МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління

Пазірович Юрій Іванович

Мобільний додаток для оцінювання екологічного впливу продуктів / Mobile Application for Assessing the Environmental Impact of Products

Спеціальність: 122 – Комп'ютерні науки
Освітньо-професійна програма – Комп'ютерні науки

Кваліфікаційна робота

Виконав студент групи КН-42

Ю. І. Пазірович

_____

Науковий керівник

к.е.н., Г. М Гладій

_____

Кваліфікаційну роботу допущено до захисту
«___»_____ 2025 р.

В.о. завідувача кафедри
_____ Н. М. Васильків

Тернопіль – 2025

Факультет комп’ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «бакалавр»
спеціальність 122 – Комп’ютерні науки
освітньо-професійна програма – Комп’ютерні науки

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Н. М. Васильків
«____» _____2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Пазірович Юрій Іванович

1. Тема кваліфікаційної роботи: Мобільний додаток для оцінювання екологічного впливу продуктів / Mobile application for evaluating the ecological impact of products
керівник роботи Г.М. Гладій, к.т.н., доцент
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від 20 грудня 2024 р. № 938.
2. Строк подання студентом закінченої кваліфікаційної роботи 25 травня 2025 р.
3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.
4. Основні питання, які потрібно розробити:
– проаналізувати мобільні застосунки для визначення екологічного сліду харчових продуктів, виявити їхні функції, обмеження та способи взаємодії з користувачем.;
– сформувати концепцію мобільного застосунку,
– розробити архітектуру системи: мобільний клієнт, сервер (API), локальна БД з урахуванням безпеки, масштабованості та гнучкості;
– реалізувати алгоритм з нечітким пошуком (fuzzy search) та розрахунком екологічного впливу на основі ваги продукту;
– провести тестування та верифікацію функціональності застосунку на Android та iOS.
5. Перелік графічного матеріалу в роботі:
– архітектура клієнт-серверної взаємодії застосунку;
– схема взаємодії мобільного застосунку з API та базою даних;
– діаграма з прикладом обробки запиту користувача до серверної частини.

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 20 грудня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Затвердження теми кваліфікаційної роботи, ознайомлення з літературними джерелами та складання плану роботи	до 01.01. 2025 р.	
2	Написання 1 розділу кваліфікаційної роботи	до 01.02. 2025 р.	
3	Написання 2 розділу кваліфікаційної роботи	до 01.04.2025 р.	
4	Написання 3 розділу кваліфікаційної роботи	до 01.05. 2025 р.	
5	Представлення попереднього варіанту кваліфікаційної роботи, перевірка та внесення змін керівником	до 15.05.2025 р.	
6	Опрацювання зауважень та представлення завершеного варіанту кваліфікаційної роботи. Підготовка супроводжуючих документів	до 25.05.2025 р.	
7	Перевірка кваліфікаційної роботи на оригінальність тексту	до 30.05.2025 р.	
8	Оформлення кваліфікаційної роботи та отримання допуску до захисту	до 10.06.2025 р.	
9	Подання кваліфікаційної роботи до захисту на засіданні атестаційної комісії	до 10.06. 2025 р.	

Студент_____ Ю. І. Пазірович_____
(підпис) (прізвище та ініціали)

Керівник кваліфікаційної роботи_____ Г. М. Гладій_____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота на тему «Мобільний додаток для оцінювання екологічного впливу продуктів» на здобуття освітнього ступеня «бакалавр» зі спеціальності 122 «Комп'ютерні науки» освітньої програми «Комп'ютерні науки» написана обсягом 47 сторінок і містить 34 ілюстрації, 1 додаток та 35 використаних джерел.

Метою роботи є розробка мобільного застосунку, що дає змогу користувачу отримати оцінку екологічного сліду продуктів харчування на основі аналізу введених даних. У межах дослідження реалізовано взаємодію між клієнтською частиною на основі React Native та серверною частиною, розробленою з використанням Django REST Framework.

Для реалізації функціоналу було використано метод аналізу (для вивчення аналогів і вимог), метод синтезу (для поєднання технологій), методи проєктування та моделювання баз даних, а також експериментальні методи тестування системи на мобільному пристрої.

У результаті роботи розроблено повнофункціональний мобільний застосунок, що дає змогу проводити реєстрацію, автентифікацію, вводити дані про продукти, аналізувати екологічні метрики, зберігати історію та взаємодіяти з API сервером. Сформовано архітектуру застосунку, запропоновано схему бази даних та протестовано застосунок на реальному пристрої.

Результати можуть бути використані у навчальних цілях, а також як прототип для розробки подібних екологічно орієнтованих мобільних систем.

Ключові слова: МОБІЛЬНИЙ ЗАСТОСУНОК, REACT NATIVE, DJANGO, ЕКОЛОГІЧНИЙ ВПЛИВ, API, БАЗА ДАНИХ, АУТЕНТИФІКАЦІЯ, ЕКОЛОГІЧНИЙ СЛІД

ANNOTATION

Qualification work on the topic "Mobile Application for Evaluating the Ecological Impact of Products" for obtaining the Bachelor's degree in specialty 122 "Computer Science" of the educational program "Computer Science" consists of 47 pages and includes 34 figures, 1 appendices, and 35 sources.

The purpose of the work is to develop a mobile application that allows users to assess the environmental footprint of food products based on entered data. The project implements interaction between the client side, developed using React Native, and the server side, built with Django REST Framework.

The following methods were used: analysis (for reviewing analogs and requirements), synthesis (for combining technologies), database design and modeling, and experimental testing on a mobile device.

As a result, a fully functional mobile application was developed, enabling registration, authentication, input of product data, analysis of environmental metrics, saving history, and communication with the API server. The architecture of the application and the database schema were created, and the application was tested on a real device.

The results can be used for educational purposes and as a prototype for further development of similar environmentally focused mobile systems.

Keywords: MOBILE APPLICATION, REACT NATIVE, DJANGO, ENVIRONMENTAL IMPACT, API, DATABASE, AUTHENTIFICATION, ENVIRONMENTAL FOOTPRINT

ЗМІСТ

Вступ 7
1 Критичний аналіз існуючих мобільних рішень у сфері екологічного впливу. 10
1.1 Аналіз сучасного стану екологічного впливу продуктів 10
1.2 Аналіз існуючих мобільних додатків у сфері екології 12
1.3 Вибір методології та технологічної платформи 14
2 Проєктування основних складових додатку 17
2.1 Розробка архітектури мобільного додатку 17
2.2 Проєктування бази даних 20
2.3 Розробка інтерфейсу користувача 23
3 Програмна реалізація мобільного додатку 27
3.1 Розробка серверної та клієнтської частини додатку 27
3.2 Інтеграція системи оцінювання екологічного впливу 32
3.3 Тестування та впровадження мобільного додатку 36
Висновки 39
Список використаних джерел 41
Додаток А Копія публікації автора 44

ВСТУП

У сучасних умовах поглиблення кліматичних змін та деградації природного середовища, екологічна відповідальність стає критично важливим фактором у всіх сферах людської діяльності. Проблеми, пов’язані з нераціональним використанням ресурсів, надмірним виробництвом та неефективною утилізацією відходів, вимагають впровадження нових інструментів, які сприятимуть формуванню свідомого споживання серед населення. Одним із шляхів досягнення цієї мети є створення цифрових рішень, що дозволяють швидко та наочно оцінити екологічний вплив окремих продуктів.

Продукти споживання проходять складний життєвий цикл – від добування сировини до утилізації. На кожному з етапів відбувається негативний вплив на довкілля, однак кінцевий користувач, як правило, не має можливості оцінити ці наслідки. Через це, у більшості випадків, вибір товару відбувається без урахування екологічної складової. Враховуючи зростання зацікавленості громадськості у питаннях сталого розвитку, існує нагальна потреба в розробленні програмного забезпечення, яке б надавало користувачам простий доступ до інформації про екологічні параметри товарів.

Аналіз існуючих мобільних застосунків у сфері екологічної оцінки засвідчив, що більшість із них мають обмежену функціональність або орієнтовані лише на певні категорії товарів, регіони або бази даних. У багатьох рішеннях інформація подається у складній або надмірно технічній формі, що ускладнює її сприйняття пересічними користувачами. Також спостерігається нестача універсальних платформ, які б узагальнювали показники екологічного впливу на основі методології оцінки життєвого циклу (Life Cycle Assessment, LCA). Це створює передумови для розроблення нового застосунку, здатного забезпечити комплексний, наочний та зручний для споживача підхід до аналізу екологічного сліду продуктів.

Метою кваліфікаційної роботи є розробка мобільного додатку, що забезпечує оцінювання екологічного впливу споживчих продуктів з

використанням принципів LCA[26], та надає користувачам доступну візуалізацію ключових екологічних параметрів.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

– провести критичний аналіз сучасного стану екологічного впливу споживчих товарів;

– здійснити огляд та оцінювання існуючих мобільних рішень у відповідній сфері;

– визначити та обґрунтувати методологію екологічного оцінювання продуктів;

– обрати технологічну платформу для реалізації мобільного застосунку;

– розробити архітектуру мобільного додатку з урахуванням масштабованості та зручності використання;

– спроєктувати базу даних для зберігання екологічної інформації;

– реалізувати інтерфейс користувача відповідно до принципів UX/UI дизайну;

– реалізувати серверну частину системи засобами Django REST Framework;

– протестувати додаток та оцінити його придатність для практичного використання.

Об’єктом дослідження є процес оцінювання екологічного впливу товарів з урахуванням параметрів життєвого циклу. Предметом дослідження виступають програмні засоби, в яких реалізовано вказане оцінювання.

Розробка програмного забезпечення здійснена з використанням фреймворку React Native[2] у поєднанні з Expo[13], що дозволяє забезпечити кросплатформенну підтримку, передусім для Android-пристроїв. Клієнтська частина взаємодіє з серверною через REST API, реалізоване засобами Django REST Framework [10], що забезпечує масштабовану структуру взаємодії з базою даних і підтримку майбутнього розширення функціональності.

Методологічною основою дослідження є об'єктно-орієнтований підхід до програмування, модульна архітектура та методи проектування користувацького

інтерфейсу. Для аналізу екологічного впливу застосовується методика LCA, адаптована до потреб кінцевого споживача.

Практичне значення кваліфікаційної роботи полягає у створенні інструменту, який дає змогу користувачам швидко оцінити екологічну ефективність товарів і, відповідно, ухвалювати більш відповідальні рішення щодо споживання. Крім того, додаток може бути корисним для екологічних організацій, освітніх установ, маркетингових кампаній та цифрових сервісів зі сталого розвитку.

Результати дослідження опубліковано в матеріалах міжнародної мультидисциплінарної наукової інтернет-конференції «Світ наукових досліджень. Випуск 41», м. Тернопіль – Україна. Ополе – Польща, 28-29 травня 2025 р. (додаток А).

1 КРИТИЧНИЙ АНАЛІЗ ІСНУЮЧИХ МОБІЛЬНИХ РІШЕНЬ У СФЕРІ ЕКОЛОГІЧНОГО ВПЛИВУ

1.1 Аналіз сучасного стану екологічного впливу продуктів

Зміни клімату, деградація ґрунтів, зменшення прісноводних ресурсів та втрата біорізноманіття є глобальними викликами сучасності. Однією з найвагоміших причин цих явищ є діяльність агропродовольчого сектору – вирощування, переробка, транспортування і споживання харчових продуктів.

Харчова промисловість становить суттєву частку світового екологічного навантаження, зокрема через викиди парникових газів, використання водних та земельних ресурсів, а також насичення довкілля органічними та неорганічними забруднювачами. У зв’язку з цим вивчення екологічного впливу продуктів є однією з ключових умов для формування стратегії сталого споживання.

За даними ресурсу Our World in Data [24], виробництво харчових продуктів становить близько 26% загального обсягу викидів парникових газів. З цієї частки найбільше навантаження припадає на тваринницьку продукцію, зокрема червоне м’ясо (яловичина, баранина), птицю, а також молочні вироби. Наприклад, виробництво 1 кг курятини (chicken thighs) супроводжується утворенням близько 409 кг CO_2 -еквіваленту, споживанням понад 23 666 літрів води, а також використанням 403 м² землі.

Ці показники демонструють масштаб екологічного сліду навіть порівняно «легкої» тваринної продукції, не кажучи вже про інші категорії. У той же час фрукти, овочі та зернові мають значно нижчі значення за всіма критеріями, хоча й не позбавлені проблем – наприклад, вирощування рису супроводжується значними викидами метану, а банани та авокадо часто транспортуються на великі відстані, що також підвищує їх вуглецевий слід.

Для кількісної оцінки екологічного навантаження, що створюється в процесі виробництва, транспортування і споживання харчових продуктів, доцільно виокремити п’ять ключових показників:

– Карбоновий слід (Carbon footprint) – кількість викидів CO_2 та інших парникових газів, що припадає на одиницю продукту (кг або 100 г). Цей показник

дозволяє оцінити внесок товару у глобальне потепління.

– Використання води (Water use) – обсяг прісної води, витраченої на виробництво, включно з поливом, переробкою та очищенням.

– Використання землі (Land use) – площа, необхідна для вирощування/виробництва одного кілограма продукту. Високі значення часто пов'язані зі зменшенням природних екосистем.

– Евтрофікація (Eutrophication) – вплив продукту на забруднення водойм і ґрунтів внаслідок надлишку азотних і фосфорних сполук.

– Дефіцит ресурсів (Resource scarcity) – агрегований показник, що оцінює споживання обмежених природних ресурсів.

Усі ці метрики застосовуються у розробленому мобільному застосунку, де користувач вводить вагу продукту (у кілограмах), а система обчислює відповідні значення. Такий підхід забезпечує гнучкість і персоналізованість оцінки, що особливо важливо у повсякденному використанні.

Оцінювання екологічного впливу ґрунтується на концепції оцінки життєвого циклу (LCA – Life Cycle Assessment) [26], яка передбачає врахування всіх етапів:

– виробництва (добування сировини, аграрна діяльність, переробка);

– транспортування і зберігання;

– споживання (енергія на приготування, упаковка);

– утилізації (відходи, переробка, компостування).

Такий підхід забезпечує повну картину впливу продукту на довкілля, а не лише оцінку одного аспекту (наприклад, CO_2). Застосування LCA [26] у мобільному додатку дозволяє сформувати у користувача розуміння повної «екологічної вартості» звичного продукту.

Незважаючи на наявність великої кількості досліджень і відкритих даних, у більшості споживачів відсутній прямий доступ до екологічної інформації про продукти. Маркування, яке надається виробниками, часто є недостатнім, незрозумілим або маніпулятивним (наприклад, грінвошинг). У цьому контексті цифрові рішення, які формують наочну, прозору і персоналізовану оцінку екологічного впливу, мають стратегічне значення.

Розроблений додаток вирішує цю проблему, забезпечуючи користувача візуалізованими даними (значення в kg CO_2 , L води тощо), які автоматично обчислюються на основі введеної ваги продукту. Використання піктограм, індикаторів та простого інтерфейсу дозволяє зробити навіть складні екологічні показники доступними широкій аудиторії.

Сучасний стан харчового споживання характеризується високим рівнем екологічного навантаження, яке, однак, може бути скориговане за допомогою відповідальної поведінки споживачів. Надання достовірної інформації про вплив окремих продуктів є важливою умовою для зміни споживчих звичок. Аналіз існуючих показників екологічного впливу підтверджує доцільність розробки програмного забезпечення, орієнтованого на доступну подачу таких даних. У цьому контексті мобільний додаток, реалізований у межах даної кваліфікаційної роботи, відіграє роль ефективного просвітницького інструменту, що сприяє формуванню культури сталого споживання.

1.2 Аналіз існуючих мобільних додатків у сфері екології

На сучасному ринку мобільних додатків представлено ряд рішень, спрямованих на інформування користувачів про екологічний вплив продуктів харчування та побутових товарів. Серед найбільш відомих і популярних можна виділити додаток Yuka [18], який отримав широку увагу завдяки можливості швидкого сканування штрих-кодів і оцінці продуктів з позиції здоров’я та екології. Однак, аналіз функціоналу та інтерфейсу таких застосунків показує низку обмежень, які впливають на зручність і доступність інформації для кінцевого користувача.

Yuka [18] пропонує користувачам інструмент сканування штрих-кодів, що дозволяє швидко отримати інформацію про харчову цінність, наявність алергенів, а також екологічний вплив товарів. Такий підхід зручний для роздрібних покупців, але має низку особливостей:

– Необхідність фізично сканувати кожен товар, що ускладнює процес оцінки великої кількості продуктів або планування покупок.

– Інтерфейс і алгоритми подачі даних інколи містять велику кількість складної інформації, що може перевантажувати користувача.

– Обмежена гнучкість у налаштуванні ваги чи кількості продукту, що унеможливлює точний розрахунок екологічного сліду відповідно до індивідуального споживання.

Open Food Facts [16] – це відкрита база даних про харчові продукти, що надає інформацію про склад, харчову цінність і частково екологічний вплив. Додаток орієнтований на збір і поширення даних користувачами, що робить його гнучким, проте:

– вимагає значного часу на пошук і аналіз інформації;

– не завжди має зрозумілий інтерфейс для швидкого користування;

– відсутні автоматизовані розрахунки екологічного впливу з урахуванням кількості продукту.

Giki [21] – це платформа, що оцінює не тільки харчові продукти, а й інші аспекти сталого способу життя, наприклад, одяг, косметику тощо. Вона акцентує увагу на соціальних і екологічних характеристиках продуктів, що сприяє комплексному підходу. Однак, Giki:

– здійснює огляд та оцінювання існуючих мобільних рішень у відповідній сфері;

– вимагає реєстрації та введення великої кількості персональних даних;

– має складний інтерфейс і масив інформації, що може бути перевантаженням для непідготовленого користувача;

– обмежена в локалізації на різні ринки.

Oroeco [9] – додаток, який відстежує особистий карбоновий слід користувача, пропонуючи рекомендації для його зниження. Основна увага приділяється поведінковим аспектам і загальному стилю життя. Проте Oroeco:

– менш орієнтований на детальний аналіз конкретних продуктів харчування;

– потребує введення багатьох даних вручну, що може бути незручним;

– не має простого інтерфейсу для швидкого доступу до основної екологічної інформації.

Відповідно до поставленої мети, розроблений мобільний додаток орієнтований на простоту, інтуїтивну зрозумілість і швидкість отримання ключової екологічної інформації. Основні відмінності та переваги:

– відсутність необхідності сканування штрих-кодів, що пришвидшує роботу користувача. Інформація вводиться вручну у вигляді ваги продукту, що забезпечує гнучкий і точний розрахунок екологічного впливу;

– чіткий, мінімалістичний інтерфейс з англомовним оформленням, що робить додаток доступним для широкої міжнародної аудиторії;

– зосередженість саме на екологічних метриках: карбоновий слід, водний слід, використання земельних ресурсів, евтрофікація та дефіцит ресурсів – без зайвих даних, які можуть відволікати;

– автоматичне обчислення впливу на основі ваги продукту дає змогу користувачу швидко оцінити реальний екологічний слід свого раціону.

Проаналізовані мобільні рішення демонструють значний потенціал у формуванні екологічної свідомості користувачів. Водночас, існує потреба в створенні більш простих, швидких і гнучких інструментів, орієнтованих на практичне застосування в повсякденному житті. Розроблений у межах цієї кваліфікаційної роботи додаток вирізняється саме цими властивостями, що забезпечує йому конкурентні переваги та перспективи для подальшого розвитку.

1.3 Вибір методології та технологічної платформи

У процесі розробки мобільного додатку для оцінки екологічного впливу продуктів було здійснено обґрунтований вибір методології розробки та технологічного стеку, що відповідають поставленим завданням і забезпечують ефективність реалізації проекту.

Розробка виконувалася із застосуванням принципів гнучкої методології (Agile) [14], яка передбачає поетапне виконання завдань, що дозволяє гнучко реагувати на зміни вимог і своєчасно інтегрувати функціональні компоненти додатку. Використання Agile [14] сприяло оптимізації процесу розробки,

полегшило тестування проміжних результатів і прискорило впровадження продукту.

Основним підходом до оцінки екологічного впливу став розрахунок за формулами, рекомендованими в наукових джерелах і стандартах Life Cycle Assessment (LCA) [26]. Зокрема, для кожного продукту екологічні метрики (карбоновий слід, водний слід, використання земельних ресурсів, евтрофікація та дефіцит ресурсів) були помножені на вагу введеного продукту у кілограмах. Такий підхід дозволив отримати коректну оцінку впливу відповідно до реального споживання.

Для створення кросплатформного мобільного додатку було обрано React Native з використанням середовища Expo [13]. Це рішення забезпечило можливість швидкої розробки і тестування додатку на різних пристроях під керуванням Android. React Native [19] надає зручні інструменти для побудови інтерфейсу користувача, а Expo спрощує розгортання і оновлення додатку, що є важливим для подальшого масштабування проекту.

Серверна частина реалізована з використанням Django [12] (версія 5.2.1) та Django REST Framework [11] (версія 3.16.0), що дозволило ефективно організувати REST API для обробки даних. Вибір Django обґрунтований його високою стабільністю, потужними засобами для швидкої розробки, а також великою кількістю вбудованих функцій, що дозволяють мінімізувати використання сторонніх пакетів, що використовують основні методології для чистого коду та зрозумілого коду, як зазначено у книгах «Clean Architecture: A Craftsman's Guide to Software Structure and Design» [33] та «Design Patterns: Elements of Reusable Object-Oriented Software» [34]. Серед основних встановлених бібліотек варто відзначити:

– django-cors-headers [27] для організації безпечного обміну даними між сервером і клієнтом;

– django-filter [25] для фільтрації запитів API;

– djangorestframework_simplejwt [29] для реалізації аутентифікації та безпеки;

– python-dotenv [17] для зберігання конфіденційних параметрів оточення.

Застосування цього технологічного стеку забезпечило швидкий цикл розробки, надійність та масштабованість сервісу.

Основними критеріями при виборі технологій були:

– продуктивність: обрані платформи дозволяють оперативно обробляти запити та працювати з великим обсягом даних.

– масштабованість: можливість подальшого розширення функціоналу та підтримка нових платформ.

– зручність розробки: інтуїтивний синтаксис React Native та широкі можливості Django для швидкої реалізації серверної логіки.

– підтримка кросплатформенності: React Native [1] дає змогу випускати додаток одночасно для Android та iOS (у перспективі).

– безпека: використання JWT-токенів [29] і CORS [27] захищає обмін даними між клієнтом і сервером.

– відповідність вимогам LCA [26]: реалізація обчислень екологічного впливу згідно з рекомендованими методиками.

Таким чином, вибір методології розробки та технологічної платформи відповідає сучасним вимогам і забезпечує якісну реалізацію завдань, поставлених у рамках кваліфікаційної роботи. Для розробки додатку використовуються загальні принципі проєктування [31]. Також, не зважаючи на те що фреймворком є django [12], книга [32] описує загальні принципи розробки веб-додатків, які використовуються не залежно від фреймворку.

2 ПРОЄКТУВАННЯ ОСНОВНИХ СКЛАДОВИХ ДОДАТКУ

2.1 Розробка архітектури мобільного додатку

Процес розробки нашого мобільного застосунку розпочинається з чіткого формулювання мети. Основною метою даного застосунку є надання користувачу інструменту для оцінювання екологічного впливу спожитих продуктів харчування. Це передбачає швидкий та інтуїтивно зрозумілий спосіб дізнатися, який вплив на довкілля має конкретний продукт, з урахуванням його ваги.

На основі поставленої мети формується базовий функціонал, який включає:

– реєстрацію та автентифікацію користувачів;

– введення продукту (назви та ваги);

– автоматичний розрахунок екологічних показників;

– перегляд результатів у зручному вигляді.

Також на цьому етапі зроблено попередній аналіз цільової аудиторії людей, які цікавляться сталим споживанням або хочуть зробити свій раціон більш екологічно відповідальним та вирішили спростити додаток для користувачів.

Після визначення функціональності обираємо архітектуру системи. У цьому проєкті реалізовано клієнт-серверну архітектуру, яка є найдоцільнішою у випадках, коли обчислення та зберігання даних виконується на сервері:

– Мобільний клієнт на базі React Native[2] – кросплатформений інтерфейс для Android та iOS, який забезпечує зручну та швидку взаємодію з користувачем.

– Серверна частина на Django[3] + Django REST Framework[4] – обробляє запити, виконує розрахунки та працює з базою даних, також має зручний інструмент ORM[22](Object Relational Mapping), який значно спрощує взаємодію з базою даних.

– База даних SQLite[15] – обрана через простоту, швидкість розгортання та відсутність потреби у масштабуванні, оскільки дані зберігаються локально та не передбачено велику кількість одночасних користувачів. Також це база даних, яку використовує Django за замовчуванням.

Комунікація між клієнтом і сервером відбувається через REST API з використанням формату JSON. Для захисту доступу реалізовано авторизацію за допомогою JWT-токенів[29], у випадку якщо програма розширюватиметься і виникне потреба у публічному API, то ми можемо реалізувати доступ по токену задля оптимізованого використовування ресурсів сервера.

Архітектура мобільного додатку передбачає поділ системи на дві основні частини: клієнтську (фронтенд) та серверну (бекенд). Такий підхід забезпечує чітке розмежування відповідальностей і дозволяє гнучко масштабувати та модифікувати систему.

Клієнтська частина (рисунок 2.1) реалізована за допомогою фреймворку React Native[5] із використанням Expo[13], що дає змогу створити кросплатформений застосунок, який працює як на Android, так і на iOS.

Рисунок 2.1 – Клієнтська частина

Вибір React Native[5] обумовлений простотою розробки, можливістю швидкого тестування і деплою, а також широкою підтримкою спільноти. У мобільному додатку передбачено інтуїтивно зрозумілий інтерфейс, розроблений англійською мовою, що полегшує взаємодію користувача з системою з першого запуску.

Навігація в додатку реалізована за допомогою базових засобів React Native (Screen Navigation)[20]. Такий підхід забезпечує послідовний перехід між основними екранами: пошук продуктів, відображення результатів, сторінка профілю користувача та інформаційна сторінка з описом впливу продуктів на довкілля.

Валідація даних на клієнті здійснюється при введенні назви продукту: система перевіряє наявність продукту у локальному списку та відповідно відображає результати. Однак основна логіка обробки даних, включно з обчисленням екологічного впливу, реалізована на сервері для забезпечення достовірності і захисту бізнес-логіки.

Серверна частина (рисунок 2.2) побудована на основі фреймворку Django з використанням Django REST Framework (DRF)[10] для реалізації API.

Рисунок 2.2 – Серверна частина

Бекенд забезпечує автентифікацію користувачів за допомогою JWT[29] (JSON Web Tokens), що гарантує безпечний обмін даними між клієнтом і сервером. Валідація введених користувачем даних, пошук найбільш відповідного продукту за допомогою алгоритму нечіткого пошуку реалізованого

за допомогою django-filters[25], а також обчислення екологічних метрик здійснюються на сервері.

Для структурування проекту бекенд розділений на два основні модулі: accounts, який відповідає за управління користувачами та аутентифікацію, та storage_api, що містить основну бізнес-логіку для обробки даних про продукти і їхній екологічний вплив.

Обмін даними між фронтендом і бекендом організований через REST API, що використовує стандартизовані HTTP-запити. Цей підхід спрощує інтеграцію та полегшує подальший розвиток системи.

Відсутність складних менеджерів стану чи кешування на клієнтській частині була свідомим вибором для збереження простоти архітектури та забезпечення стабільної роботи додатку на різних пристроях.

Таким чином, архітектура мобільного додатку базується на чіткій поділі функціональності між клієнтом і сервером, що забезпечує ефективну взаємодію користувача із системою та надійність обробки даних.

2.2 Проєктування бази даних

У розробленому застосунку використовується база даних реляційного типу SQLite[15], що дає змогу ефективно зберігати структуровану інформацію при невеликому обсязі даних. Вибір цієї СУБД обумовлений простотою розгортання, відсутністю потреби в окремому сервері, а також високою сумісністю з Django[12].

База даних була спроєктована з урахуванням розділення відповідальностей між компонентами системи: з одного боку – екологічні показники продуктів харчування, з іншого – запити користувачів і результати обчислень.

Ключові особливості реалізації:

– Зв’язок між таблицями: Product має зовнішній ключ на User, що дозволяє зберігати історію запитів для кожного користувача.

– Зберігання розрахованих метрик: Product містить обчислені значення на основі базових коефіцієнтів із ProductImpact.

– Окрема таблиця для довідкових даних: ProductImpact не змінюється під час використання, а лише читається при обробці запиту.

– Використання фреймворку Django [6] ORM [22]: моделі описані як Python-класи, що спрощує роботу з ними на рівні API.

Основними сутностями, що лежать в основі структури БД, є:

1. Користувач (User). Ця сутність відповідає стандартній моделі Django User (рисунок 2.3) та містить інформацію про зареєстрованих користувачів. Застосовується для реалізації аутентифікації й авторизації. Кожен запит у системі пов’язаний з певним користувачем через зовнішній ключ.

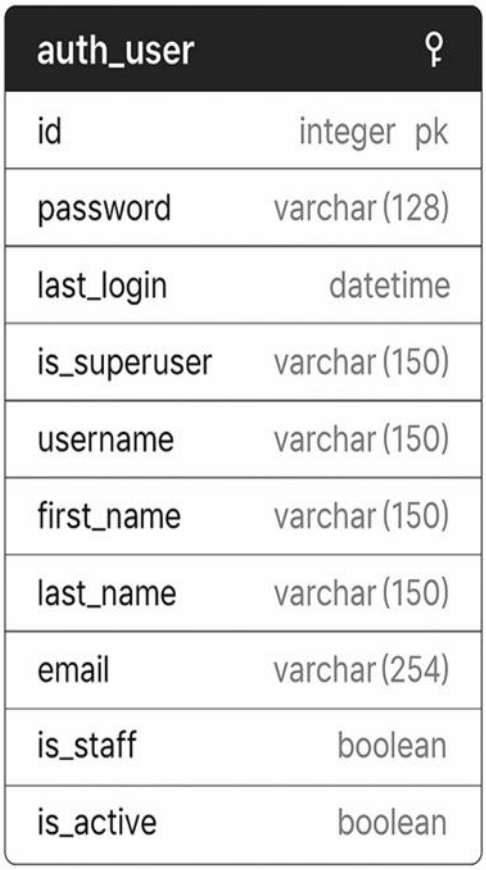

Рисунок 2.3 – Стандартна модель користувача джанго

2. Вплив продукту (ProductImpact). Ця таблиця містить попередньо зібрані та оброблені екологічні метрики для широкого переліку продуктів. Вона є довідниковою, і дані в ній не змінюються під час використання системи. Для кожного продукту зберігаються наступні поля:

– name – назва продукту (унікальне поле);

– carbon – вуглецевий слід (грамів CO_2 еквіваленту на кг);

– eutrophication – евтрофікація (мг фосфору на кг);

– water_use – використання води (літрів на кг);

– land_use – використання землі (м² на кг);

– scarcity – ресурсна обмеженість (індекс одиниць на кг).

3. Запит користувача (Product). Ця таблиця фіксує кожен окремий запит користувача. При створенні запиту здійснюється нечіткий пошук по таблиці ProductImpact, обчислюються всі метрики впливу, які потім зберігаються разом з введеними користувачем даними:

– name – введена назва продукту;

– weight – вага продукту у кілограмах;

– carbon, eutrophication, water_use, land_use, scarcity – результат обчислення метрик на введену вагу;

– user_id – зовнішній ключ на користувача.

Таким чином, реалізовано зв’язок "один до багатьох" (1:N) [8] між моделями User і Product: один користувач може створити багато запитів.

Базу даних спроєктовано з дотриманням третьої нормальної форми (3NF)[8], оскільки:

– зберігання розрахованих метрик: Product містить обчислені значення на основі базових коефіцієнтів із ProductImpact;

– всі атрибути залежать лише від первинного ключа;

– не має транзитивних залежностей;

– дублювання даних зведено до мінімуму.

Обґрунтування вибору схеми. Обрана модель забезпечує розділення постійних довідкових даних (ProductImpact) від динамічних запитів користувача (Product). Це дозволяє з одного боку – зберігати чисту і стабільну таблицю з джерелом екологічної інформації, з іншого – накопичувати історію використання для подальшого аналізу або статистики. Повну діаграму бази данних можна побачити на (рисунок 2.4).

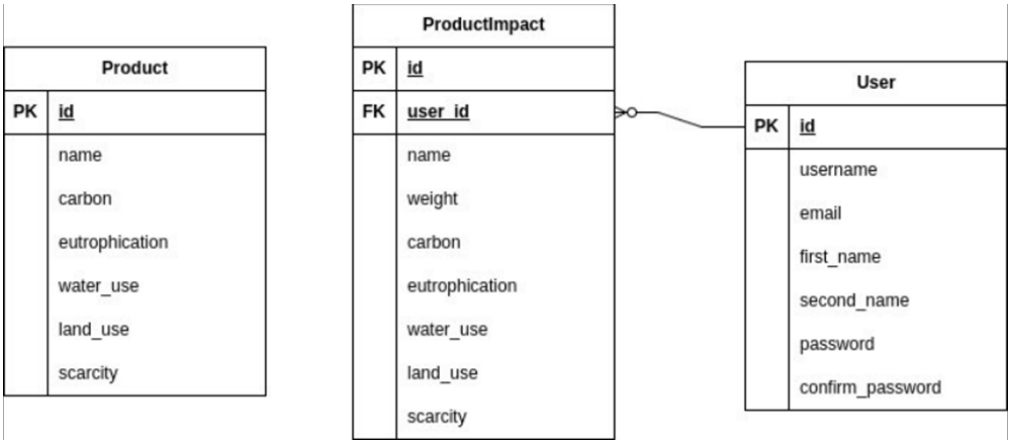

Рисунок 2.4 – Діаграма таблиць додатку

2.3 Розробка інтерфейсу користувача

Інтерфейс користувача буде реалізовано за допомогою React Native [2] – фреймворку, який дозволяє створювати кросплатформенні мобільні застосунки для Android та iOS, використовуючи єдину кодову базу. Такий підхід значно пришвидшує розробку та спрощує підтримку проєкту, оскільки не вимагає дублювання логіки інтерфейсу для кожної з платформ окремо.

Основними принципами побудови інтерфейсу будуть:

– Простота та інтуїтивність – користувач має змогу одразу розпочати взаємодію без потреби у додаткових інструкціях.

– Фокус на функціональність – головна мета застосунку – інформування про екологічний вплив продуктів, тому акценти зроблено саме на цьому.

– Зручність навігації – використано стандартну навігаційну архітектуру з переходами між екранами через Stack Navigator. Інтерфейс поділений на кілька основних екранів.

Основний екран (рисунок 2.5), з якого починається робота користувача. Тут можна ввести назву продукту та його вагу в кілограмах. Після введення даних застосунок автоматично надсилає запит до серверної частини, отримує всі доступні результати та відображає екологічні метрики:

– Фокус на функціональність – головна мета застосунку – інформування про екологічний вплив продуктів, тому акценти зроблено саме на цьому. «Carbon footprint», «Water use», «Land use», «Eutrophication», «Resource scarcity».

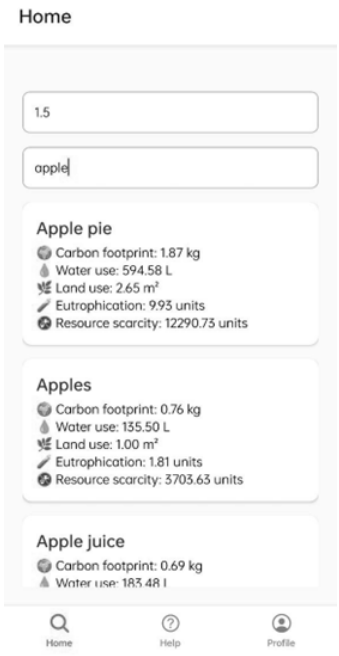

Рисунок 2.5 – Екран результатів пошуку

– Login/Register (рисунок 2.6): Сторінки автентифікації реалізовано з використанням JWT[29]. Після успішного входу або реєстрації, токен зберігається локально для подальших авторизованих запитів до API.

Рисунок 2.6 – Сторінки логіну та реєстрації

– Help (рисунок 2.7): Довідковий екран, який містить опис кожної з мрик впливу. Тут користувач може дізнатися, що означає, наприклад, показник eutrophication або scarcity, і чому це важливо для оцінки екологічного сліду.

Рисунок 2.7 – Довідкова сторінка

– Profile (рисунок 2.8): Екран профілю користувача буде важливим елементом інтерфейсу, що надасть повну інформацію про взаємодію користувача з системою. На цьому екрані відображатимиться загальна кількість здійснених запитів, кількість активних сесій, а також загальний час, який користувач првів у додатку. Це дозволяє сформувати уявлення про рівень активності користувача та його залучення.

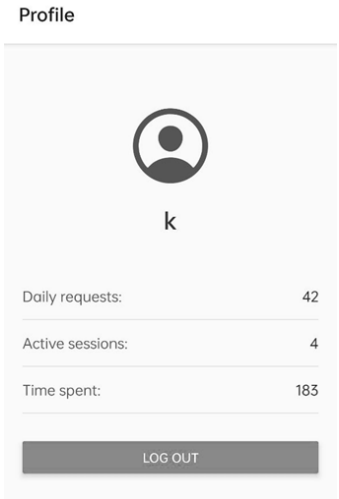

Рисунок 2.8 – Профіль користувача

Додаткові елементи UX:

– Валідація введених даних: у застосунку реалізовано базову валідацію на клієнтській стороні – якщо користувач вводить порожнє поле ваги, то сервер обчислює метрики використовуючи базову вагу вибраного продукту, що становить один кг. по замовчуванню . У випадку якщо продукт та вага не вказані, буде виведено «No product found».

– Обробка помилок: якщо продукт не знайдено на сервері або нечіткий пошук не дав задовільного результату, застосунок повідомляє користувача про це та пропонує ввести іншу назву.

– Мультимовність: незважаючи на те, що застосунок реалізовано англійською мовою, структура інтерфейсу дозволяє легко реалізувати підтримку локалізації у майбутньому.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ МОБІЛЬНОГО ДОДАТКУ

3.1 Розробка серверної та клієнтської частини додатку

Мобільний додаток реалізовано за клієнт-серверною архітектурою, де клієнтська частина написана з використанням React Native, а серверна – на Django [12] із застосуванням Django REST Framework (DRF) [11]. Це дозволило забезпечити ефективну взаємодію між інтерфейсом користувача та базою даних.

На головному екрані (рисунок 3.1) користувач вводить назву продукту та його вагу у кілограмах. Після цього здійснюється запит на сервер та бачить результат одразу на екрані – кожна метрика супроводжується іконкою та одиницями виміру для зручності.

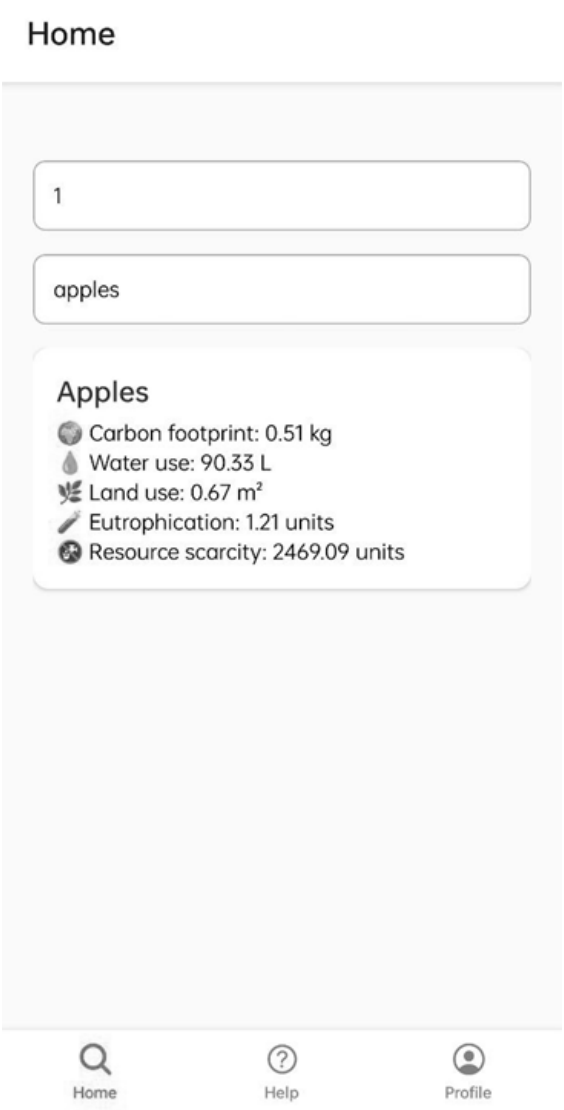

Рисунок 3.1 – Відображення екологічних метрик після обробки запиту

Гнучке фільтрування. Для обробки запитів і реалізації гнучкого фільтрування результатів використовується вбудована реалізацію пошуку за допомогою класу rest_framework.filters.SearchFilter [25], що дозволяє фільтрувати запити за назвою продукту. Наприклад можна отримати інформацію не тільки про кілограм яблук «Apples», а й пов’язані з ним випічки, соки тощо (рисунок 3.2).

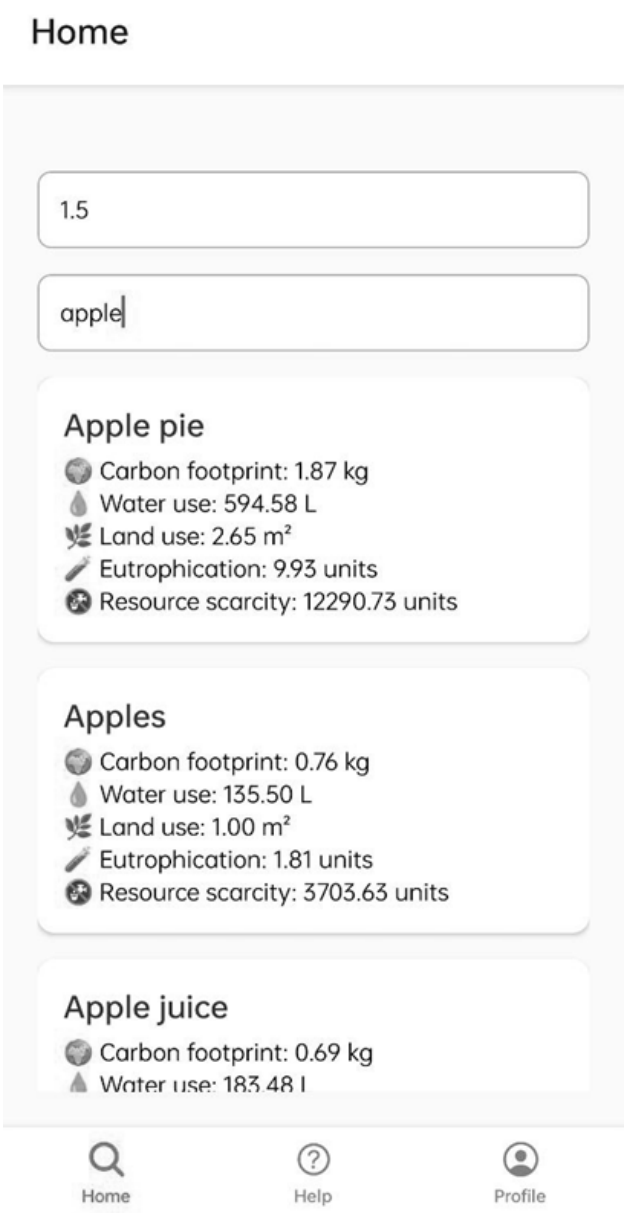

Рисунок 3.2 – Пов’язані з запитом продукти

Відсутність даних. У випадку, якщо продукт не знайдено, необхідні данні не введено або збіг занадто неточний, виводиться відповідне повідомлення (рисунок 3.3).

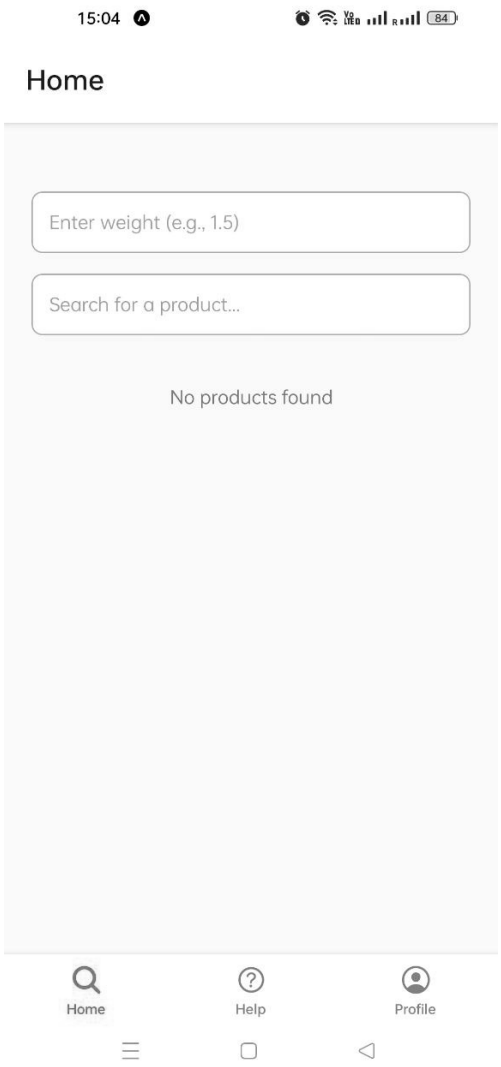

Рисунок 3.3 – Результат без даних

Додатковий функціонал:

– Механізм аутентифікації користувачів у додатку реалізований з використанням технології JWT (JSON Web Token) [29], що дозволяє забезпечити безпечну та масштабовану систему авторизації без необхідності зберігати сесії на сервері. Для реалізації даного функціоналу було створено окремі екрани для реєстрації та входу користувача (рисунок 3.4), які забезпечують зручний інтерфейс взаємодії з системою. Процес реєстрації був розроблений самостійно та передбачає створення нового користувача за допомогою відповідного API-ендпоінту /api/register/. Реєстраційна логіка винесена в окремий серіалізатор, який відповідає за валідацію введених користувачем даних і подальше створення облікового запису в системі.

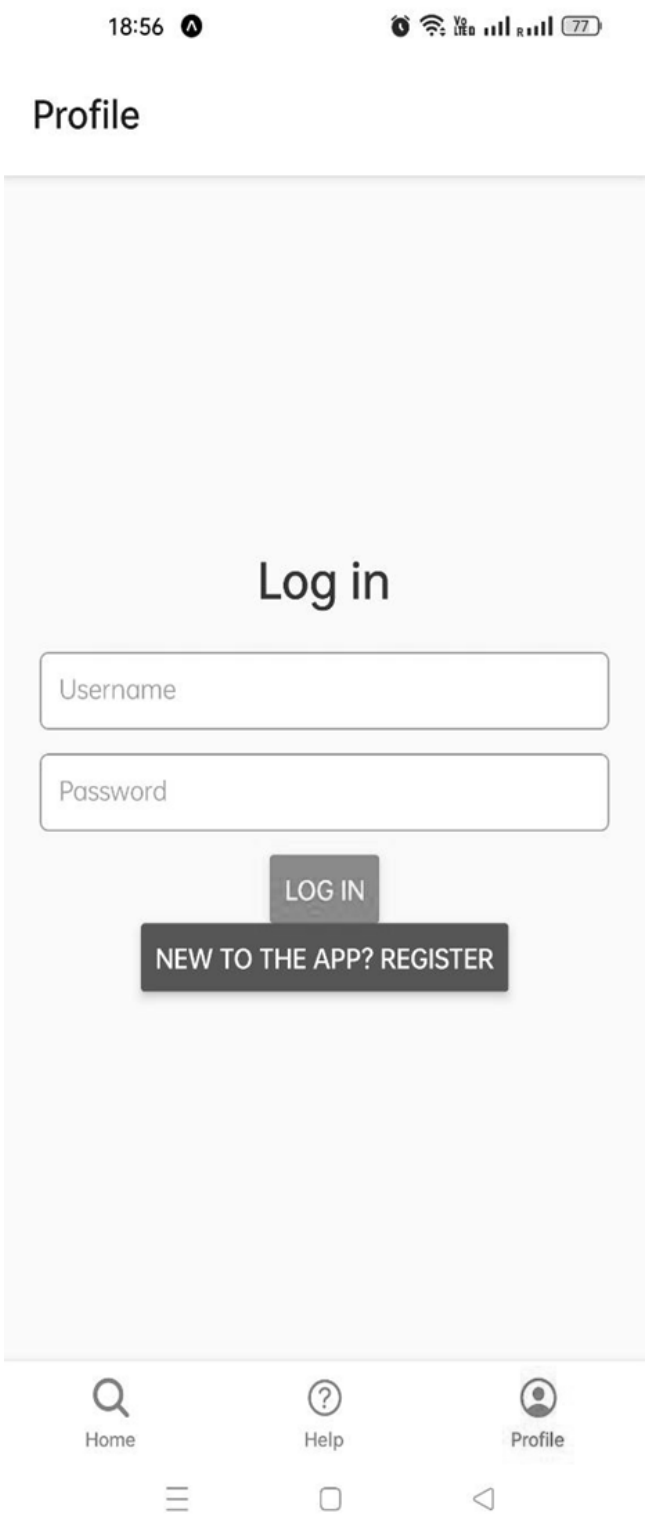

Рисунок 3.4 – Реєстрація та вхід

– Help (рисунок 3.5). Сторінка довідки (Help) реалізована як окремий екран із текстовими поясненнями екологічних метрик. Вона допомагає користувачам зрозуміти значення показників і полегшує користування додатком. Інформація відображається у зручному форматі та може оновлюватись без змін у коді.

Рисунок 3.5 – Пояснення значень метрик

– Profile (рисунок 3.6). Було реалізовано відповідно до попередньо спроєктованої логіки. Для отримання інформації про взаємодію користувача із системою (кількість запитів, активні сесії, загальний час використання) були створені відповідні API-ендпоінти на сервері. Дані обчислюються на основі логів активності та метрик, що зберігаються у базі даних. Для збереження цих показників використовувалися окремі моделі, які фіксують ключові події: запити, початок і завершення сесії, тривалість перебування у додатку.

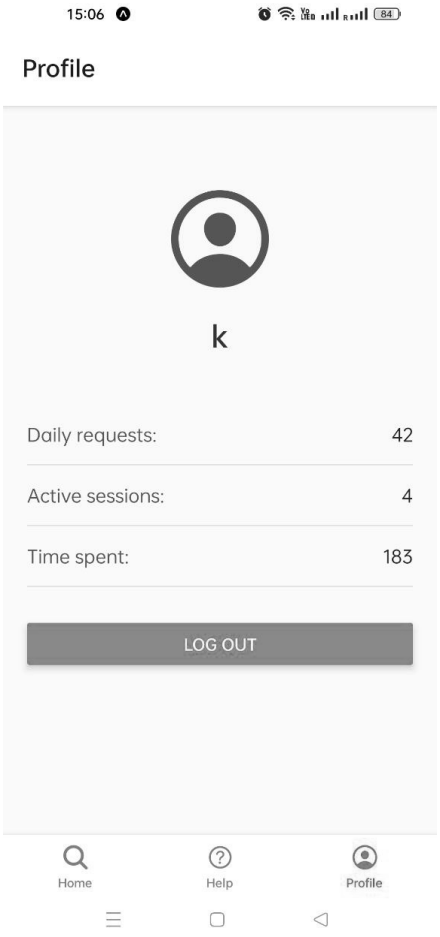

Рисунок 3.6 – Статистика запитів користувача

3.2 Інтеграція системи оцінювання екологічного впливу

Першим етапом інтеграції стало створення необхідної інфраструктури для збереження довідкових даних про екологічний вплив. Для цього було реалізовано:

– Серіалізатор ProductImpactSerializer, який описує формат прийому та збереження даних у відповідну модель (рисунок 3.7);

```
75
76
77 class ProductImpactSerializer(serializers.ModelSerializer):
78     class Meta:
79         model = ProductImpact
80         fields = "__all__"
```

Рисунок 3.7 – Серіалайзер для прийому даних

– View[7]-клас, що обробляє POST та GET запити та зберігає отриману інформацію у базу даних (рисунок 3.8);

```
28
29 class ProductImpactView(ListCreateAPIView):
30     queryset = ProductImpact.objects.all()
31     serializer_class = ProductImpactSerializer
32     filter_backends = [SearchFilter]
33     search_fields = ["name"]
```

Рисунок 3.8 – Клас, що відповідає за запис даних у БД

– URL-шлях, який вказує маршрут, на який будем робити запити для отримання і обрахування метрик (рисунок 3.9).

```
path("product_impacts/", ProductImpactView.as_view()),
```

Рисунок 3.9 – Обробка даних

Для забезпечення роботи системи оцінювання екологічного впливу були використані відкриті дані з ресурсу Our World Data [24].

Відповідні таблиці були завантажені у форматі CSV, після чого за допомогою спеціального Python-скрипта конвертовані у формат JSON та автоматично передані на локальне API сервера (рисунки 3.10 - 3.11).

Для того, щоб зробити код зрозумілим і максимально простим, було використано поради з книги «Прагматичний програміст» [30].

```python
import json
import os      "os" is not accessed
import requests

dir_name = "datasets"
url = "http://127.0.0.1:8000/product_impacts/"

def load_impact_data(filepath):
    with open(filepath, "r", encoding="utf-8") as f:
        raw = json.load(f)
    data = {}
    for item in raw:
        entity = item.get("Entity") or item.get("entity")
        impact = item.get("Impact") or item.get("impact")

        if entity is None or impact is None:
            continue

        try:
            data[entity] = float(impact)
        except (ValueError, TypeError):
            print(f"Skipping invalid impact for entity: {entity}")
    return data

carbon_dict = load_impact_data(f"{dir_name}/carbon_dataset.json")
eutrophication_dict = load_impact_data(f"{dir_name}/eutrophication_dataset.json")
water_use_dict = load_impact_data(f"{dir_name}/freshwater_dataset.json")
land_use_dict = load_impact_data(f"{dir_name}/landuse_dataset.json")
scarcity_dict = load_impact_data(f"{dir_name}/waterscarce_dataset.json")

```

Рисунок 3.10 – Обробка отриманих файлів

```python
all_entities = set().union(
    carbon_dict,
    eutrophication_dict,
    water_use_dict,
    land_use_dict,
    scarcity_dict,
)

for entity in all_entities:
    payload = {
        "name": entity,
        "carbon": carbon_dict.get(entity, 0.0),
        "eutrophication": eutrophication_dict.get(entity, 0.0),
        "water_use": water_use_dict.get(entity, 0.0),
        "land_use": land_use_dict.get(entity, 0.0),
        "scarcity": scarcity_dict.get(entity, 0.0),
    }

    try:
        response = requests.post(url, json=payload)
        if response.status_code == 201:
            print(f"✓ Created: {entity}")
        else:
            print(f"× Failed ({response.status_code}): {entity} - {response.text}")
    except requests.exceptions.RequestException as e:
        print(f"Error posting {entity}: {e}")
```

Рисунок 3.11 – Вставка API

Ці дані завантажуються у модель ProductImpact, яка зберігає базові коефіцієнти екологічного впливу (викиди CO_2 , використання води, землі, евтрофікація та дефіцит ресурсів) для кожного продукту.

Клієнтська частина мобільного застосунку надсилає POST-запит з назвою продукту та його вагою на відповідний URL (рисунок 3.12).

```
export default function HomeScreen() {
  const [query, setQuery] = useState('');
  const [weight, setWeight] = useState('1');
  const [data, setData] = useState([]);
  const [loading, setLoading] = useState(false);

  const fetchData = async (searchTerm) => {
    setLoading(true);
    try {
      const response = await fetch(
        `http://192.168.0.106:8000/product_impacts/?search=${encodeURIComponent(searchTerm)}`
      );

      if (!response.ok) throw new Error(`HTTP error! status: ${response.status}`);
      const json = await response.json();
      if (!Array.isArray(json.results)) throw new Error('Invalid response format');

      setData(json.results);
    } catch (e) {
      console.error('Fetch error:', e);
    } finally {
      setLoading(false);
    }
  };
```

Рисунок 3.12 – Запит клієнта

На сервері отримані дані обробляються в ProductSerializer, відбувається валідація даних на часткову схожість, після чого перевіряється, чи всі данні присутні, для того, щоб не виникала помилка. Після перевірки даних на присутність перевіряються дані, введені безпосередньо в API, на коректність, що дозволить у майбутньому запобігти проблемам у БД (рисунок 3.13). Також перевіряється, чи користувач зареєстрований у додатку, а якщо ні, надсилають запити адмінові. Розрахунок кінцевих значень здійснюється множенням коефіцієнтів на масу введеного продукту (рисунок 3.14). Додатково перевіряються дані, введені безпосередньо в API, на коректність, що дозволить у майбутньому запобігти проблемам у БД та надсилати повідомлення про сутність помилки.

```
def validate(self, attrs):
    if any(i.isdigit() for i in attrs["name"]):
        raise serializers.ValidationError("The name should not contain digits")
    if not isinstance(attrs["weight"], (int, float)):
        raise serializers.ValidationError(
            "The field should only contain integer or float numbers"
        )

    return attrs
```

Рисунок 3.13. Валідація данних на тип помилок

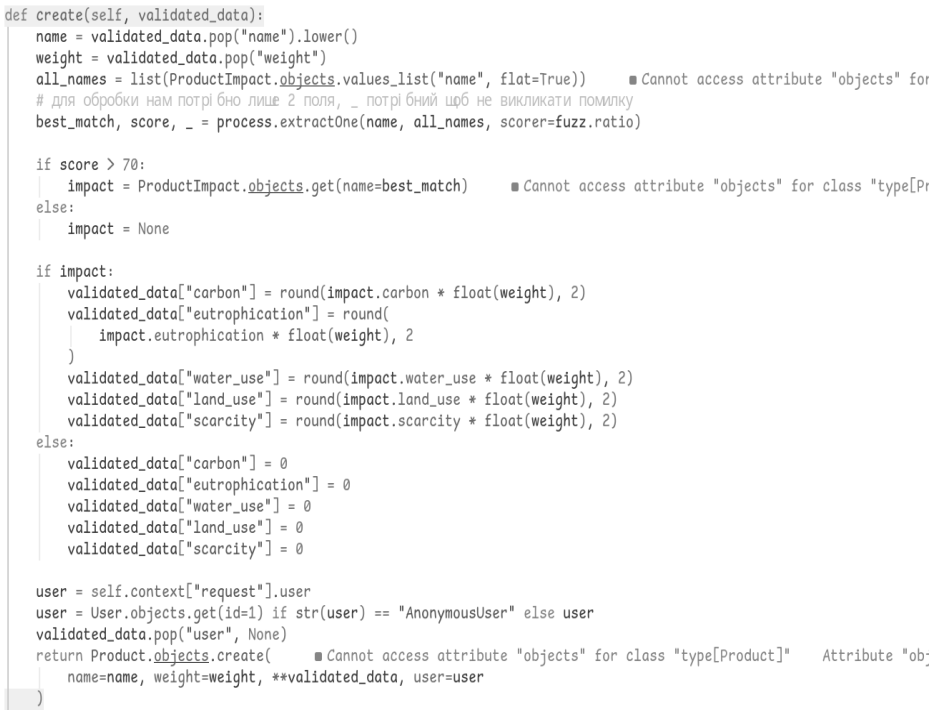

Рисунок 3.14 – Обробка даних

Усі ці розрахунки записуються в окрему таблицю Product, пов’язану з користувачем, щоб зберігати історію запитів.

Для успішної роботи REST API з мобільним додатком були додані відповідні налаштування у CORS_HEADERS[27], які дозволяють клієнту взаємодіяти з сервером.

На клієнті React Native[5] отримані результати з API обробляються та відображаються у вигляді таблиці значень екологічного впливу.

3.3 Тестування та впровадження мобільного додатку

Після реалізації всіх основних функцій мобільного застосунку було проведено його повноцінне тестування на реальному пристрої з операційною системою Android. Для цього:

– було встановлено Android Studio [28] для емуляції та запуску додатка локально;

– згенеровано QR-код через Expo Go[13], який дозволив протестувати застосунок без ручної збірки APK (рисунок 3.15);

– локальний сервер було під’єднано до мобільного пристрою за допомогою IP-адреси Wi-Fi мережі (як у нашому випадку, 192.168.0.106:8000), що дозволило мобільному клієнту надсилати запити на локальний Django-сервер.

У зв’язку з цим у налаштуваннях Django (файл settings.py) було додано IP-адресу локальної мережі до параметра, щоб клієнтська частина вільно могла надсилати запити на серверну частину:

```
ALLOWED_HOSTS = ['192.168.0.106', 'localhost', '127.0.0.1']
CORS_ALLOWED_ORIGINS = [
   'http://192.168.0.106:8081',
]
```

Рисунок 3.15 – QR-код для Expo go

Перевірка функціоналу:

– Аутентифікація: була перевірена робота форми входу та реєстрації. Зокрема, при спробі входу з неправильними даними система коректно повертала повідомлення про помилку (рисунок 3.16).

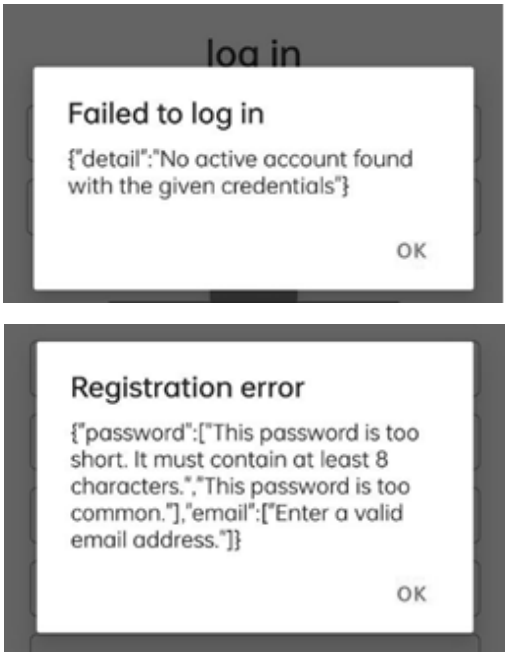

Рисунок 3.16 – Невдалий вхід і невдала реєстрація

– Реєстрація нового користувача: після створення облікового запису надається повний доступ до функцій додатку (рисунок 3.17).

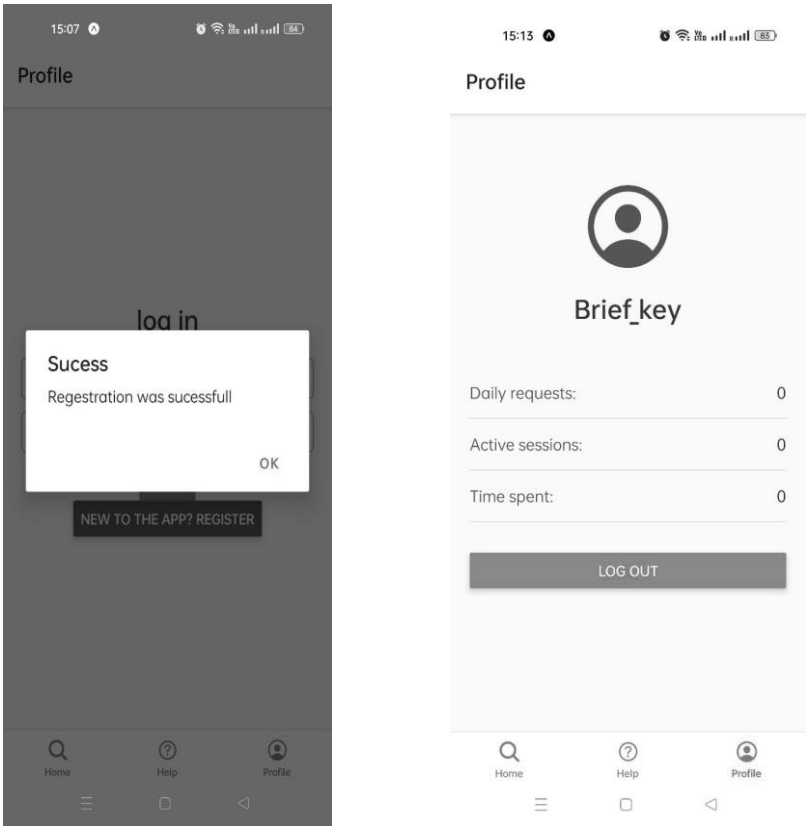

Рисунок 3.17 – Успішна реєстрація і подробиці нового акаунта

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було проведено:

1. Детальний аналіз сучасних мобільних додатків у сфері екології та екологічного впливу харчових продуктів. Виявлено, що більшість існуючих рішень мають складні інтерфейси або обмежений функціонал, що ускладнює доступ пересічного користувача до корисної інформації. На відміну від них, створений додаток орієнтований на простоту та зрозумілість з першого запуску, що підвищує його привабливість для широкої аудиторії. Застосунок має англомовний інтерфейс, що розширює його потенційну користувацьку базу на міжнародному рівні.

2. Технічна реалізація додатку базується на стеку React Native для клієнтської частини та Django з Django REST Framework для бекенду. Вибір цих технологій обумовлений їх надійністю, широкими можливостями та великою підтримкою спільноти. React Native дозволяє швидко розробляти кросплатформні мобільні додатки для iOS та Android, що важливо для доступності програми на різних пристроях. Django REST Framework забезпечує ефективне створення API з підтримкою сучасних методів аутентифікації – у нашому випадку, JWT – та зручною інтеграцією з базою даних.

3. Особливістю системи є використання даних про екологічний вплив продуктів, взятих із відкритого джерела Our World in Data. Дані було оброблено та приведено до формату JSON, після чого вони були завантажені на локальне API за допомогою скрипта. Це забезпечило актуальність інформації в додатку та можливість її подальшого оновлення. На сервері застосовано нечітке порівняння назв продуктів (RapidFuzz), що дозволяє враховувати помилки та варіації у введених користувачем даних, підвищуючи точність результатів.

4. База даних побудована на SQLite, що є зручним для локального розгортання й тестування. Вона містить основні сутності: користувачів, їхні запити з інформацією про продукт і вагу, а також довідкові дані про екологічний вплив різних продуктів. Використання зовнішніх модулів, таких як django-filter,

було відхилено через простоту проєкту та статичність даних. Аутентифікація реалізована через JWT, що забезпечує безпеку та зручність користування.

5. Розробка інтерфейсу користувача у React Native включала створення основних екранів: введення продукту з вагою, сторінки допомоги з детальним описом усіх екологічних метрик, а також профілю користувача з історією запитів. Особливу увагу приділено простоті і швидкості взаємодії. На клієнті додатково реалізована перевірка валідності введених даних, а також відображення повідомлень про відсутність знайдених продуктів.

6. Тестування додатку проводилося шляхом встановлення та запуску на реальному пристрої Android. Для цього було налаштовано Android Studio, скановано QR-код, а також налаштовано локальний сервер із відповідними CORS-заголовками для коректної роботи запитів. Процес тестування виявив деякі проблеми з аутентифікацією, зокрема неправильний логін нових користувачів, які було успішно усунуто. Після виправлень додаток працює стабільно, забезпечуючи швидкий та коректний обмін даними між клієнтом і сервером.

Отже, реалізований мобільний додаток є дієвим інструментом для підвищення обізнаності користувачів щодо екологічних наслідків їх харчування. Він поєднує сучасні технології з перевіреними даними, має інтуїтивний інтерфейс та відповідає сучасним вимогам безпеки. Результати роботи можуть бути використані не лише в контексті індивідуального користування, а й для створення масштабних платформ, що підтримують екологічно свідоме споживання. Перспективи розвитку проекту включають додавання соціальних функцій, персоналізованих рекомендацій та інтеграцію з іншими сервісами здорового харчування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Banks F., Masiello E. React Native in Action. Manning Publications, 2019. 312 с.

2. Eberhard B. Fullstack React Native: Create beautiful mobile apps with JavaScript and React Native. Fullstack.io, 2019. 486 с.

3. Greenfeld D. R., Greenfeld A. R. Two Scoops of Django 3.x: Best Practices for the Django Web Framework. Two Scoops Press, 2020. 532 с.

4. Vincent W. S. Django for APIs: Build web APIs with Python & Django. William S. Vincent, 2020. 298 с.

5. Academind. React Native tutorial for beginners [Електронний ресурс]. – Режим доступу: https://academind.com/tutorials/react-native-tutorial/ (дата звернення: 01.06.2025).

6. Awesome Django (добірка кращих ресурсів по Django) [Електронний ресурс]. – GitHub. – Режим доступу: https://github.com/wsvincent/awesome-django (дата звернення: 01.06.2025).

7. Django. View класи, що відповідають за фільтрування та збереження даних [Електронний ресурс]. – Режим доступу: https://docs.djangoproject.com /en/5.2/topics/http/views/ (дата звернення: 01.06.2025).

8. Database Normalization - Normal forms 1nf 2nf 3nf Table Examples [Електронний ресурс]. – FreeCodeCamp. – Режим доступу: https://www.freecodecamp.org/news/database-normalization-1nf-2nf-3nf-table-examples/ (дата звернення: 01.06.2025).

9.Додаток Oroeco [Електронний ресурс]. – Режим доступу: https://www.oroeco.org/ (дата звернення: 01.06.2025).

10. Django REST Framework на GitHub [Електронний ресурс]. – Режим доступу: https://github.com/encode/django-rest-framework (дата звернення: 01.06.2025).

11. Django REST Framework. Офіційна документація [Електронний ресурс]. – Режим доступу: https://www.django-rest-framework.org/ (дата звернення: 01.06.2025).

12. Django. Офіційна документація [Електронний ресурс]. – Режим доступу: https://docs.djangoproject.com/ (дата звернення: 01.06.2025).

13. Expo (фреймворк для React Native) [Електронний ресурс]. – Режим доступу: https://docs.expo.dev/ (дата звернення: 01.06.2025).

14. Методологія Agile [Електронний ресурс]. – Режим доступу: https://www.atlassian.com/agile (дата звернення: 01.06.2025).

15. База даних SQLite [Електронний ресурс]. – Режим доступу: https://sqlite.org/docs.html (дата звернення: 01.06.2025).

16. Додаток Open Food Facts [Електронний ресурс]. – Режим доступу: https://world.openfoodfacts.org/ (дата звернення: 01.06.2025).

17. Python-dotenv. Офіційна документація [Електронний ресурс]. – Режим доступу: https://saurabh-kumar.com/python-dotenv/ (дата звернення: 01.06.2025).

18. Додаток Yuka [Електронний ресурс]. – Режим доступу: https://yuka.io/en/ (дата звернення: 01.06.2025).

19. React Native. Офіційна документація [Електронний ресурс]. – Режим доступу: https://reactnative.dev/docs/getting-started (дата звернення: 01.06.2025).

20. React Navigation. Документація [Електронний ресурс]. – Режим доступу: https://reactnavigation.org/docs/getting-started (дата звернення: 01.06.2025).

21. Платформа Giki [Електронний ресурс]. – Режим доступу: https://giki.edu.pk/course/environmental-engineering/ (дата звернення: 01.06.2025).

22. Django ORM. Офіційна документація [Електронний ресурс]. – Режим доступу: https://docs.djangoproject.com/en/stable/topics/db/ (дата звернення: 01.06.2025).

23. Пазірович Ю. І. Мобільний додаток для оцінювання екологічного впливу продуктів [Електронний ресурс]. Міжнародна науково-практична інтернет-конференція. URL: https://www.economy-confer.com.ua/full-article/6290/ (дата звернення: 01.06.2025).

24. Environmental impacts of food production [Електронний ресурс] // Our World in Data. – Режим доступу: https://ourworldindata.org/environmental-impacts-of-food (дата звернення: 01.06.2025).

25. Django-filter. Офіційна документація [Електронний ресурс]. – Режим доступу: https://django-filter.readthedocs.io/en/stable/ (дата звернення: 01.06.2025).

26. Ecochain. Input data for Life Cycle Assessment (LCA) [Електронний ресурс]. – Режим доступу: https://ecochain.com/blog/input-data-for-lca/ (дата звернення: 01.06.2025).

27. Django-cors-headers. Офіційна документація [Електронний ресурс]. – Режим доступу: https://github.com/adamchainz/django-cors-headers (дата звернення: 01.06.2025).

28. Android Studio. Офіційна документація [Електронний ресурс]. – Режим доступу: https://developer.android.com/studio/intro (дата звернення: 01.06.2025).

29. Simple JWT. Офіційна документація [Електронний ресурс]. – Режим доступу: https://django-rest-framework-simplejwt.readthedocs.io/en/latest/ (дата звернення: 01.06.2025).

30. Hunt A. The Pragmatic Programmer: Your Journey to Mastery. 20th Anniversary Edition. Boston: Addison-Wesley Professional, 2019. 352 с.

31. Freeman E., Robson E. Head First Design Patterns: A Brain-Friendly Guide. 2nd ed. Sebastopol: O’Reilly Media, 2021. 694 с.

32. Grinberg M. Flask Web Development: Developing Web Applications with Python. 2nd ed. Sebastopol: O’Reilly Media, 2018. 258 с.

33. Martin R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Boston: Prentice Hall, 2017. 432 с.

34.Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Boston: Addison-Wesley Professional, 1994. 395 с.

35. Комар М. П., Саченко А. О., Васильків Н. М., Гладій Г. М., Коваль В. С., Ліп’яніна-Гончаренко Х. В. Методичні рекомендації до виконання кваліфікаційної роботи з освітньо-професійної програми «Комп’ютерні науки» спеціальності 122 «Комп’ютерні науки» за першим (бакалаврським) рівнем вищої освіти. Тернопіль: ЗУНУ, 2024. 57 с.

Додаток А

Копія публікації автора

Котляр Віталій Сергійович
МЕТОДИ ТА МОДЕЛІ ОПТИМІЗАЦІЇ РОЗКЛАДУ ЗАНЯТЬ У ЗАКЛАДАХ ВИЩОЇ ОСВІТИ ІЗ ЗАСТОСУВАННЯМ ШТУЧНОГО ІНТЕЛЕКТУ.....238

Мінєєв Сергій Павлович, Антончик Володимир Євгенійович, Ганкевич Валентин Феодосійович, Кіба В`ячеслав Якович
ШЛЯХИ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ БУРІННЯ СВЕРДЛОВИН У МІЦНИХ ГІРСЬКИХ ПОРОДАХ.....240

Пазірович Юрій Іванович
МОБІЛЬНИЙ ДОДАТОК ДЛЯ ОЦІНЮВАННЯ ЕКОЛОГІЧНОГО ВПЛИВУ ПРОДУКТІВ.....244

Саєнко Сергій Юрійович, Мсаллам Катерина Петрівна
ОСОБЛИВОСТІ РІДИН У СИСТЕМІ ОХОЛОДЖЕННЯ КАБЕЛІВ.....246

Військова справа

Губський Андрій Володимирович, Москальов Геннадій Юрійович
МЕТОДИ ТА ЗАХОДИ ЩОДО ЗБІЛЬШЕННЯ ЖИВУЧОСТІ ФОРТИФІКАЦІЙНИХ СПОРУД.....251

Губський Максим Володимирович, Москальов Геннадій Юрійович
ВЛАШТУВАННЯ СУЦІЛЬНОГО НАКРИТТЯ ТРАНШЕЙ ТА ЗАКРИТИХ ВОГНЕВИХ СПОРУД.....253

Гусак Оксана Борисівна, Івашкевич Євген Михайлович
ЗАСТОСУВАННЯ ХІМІЧНОЇ ЗБРОЇ В УМОВАХ ВІЙНИ В УКРАЇНІ. СУЧАСНИЙ ПОГЛЯД ТА АНАЛІЗ.....255

МОБІЛЬНИЙ ДОДАТОК ДЛЯ ОЦІНЮВАННЯ ЕКОЛОГІЧНОГО ВПЛИВУ ПРОДУКТІВ

Пазірович Юрій Іванович
студент, кафедра інформаційно-обчислювальних систем і управління, Західноукраїнський національний університет

Науковий керівник: Гладій Григорій Михайлович
кандидат економічних наук, доцент, Західноукраїнський національний університет

Інтернет-адреса публікації на сайті:
https://www.economy-confer.com.ua/full-article/6290/

Актуальність теми

Проблема кліматичних змін набула глобального масштабу. Викиди CO_2 є головним чинником підвищення середньої температури на планеті, що веде до катастрофічних наслідків: екстремальних погодних явищ, деградації екосистем, затоплення територій. Одним із ключових напрямів боротьби зі зміною клімату є підвищення екологічної свідомості громадян через використання цифрових інструментів оцінки індивідуального впливу.

Мета і завдання

Мета роботи – створити зручний мобільний додаток, який дозволяє користувачу оцінити кількість викидів CO_2, що виникають внаслідок його повсякденних дій, та отримати поради щодо зменшення екологічного сліду.

До основних завдань належали:

- Аналіз аналогічних рішень та виявлення їх обмежень;
- Проєктування архітектури клієнт-серверного додатку;
- Реалізація серверної частини на базі Django REST Framework;
- Створення мобільного застосунку за допомогою Kivy;

- Інтеграція з API Carbon Interface та власною базою Our World in Data;
- Тестування, збірка та розгортання APK-файлу на Android.

Наукова новизна та практична цінність

Розроблений застосунок об'єднує кілька джерел викидів – електрика, транспорт, перельоти, продукти – в одному інтерфейсі. Його особливістю є підтримка точного ручного введення параметрів користувачем, генерація персоналізованих порад та можливість подальшої інтеграції з іншими сервісами.

Практична цінність полягає у використанні застосунку як навчального засобу для підвищення екологічної обізнаності в школах, вишах або як інструменту внутрішньої просвіти в екологічно орієнтованих компаніях.

Використані технології

- **Frontend:** Python + Kivy (з використанням ScreenManager та адаптивного дизайну)
- **Backend:** Django 5.2 + Django REST Framework
- **База даних:** SQLite (з можливістю переходу на PostgreSQL)
- **Збірка мобільного додатку:** Buildozer (платформа Android)
- **API-сервіси:** Carbon Interface (для розрахунків викидів CO_2)
- **Інструменти:** dotenv, requests, unittest

Результати роботи

- Реалізовано повноцінний мобільний додаток з мінімалістичним інтерфейсом;
- Реалізовано інтеграцію із зовнішнім API для розрахунку викидів;
- Здійснено власну реалізацію підрахунків за категорією «продукти» на основі маси продукту та середнього коефіцієнта емісії;
- Додаток зібрано для Android, проведено базове модульне та функціональне тестування;
- Створено API для рекомендацій і бази аеропортів для перельотів.

Висновки

Розроблений застосунок демонструє, що за допомогою відкритих даних і сучасних технологій Python можна створити ефективний інструмент екопросвіти. Подальший розвиток можливий через:

- інтеграцію нових типів активностей;
- персоналізацію порад;
- розширення аналітики для збереження історії дій.