 3.2). Цей інструмент надає зручний графічний інтерфейс для проєктування, адміністрування та оптимізації бази даних. За допомогою MySQL Workbench можна легко створювати та редагувати таблиці, визначати зв'язки між ними, а також задавати необхідні обмеження. Крім того, Workbench забезпечує потужні засоби для написання та відлагодження SQL-запитів, що значно підвищує ефективність роботи з базою даних.

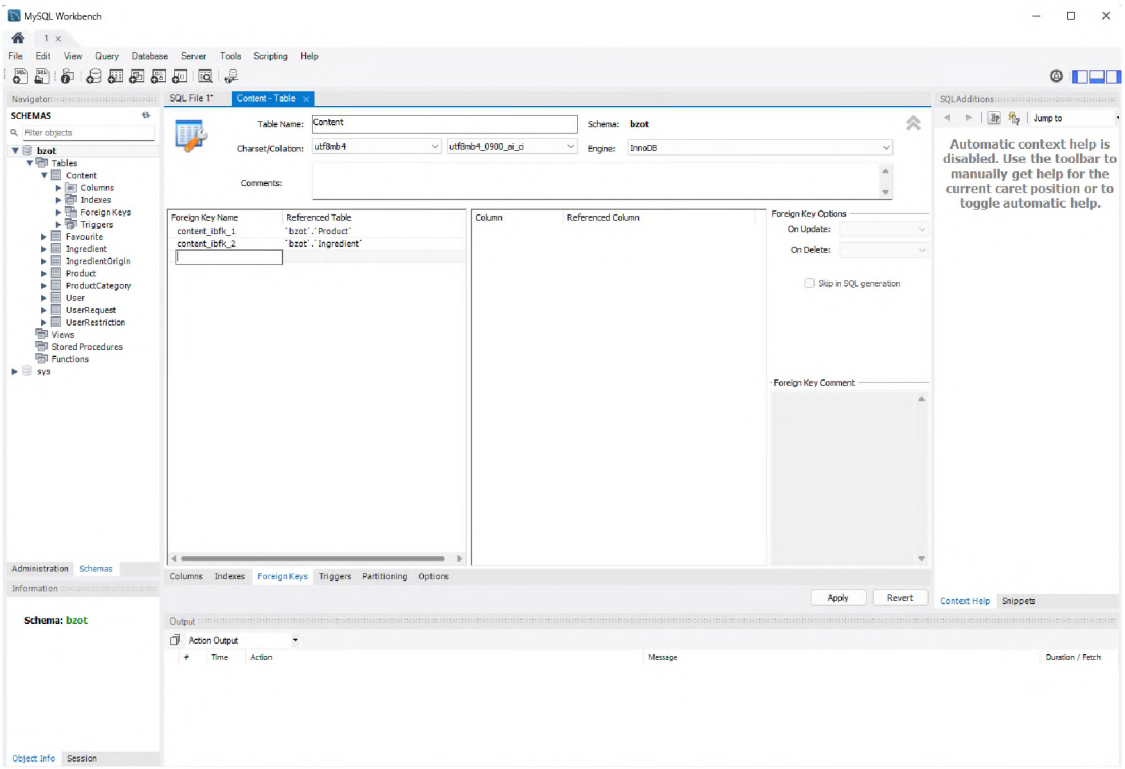


Рисунок 3.2 – Середовище MySQL Workbench

Фреймворк Django, у допомогою якого реалізований REST-сервіс, надає можливість здійснювати більшість взаємодій з базою даних за допомогою механізму об'єктно-орієнтованого відображення (Object-Relational Mapper або ORM) – функціональності, наявної, крім Django, та в інших сучасних інфраструктурах веброзробки [30].


Рисунок 3.3 – Фреймворк Django

Системи ORM набувають все більшої популярності серед розробників, оскільки вони автоматизують безліч типових взаємодій з базою даних і використовують знайомий об'єктно-орієнтований підхід замість інструкцій SQL [22]. Для створення сутностей бази даних продуктів та обмежень користувачів використано код, представлений у додатку А.

Для забезпечення взаємодії програми та бази даних продуктів та обмежень користувачів у REST-сервісі реалізовані функції, які дозволяють робити запит до БД, а також уявлення, які при запиті на відповідну адресу повертають результати функцій у вигляді тексту або JSON-об'єктів. Функції зберігаються у файлі query.py , а представлення – у файлі views.py (рис. 3.4). Функції додавання інформації до бази даних продуктів та обмежень користувачів, а також функції взаємодії з додатком та їх подання реалізовані через код, UML діаграми якого наведені у додатку А.

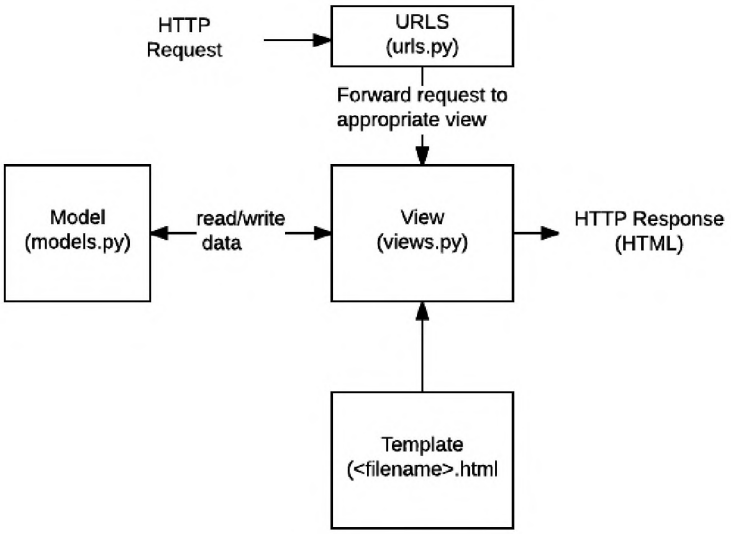


Рисунок 3.4 – Обробка файлів при взаємодії клієнта та сервера

Взаємодія відбувається за допомогою HTTP-запиту з параметрами на URL-адресу, яка відповідає за подання. URL-адреси зберігаються у файлі urls.py (рис. 3.4).

На сервері у папці media зберігаються зображення продуктів. Завантаження зображень здійснюється також за допомогою HTTP-запиту на URL-адресу. Імена зображень формуються так: <штрих-код продукту>. jpg .

### 3.2 Проєктування інтерфейсу мобільного додатку

Мобільний додаток вибору продуктів харчування забезпечує можливість використання без підключення до Інтернету. Для цього в локальній базі даних зберігається інформація з сервера. Локальна база даних у цілому повторює структуру бази даних продуктів та обмежень користувачів, але локальна БД також містить інформацію про час останньої синхронізації із сервером. Крім локальної БД, додаток використовує зберігання даних Shared Preferences. Shared Preferences дозволяє зберігати на пристрої невелику кількість даних у форматі ключ/значення [9].

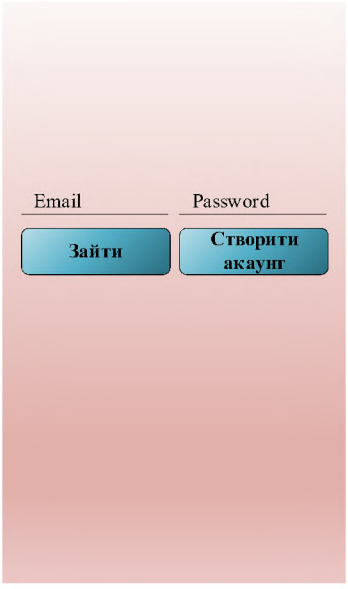


Рисунок 3.5 – Екран автентифікації у додатку

При запуску програми користувач потрапляє на екран аутентифікації, на якому йому пропонується увійти в обліковий запис або зареєструватися в програмі. На рис. 3.5 представлений екран автентифікації у додатку. Поле email здійснює перевірку формату введених даних. Поле password перевіряє складність пароля. У разі неправильно введених даних користувачеві буде виведено відповідне повідомлення. Після успішної аутентифікації у програмі користувач потрапляє на головний екран.

Головний екран програми є чотири фрагменти, перемикання між якими здійснюється за допомогою навігаційного меню внизу екрана. Перший фрагмент – екран списку дозволених продуктів. Фрагмент складається із вкладок, що представляють категорії продуктів, а також кнопки «оновити каталог». Кожна вкладка містить список дозволених продуктів, що належать даній категорії, та рядок пошуку. При натисканні на елемент списку відбувається перехід на сторінку з детальною інформацією про продукт. На рис.3.6 представлений екран списку категорій дозволених продуктів.


Рисунок 3.6 – Екран списку дозволених продуктів

При натисканні на кнопку «оновити каталог» створюється об'єкт класу UpdateLocalDB , який здійснює оновлення локальної бази даних. За відсутності підключення до Інтернету виводиться відповідне повідомлення. Клас UpdateLocalDB є спадкоємцем класу AsyncTask , призначеного для виконання складних завдань у фоновому режимі. UpdateLocalDB реалізує методи doInBackground (виконує дії у фоновому потоці, не має доступу до графічного інтерфейсу) та onPostExecute (виконується після doInBackground , має доступ до графічного інтерфейсу). Також UpdateLocalDB реалізує методи взаємодії з сервером: отримання списку продуктів (GetAllProducts) та їх складів (GetContent); отримання списку інгредієнтів (GetIngredients) та їх походження (GetIngredientsOrigin); отримання категорій продуктів (GetProductsCategories). Якщо користувач вже оновлював локальну базу даних, при натисканні на кнопку «оновити каталог» будуть завантажені лише ті дані, які були додані після останнього оновлення. Після виконання методу doInBackground відбувається оновлення головного екрана. У додатку А представлені UML діаграми реалізації методу doInBackground.

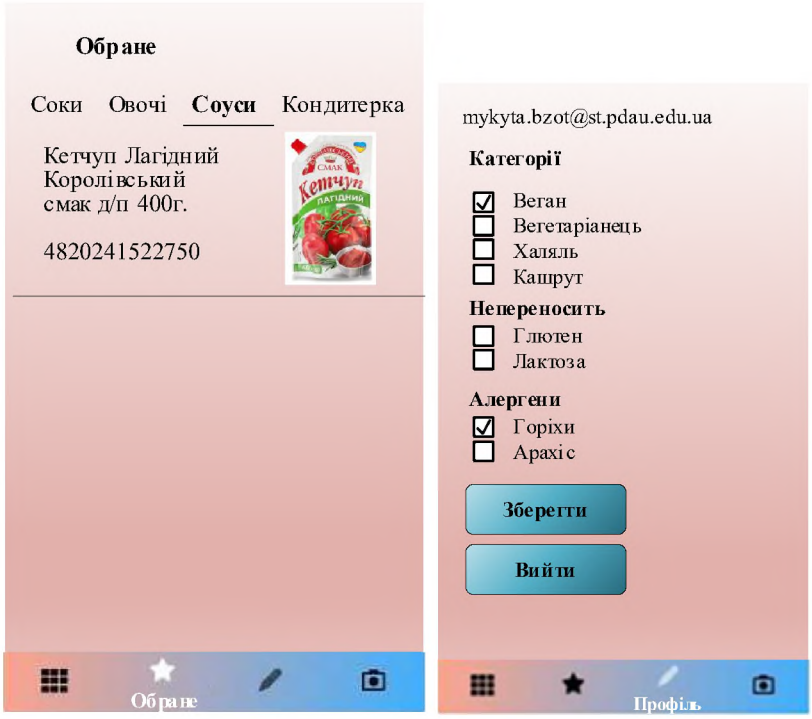


Рисунок 3.7 – Екрани «Обране» та «Профіль»

На другому фрагменті розташований список категорій обраних продуктів. Фрагмент є спрощеною версією першого фрагмента, що складається тільки з вкладок з продуктами, розбитими на категорії.

На третьому фрагменті знаходиться профіль користувача. Профіль є список обмежень, з якого користувач може вибрати потрібні, кнопку «зберегти» і кнопку «вийти». Якщо спробувати зберегти зміни без підключення до Інтернету, буде виведено відповідне повідомлення, оскільки для синхронізації з сервером потрібне інтернет-з'єднання. На рис. 3.7 представлені екран «Обране» та екран «Профіль».

На четвертому фрагменті розташована кнопка Сканувати, при натисканні на яку відбудеться перехід до сканера. На рис. 3.8 представлений фрагмент «Сканер» та екран сканера.

Сканер реалізований за допомогою SDK Firebase ML Kit. Сканер працює в режимі реального часу і під час знаходження штрих-коду потрібного формату (EAN-8 або EAN-13) відкриває сторінку продукту. Для обробки кадрів використовується бібліотека com.otaliastudios:cameraview. Кадр обробляється кожні 0,2 секунди. У додатку А подано UML діаграму реалізації сканера.

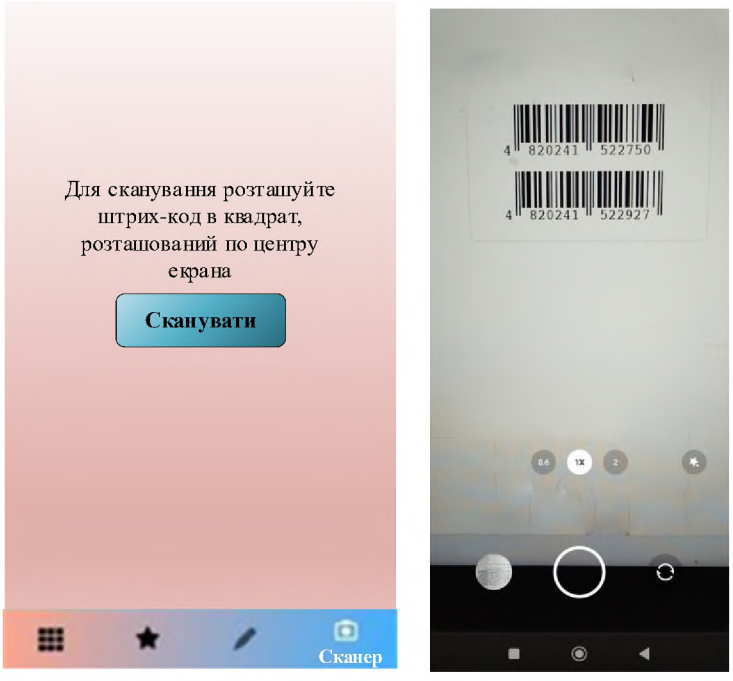


Рисунок 3.8 – Екран фрагмента «Сканер» та екран сканера

При успішному зчитуванні штрих-коду у програмі відкривається новий екран. Якщо продукт буде знайдено в базі, то з'явиться екран з детальною інформацією про цей продукт. Інакше відкриється екран із пропозицією надіслати запит на внесення продукту до бази.

Екран із детальною інформацією про продукт містить назву продукту, його склад, зображення та короткий вердикт про придатність даного продукту для користувача. Також на екрані знаходиться кнопка, що дозволяє додати продукт до вибраного. Якщо продукт вже знаходиться у вибраному, то натискання кнопки видалить його звідти. Якщо продукт було додано до вибраного або видалено з обраного під час відсутності Інтернет-з'єднання, інформація про це заноситься в SharedPreferences. Це дозволить при наступному з'єднанні з мережею синхронізуватися із сервером без втрати даних. На рис. 3.9 представлені екрани з детальною інформацією про продукт та з пропозицією надіслати запит на додавання продукту до бази.

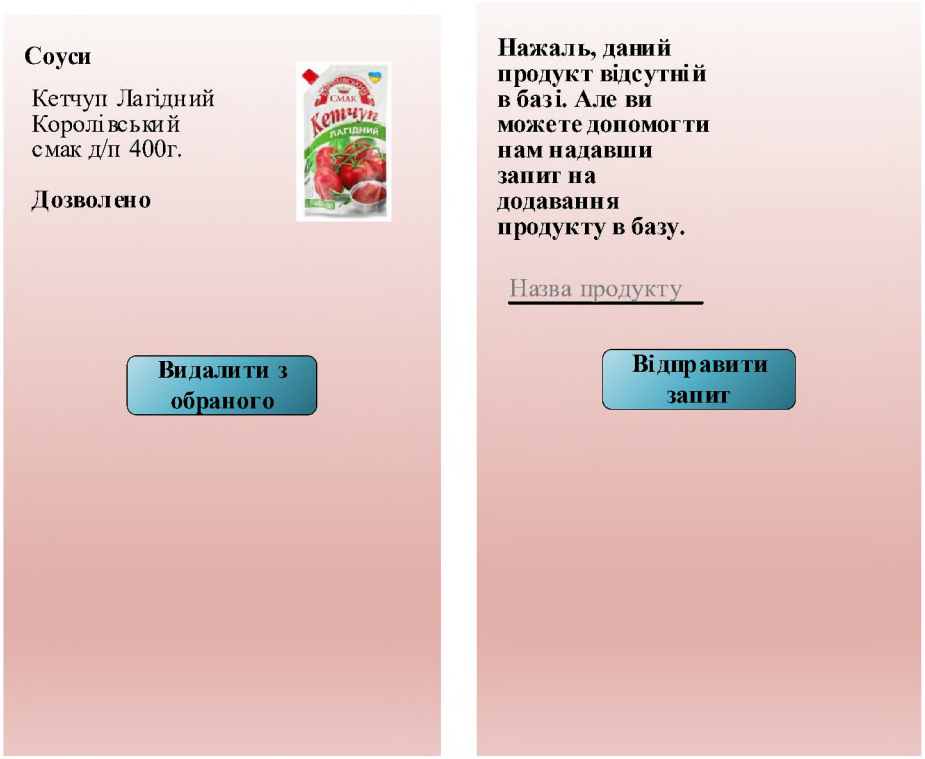


Рисунок 3.9 – Екран з детальною інформацією про продукт та екран для надсилання запиту на додавання продукту до бази

У результаті, запропонована структура мобільного додатку забезпечує зручний та інтуїтивно зрозумілий інтерфейс для користувачів. Використання локальної бази даних та зберігання даних у Shared Preferences гарантує надійність та безперервну роботу додатку навіть за відсутності Інтернет-з‘єднання.

## 3.3 Розрахунок витрат забезпечення функціонування мобільного додатку

Витратна частина створення мобільного додатку складається з таких витрат як: витрати по електроенергії, витрати по розміщенню серверної частини в мережі інтернет (хостинг), заробітна плата програмістів та інші витрати на канцелярські товари та витратні матеріали для комп'ютерів, оренду приміщення, амортизації комп‘ютерів і оргтехніки та інші витрати.

Розрахунок електроенергії виконується за тарифом для підприємств 1 кВт/г = 4,32 грн. Тоді в місяць вартість оплати підприємства складе 1263,6 грн.

У мережі серверну частину мобільного додатку планується розмістити на ресурсах провайдера міста Полтава, що забезпечить зручне обслуговування.

Розрахунок щомісячних витрат на підтримку серверу наведено в табл. 3.1.

Таблиця 3.1 – Розрахунок щомісячних витрат на підтримку серверної частини мобільного додатку

| Найменування | Сума, грн | ЄСВ, грн |
|---|---|---|
| Зарплата програміста | 14000 | 422,65 |
| Зарплата кур’єра | 2000 | 422,65 |
| Транспортні витрати кур’єра | 350 | × |
| Електроенергія | 1263,6 | × |
| Хостинг | 200 | × |
| Інтернет | 100 | × |
| Інші витрати | 300 | × |
| Разом: | 8213,6 | 845,3 |
| Всього витрат: | 19058,9 | |

Таблиця 3.2 – Розрахунок щомісячних матеріальних витрат

| Найменування | Сума, грн/міс. |
|---|---|
| Електроенергія | 1263,6 |
| Хостинг | 200 |
| Інтернет | 100 |
| Інші витрати | 300 |
| Разом: | 1863,6 |

Витрати на період розробки програмного продукту складають $З_{пр}$ =847,1 грн.

Розрахуємо собівартість програмного продукту за формулою:

$$С_{ст} = З_{пр} + ЗП_{пр} + ЕСВ + А . \quad (3.1)$$

Тоді $С_{ст}$ – собівартість розробки мобільного додатку складе:

$С_{ст}$ = 847,1+1818+422,65+108,5 = 3196,25 гривень.

### 3.4 Розробка планів тестування системи вибору продуктів харчування

Тестування системи складається з функціонального тестування сервера та мобільного додатку. Функціональне тестування сервера полягає у перевірці коректності роботи функцій REST -сервісу. Оскільки взаємодія з REST -сервісом здійснюється за допомогою HTTP -запитів, всі дії є найпростішими і не можуть бути поділені на кроки. У таблиці 3.3 наведено план етапів тестування основних функцій REST -сервісу.

Функціональне тестування програми полягає у перевірці коректності роботи всіх варіантів використання. Оскільки однією з головних можливостей програми є сканування штрих-коду продукту для подальшого аналізу та винесення вердикту, розглянемо цей прецедент докладніше для користувача, який користується цією програмою вперше.

При запуску програми користувачеві необхідно зареєструватись у системі. Після успішної реєстрації користувач потрапляє на головний екран, на якому пропонується завантажити каталог продуктів для подальшої роботи програми.

Після завантаження каталогу у користувача з'являється можливість переглядати список категорій продуктів.

Таблиця 3.3 – Етапи функціонального тестування REST -сервісу

| № | Передумова | Дія | Очікуваний результат |
|---|---|---|---|
| 1. | Користувача із заданим uid не існує у БД | Створення користувача у БД | Користувач доданий до БД |
| 2. | - | Перевірка на існування користувача | Якщо користувач існує – true, інакше – false |
| 3. | Продукт відсутній у БД | Додавання продукту до БД | Продукт доданий до БД |
| 4. | Продукт існує у БД | Видалення продукту з БД | Продукт та вся інформація про нього видалені з БД |
| 5. | Продукт існує у БД | Отримання продукту із БД | Інформація про продукт у форматі JSON |
| 6. | - | Одержання всіх продуктів із БД | JSON – масив з інформацією про всі продукти в базі |
| 7. | - | Отримання продуктів, доданих після певної дати | JSON – масив з інформацією про продукти, додані після певної дати |
| 8. | - | Видалення всіх продуктів із БД | Таблиця з продуктами порожня |

Далі користувачу необхідно встановити обмеження на вкладці «Профіль» та натиснути кнопку «Зберегти». Після цього користувачеві необхідно перейти на вкладку «Сканер» та натиснути кнопку «Сканувати».

Після натискання на кнопку «Сканувати» користувач потрапляє на екран сканера. Програма запитує у користувача дозвіл на використання камери. Далі за допомогою камери пристрою користувач сканує штрих-код продукту, після чого потрапляє на екран з інформацією про цей продукт.

Таким чином, розроблений мобільний додаток для вибору продуктів харчування дає користувачам можливість шукати та обирати продукти харчування, які відповідають їхнім потребам та обмеженням. Додаток містить базу даних продуктів з інформацією про їх склад, харчову цінність та можливі

алергени. Користувачі можуть налаштувати фільтри, щоб бачити лише ті продукти, які відповідають їхнім потребам, сканувати штрих-коди продуктів, щоб отримати інформацію про них. Це зручний спосіб швидко знайти інформацію про продукт, який ви купуєте.

Додаток дозволяє користувачам створювати та редагувати списки покупок, що може допомогти їм краще планувати свої покупки. Це означає, що користувачі можуть отримувати доступ до своїх даних з будь-якого пристрою. Додаток також має зручний та інтуїтивно зрозумілий інтерфейс, що робить його простим у використанні для користувачів будь-якого віку та рівня досвіду. Загалом, розроблений мобільний додаток є цінним інструментом для людей, які хочуть харчуватися здорово та уникати продуктів, які їм не підходять.

# ВИСНОВКИ

В результаті виконання роботи була досягнута її початкова мета та вирішені поставлені завдання. Було проаналізовано основну наукову, методичну та нормативну літературу з теми програмування мобільних додатків з клієнт-серверною архітектурою.

Розглянуто стандарти та системи позначення товарів: проаналізовано систему GS1 та її коди (GTIN, EAN-13, EAN-8) для маркування товарів, а також QR-коди та інші типи кодування. Проаналізовано наявні на ринку мобільні додатки для розпізнавання штрих-кодів: проаналізовано Firebase ML Kit, ZXing, Google Lens, QR & Barcode Scanner, MyFitnessPal, Yuka, ShopSavvy та інші додатки, а також їхні можливості та недоліки. Визначено функціональні (можливості для користувачів) та нефункціональні (технічні) вимоги до розроблюваної системи. На основі проведеного аналізу зроблено висновок, що на ринку мобільних додатків не існує рішень, які повністю відтворюють функціонал системи, що розробляється.

Деталізовано архітектуру мобільного додатку: представлено схему MVC, що лежить в основі розробки, а також описано взаємодію компонентів (модель, вигляд, контролер) та функціональні можливості кожного з них. Обґрунтовано вибір REST-архітектури, описано послідовність взаємодії мобільного додатку з сервером, а також наведено діаграму компонентів системи. Представлено ER-діаграму, що описує структуру бази даних, а також детально описано кожну з таблиць та зв'язки між ними.

Зроблено висновок, що розроблена архітектура та проектування бази даних відповідають вимогам до системи вибору продуктів харчування. Використання RESTful API, клієнт-серверної архітектури та локальної бази даних забезпечує гнучкість, масштабованість, автономність та надійність системи.

Описано розробку та планування тестування мобільного додатку для вибору продуктів харчування. Розроблено серверну частину системи, яка включає авторизовану базу даних, базу даних продуктів та обмежень

користувачів, а також REST-сервіс. Запропоновано структуру мобільного додатку, яка забезпечує зручний та інтуїтивно зрозумілий інтерфейс для користувачів. Розроблено плани тестування системи вибору продуктів харчування.

Використання локальної бази даних та зберігання даних у Shared Preferences гарантує надійність та безперервну роботу додатку навіть за відсутності Інтернет-з'єднання. Запропонований мобільний додаток є цінним інструментом для людей, які хочуть харчуватися здорово та уникати продуктів, які їм не підходять.

Таким чином, поставлені задачі розв‘язано у повному обсязі. Напрямком подальших досліджень є вирішення питання забезпечення безпеки системи. Необхідно провести додатковий аналіз та розробити заходи щодо захисту даних користувачів та запобігання несанкціонованому доступу до системи. Також можливе розширення функціональності системи в майбутньому, наприклад, додавання можливості сканування QR-кодів, персональних рекомендацій продуктів, функцій соціальної взаємодії тощо.