

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
«КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мультимедійних систем

Розробка застосунку для здорового харчування з системою рекомендацій на
базі Core ML: ефективність інтеграції локальних моделей у мобільні
HealthTech-рішення

Текстова частина до кваліфікаційної роботи
за спеціальністю 121 «Інженерія програмного забезпечення»

Керівник кваліфікаційної роботи
доцент Афонін А. О.

_____ (підпис)
“ ____ ” _____ 2025 р.

Виконала студентка
Лиса А.М.
“ ____ ” _____ 2025 р.

Київ 2025

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мультимедійних систем і технологій

ЗАТВЕРДЖУЮ

Зав.кафедри мультимедійних систем,
доцент, к.ф-м.н.
_____ О. П. Жежерун (підпис)
“ _____ ” _____ 2025 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на кваліфікаційну роботу

студентці Лисій Анастасії Миколаївні факультету інформатики 4 курсу

ТЕМА: Розробка застосунку для здорового харчування з системою
рекомендацій на базі Core ML: ефективність інтеграції локальних моделей у
мобільні HealthTech-рішення

Зміст ТЧ до кваліфікаційної роботи:

Індивідуальне завдання

Анотація

Вступ

1. Теоретичний фундамент дослідження
2. Технології та інструменти, використані під час розробки
3. Практична реалізація: створення застосунку з системою рекомендацій

Висновок

Джерела

Додатки

Дата видачі “ _____ ” _____ 2025 р. Керівник _____ (підпис)

Завдання отримано _____ (підпис)

Календарний план виконання роботи

№	Назва етапу кваліфікаційної роботи	Термін виконання етапу
1	Вибір і затвердження теми	01.10.2024 – 12.10.2024
2	Дослідження теоретичних матеріалів	15.10.2024 – 10.11.2024
3	Реалізація застосунку	11.11.2024 – 10.01.2025
4	Написання текстової частини роботи	15.01.2025 – 23.03.2025
5	Коригування роботи	25.03.2025 – 01.04.2025
6	Оформлення презентації	15.04.2025 – 05.05.2025

Лиса А. М. _____

Афонін А. О. _____

“ ____ ” _____ 2025 р.

Зміст

Вступ	7
Розділ 1: Теоретичний фундамент дослідження.....	9
1.1 Поняття рекомендаційної системи	9
1.2 Аналіз популярних сервісів з використанням рекомендацій.....	10
1.3 Типи рекомендаційних систем.....	11
1.3.1 Collaborative Recommender system.....	11
1.3.2 Підхід на основі моделі (model-based).....	11
1.3.3 User-based Collaborative Filtering.....	12
1.3.4 Item-based Collaborative Filtering.....	14
1.3.5 Content-based Recommender system.....	14
1.3.6 Demographic-based Recommender system	15
1.3.7 Utility-based Recommender system.....	16
1.3.8 Knowledge-based Recommender system.....	17
1.3.9 Hybrid Recommender system	17
1.4 Проблема холодного старту в рекомендаційних системах	20
Висновок до першого розділу	25
Розділ 2: Технології та інструменти, використані під час розробки	26
2.1 Core ML	26
2.2 Core Data.....	28
2.3 SwiftUI.....	29
Висновок до другого розділу	30
Розділ 3: Практична реалізація: створення застосунку з системою рекомендацій	32
3.1 Вибір датасету та його підготовка.....	32
3.2 Створення рекомендаційної системи	33
3.3 Реалізація збереження даних.....	35
3.4 MVVM – як обрана архітектура.....	36
3.5 Реалізація користувацького інтерфейсу.....	37
3.5.1 Головний екран застосунку	37

3.5.2 Екран рецепта	38
3.5.3 Екран зі списком покупок.....	38
3.5.4 Екран зі збереженими рецептами	39
Висновок до третього розділу	39
Висновок	40
Джерела.....	42
Додатки	44

Анотація

У даній роботі детально досліджено рекомендаційні системи, їх класифікації, механізми роботи та поширену проблему холодного старту. Проведено аналіз популярних сервісів, які використовують рекомендаційні системи. Окрім цього, у роботі розглянуто інструменти для розробки мобільних додатків для iOS, а також реалізовано HealthTech-застосунок Dish, що надає рекомендації рецептів на основі локальної ML-моделі.

У першому розділі проаналізовано теоретичні основи створення рекомендаційних систем, їх види та принципи роботи, а також проблему холодного старту. У другому розділі описано технології та інструменти що були використанні під час створення мобільного додатку. У третьому розділі наведено практичну реалізацію застосунку для здорового харчування, включаючи побудову рекомендаційної системи, збереження даних та створення користувацького інтерфейсу.

Результатом роботи є функціональний застосунок для iOS, що надає індивідуальні рекомендації рецептів на основі вподобань користувачів. Це доводить можливість ефективного впровадження локальних ML-моделей у мобільні рішення.

Вступ

У сучасному світі вести здоровий спосіб життя вже не є трендом - це стало необхідністю, частиною базового піклування про себе. Правильне харчування, фізична активність, контроль за самопочуттям - усе це більше не сприймається як щось додаткове, це стало must-have для кожної людини. Завдяки розвитку цифрових технологій слідкувати за здоровим способом життя стало значно простіше. Особливо зручно для цього використовувати різноманітні застосунки, які можна встановити на свій смартфон, що завжди під рукою. Таким чином можна легко знайти необхідну інформацію не витрачаючи багато часу, а використання рекомендаційних систем ще більше спрощує цей процес.

Застосунок Dish, розроблений у межах цієї роботи, є прикладом такого рішення: він дозволяє користувачеві швидко отримати персоналізовану рекомендацію страв на основі власних смакових вподобань завдяки використанню рекомендаційної системи.

Об'єктом дослідження є процес створення персоналізованих рекомендацій у сфері здорового харчування на iOS. Використання локальних моделей машинного навчання на базі Core ML для реалізації рекомендаційної системи у мобільному HealthTech-застосунку на платформі iOS є предметом даного дослідження є.

Мета роботи - дослідити рекомендаційні системи та реалізувати застосунок з урахуванням потреб користувачів та особливостей iOS-середовища.

Робота складається з трьох розділів:

У першому розділі закладено теоретичний фундамент дослідження: розглянуто основи роботи рекомендаційних систем, їх типи, приклади використання у популярних сервісах та детально описано проблему холодного старту і методи її вирішення.

У другому розділі описано технології та інструменти, використані під час розробки: проаналізовано платформи й інструменти iOS-розробки, зокрема Core ML, Create ML, Core Data та SwiftUI, які використовувалися під час розробки застосунку.

В останньому розділі розписано практичну реалізацію: створення застосунку Dish з системою рекомендацій: описано архітектуру застосунку, реалізацію рекомендаційної логіки, збереження даних, вибір архітектури та розробку користувацького інтерфейсу.

Практичне значення одержаних результатів полягає у можливості застосування розробленого застосунку Dish. Дане рішення може стати основою для створення комерційних HealthTech-продуктів, орієнтованих на індивідуальні потреби користувачів, а також як основа для подальших досліджень і вдосконалення рекомендаційних систем в умовах мобільного середовища.

Розділ 1: Теоретичний фундамент дослідження

1.1 Поняття рекомендаційної системи

Сьогодні ринок пропонує настільки багато варіантів, що вибір який фільм переглянути чи яку страву приготувати на вечерю може стати справжнім викликом для користувача. Зараз важко уявити якісний програмний продукт без рекомендаційної системи. Адже вона аналізує інтереси користувачів і допомагає підібрати щось релевантне серед безлічі доступних опцій.

Рекомендаційна система - це інструмент, який належить до сфери штучного інтелекту та машинного навчання і працює з великими обсягами даних, щоб допомагати людям знаходити відповідні товари, послуги чи інформацію серед великої кількості доступних варіантів. Завдяки даному інструменту можна легко проаналізувати дані про поведінку користувачів, зокрема історію переглядів, покупки, вподобання, а також можна врахувати певні загальні характеристики групи користувачів [1].

Такі системи створюються для виявлення закономірностей у взаємодії користувачів з об'єктами. Вони здатні визначати інтереси окремої людини і пропонувати саме той продукт, який найімовірніше буде цікавим або корисним. Завдяки цьому інструменту компанії можуть покращити досвід користувачів, пропонуючи персоналізовані рекомендації - від фільмів і музики до товарів чи послуг.

Рекомендаційні системи, також відомі як системи рекомендацій, належать до типу інформаційних фільтрів, які намагаються передбачити, що саме може сподобатися користувачеві або який рейтинг він може поставити певному продукту чи контенту. Ці системи широко використовуються у

цифровому середовищі, особливо там, де важлива персоналізація - наприклад, в інтернет-магазинах, соціальних мережах, стрімінгових платформах.

1.2 Аналіз популярних сервісів з використанням рекомендацій

Важко уявити сучасну популярну платформи без рекомендаційної системи. І справді, велика кількість продуктів на ринку почали використовувати рекомендації щоб підвищити залученість користувачів у свій продукт.

До прикладу, Netflix рекомендує фільми використовуючи *model-based collaborative filtering*. Модель аналізує взаємодії великої кількості користувачів і будує прогнози щодо фільмів та серіалів, які можуть зацікавити конкретного глядача. Це допомагає скоротити час на пошук контенту і таким чином значно покращує користувацький досвід.

Amazon використовує *item-based collaborative filtering* з використанням моделей, таким чином маркетплейс рекомендує товари на основі того що інші користувачі переглядали, купували або ж додавали в корзину разом. Головна перевага такого підходу в тому що можна створювати персоналізовані пропозиції навіть при обмеженій інформації про нових користувачів.

Spotify широко відомий своєю рекомендаційною системою - багато користувачів відмічають високу якість рекомендацій на стрімінговій платформі. Spotify поєднує *model-based collaborative filtering* з контентними методами, аналізуючи прослуховування користувачів і знаходячи подібні музичні вподобання. Для створення рекомендацій на основі контенту платформа самостійно визначає багато ознак музичних творів як от ритмічність [2].

1.3 Типи рекомендаційних систем

1.3.1 Collaborative Recommender system

Колаборативна фільтрація є чи не найпопулярнішим методом побудови рекомендацій, що базується на вподобаннях великої кількості користувачів. Її суть полягає в тому, що система використовує попередні дії користувачів (оцінки, покупки, перегляди), щоб знаходити закономірності та пропонувати нові найбільш відповідні варіанти.

Таким чином алгоритм не аналізує самі об'єкти, а обробляє поведінкові дані користувачів. Тобто, якщо кілька людей у минулому мали багато збігів у вподобаннях, мали схожі смаки то можна припустити, що й надалі вони цікавитимуться схожим контентом. До прикладу, якщо користувач поставив хорошу оцінку певному рецепту страви, а інший користувач має схожий смак то система рекомендуватиме цей рецепт і йому.

Колаборативну фільтрацію розділяють на два основні підходи - це підхід на основі моделі та підхід на основі пам'яті, останній в свою чергу ділиться на фільтрацію за користувачем та фільтрація за товаром.

1.3.2 Підхід на основі моделі (model-based)

Фільтрація на основі моделі передбачає використання моделі машинного навчання. В основі такого підходу лежить аналіз минулих взаємодій між користувачами та об'єктами, на основі яких модель навчається передбачати вподобання користувачів.

Однією з головних переваг даного методу є його здатність ефективно працювати з великими й нещільними наборами даних - система не порівнює

кожного користувача з усіма іншими напряму, а вивчає загальні закономірності в поведінці. Таким чином рекомендаційна система працює дуже швидко. Окрім того, існує дослідження [3] за результатами якого, метод на основі моделі має середній час обчислення в десять разів швидший, ніж методи, що базуються на пам'яті.

Проте, цей підхід має і певні недоліки. Його реалізація може бути дещо складнішою, порівнюючи з підходом на основі пам'яті, адже потребує розуміння принципів машинного навчання. Також при використанні моделей може бути важко зрозуміти чому були саме такі рекомендації, чи точно вони релевантні. Моделі дають досить точні результати, однак зрозуміти, чому була зроблена саме така рекомендація, буває досить складно [4].

1.3.3 User-based Collaborative Filtering

User-based Collaborative Filtering - один з видів memory-based підходу який ґрунтується на припущенні, що користувачі які мали схожі вподобаннями в минулому, поділятимуть схожі смаки і в майбутньому. Наприклад, якщо двоє користувачів переглянули однакові рецепти та виставили їм схожі оцінки, то можна очікувати, що їхні оцінки для інших рецептів також будуть подібними. Припустимо, що користувач А натрапив на рецепт, який користувач В ще не бачив, і поставив йому високу оцінку. У такому випадку якщо користувачі А і В мають високий ступінь схожості з огляду на попередні вподобання, то можна обґрунтовано спрогнозувати, що цей рецепт також сподобається користувачу В.

Даний вид фільтрації реалізується через алгоритм найближчих сусідів, який виконує дві ключові задачі:

1. Визначення K -найближчих користувачів, найбільш подібних до активного користувача a , шляхом застосування метрики схожості w , що оцінює ступінь відповідності між парами користувачів.

$$\text{Similarity}(a, i) = w(a, i), i \in K$$

2. Використання оцінок знайдених сусідів для прогнозування оцінок або формування рекомендацій для користувача a .

Тобто, формується матриця взаємодій між користувачами та об'єктами (див. Рис. 1), де система намагається передбачити оцінки для тих об'єктів, з якими активний користувач ще не взаємодіяв. Прогноз ґрунтується на інформації, отриманій від інших користувачів зі схожими вподобаннями [5].


























					
A					
B					
C					
D					
E					

Рисунок 1. Матриця взаємодій

1.3.4 Item-based Collaborative Filtering

Item-based Collaborative Filtering - це memory-based підхід зосереджений на пошуку тих об'єктів, які найбільше схожі на ті, що вже сподобалися потрібному користувачеві. Наприклад, якщо користувачі, які переглядали рецепт А, часто переглядають рецепт В, то система запропонує рецепт В кожному, хто дивився рецепт А незалежно від поведінки інших користувачів.

Цей метод можна умовно поділити на два основні етапи:

1. Визначення подібності між елементами, що може виконуватись за допомогою:
 - косинусної подібності
 - кореляційної міри
 - модифікованої косинусної подібності
 - коефіцієнта Жаккара
2. Обчислення прогнозованих оцінок на основі:
 - зваженого середнього значення
 - методів регресії

На відміну від фільтрації за користувачем у даному підході схожість між об'єктами обчислюється наперед, що дозволяє уникнути додаткового етапу пошуку найближчих сусідів користувача при формуванні рекомендацій.

1.3.5 Content-based Recommender system

Контентно-орієнтована фільтрація (Content-based Filtering) - це підхід у побудові рекомендаційних систем, який пропонує об'єкти, подібні до тих, якими вже цікавився користувач, ґрунтуючись на їх властивостях. Такими характеристиками можуть бути, наприклад, категорія рецепту, автор, час

приготування, калорійність або ж основні інгредієнти страви. Дана рекомендаційна система дуже добре підходить для платформ, які пропонують велику різноманітність об'єктів, наприклад маркетплейс, адже за допомогою цього підходу можна сформувати персоналізовані рекомендації на основі попередніх дій користувача.

Контентно-орієнтована фільтрація зазвичай створюється з використанням класифікаційних моделей машинного навчання через підхід векторних представлень, що передбачає порівняння векторів користувача та об'єкта у просторі ознак або з використанням дерев рішень.

Чи не найголовнішою перевагою цього методу є незалежність від інших користувачів, що надзвичайно важливо для нових або непопулярних об'єктів, а також для користувачів з особливими вподобаннями. Тим не менш, ефективність даного підходу сильно залежить від якості та повноти характеристик товарів [6].

1.3.6 Demographic-based Recommender system

Демографічна рекомендаційна система формує рекомендації беручи за основу соціально-демографічні характеристики певних груп користувачів.

Такий підхід використовує доступну інформацію про користувачів яку може зібрати веб-сайт або застосунок, наприклад країну, вік, стать, локацію, тип пристрою та інші атрибути за якими можна групувати користувачів. Таким чином рекомендаційна система аналізує типові переваги всередині кожної групи, аби запропонувати релевантні об'єкти, а також щоб заповнити дані у випадках недостатньої персоналізації.

Рекомендації, що ґрунтуються на демографічних даних, досить часто застосовуються на різноманітних платформах, адже вони не потребують детальної інформації про індивідуальні вподобання користувача. Тому такі системи можуть забезпечувати базову персоналізацію при першому використанні - таким чином усуваючи проблему холодного старту [7].

Отже, демографічна фільтрація належить до чи не найлегших у реалізації рекомендаційних механізмів, оскільки вимагає обмеженого обсягу вхідних даних для створення загальних рекомендацій і немає проблеми холодного старту, яка зустрічається в багатьох інших рекомендаційних системах. Проте її ефективність дуже сильно залежить від якісної сегментації цільової аудиторії та її дослідженні.

1.3.7 Utility-based Recommender system

Рекомендаційні системи, що використовують підхід на основі корисності, формують пропозиції, виходячи з індивідуальної оцінки цінності кожного об'єкта для конкретного користувача. Основною проблемою при реалізації такого методу є створення релевантної функції корисності, яка б точно відображала уподобання користувача.

У даній системі потрібно розробляти особливий підхід до побудови цієї функції в кожній галузі окремо беручи до уваги її особливості, щоб мати змогу обчислювати корисність кожного об'єкта з урахуванням усіх потреб.

Важливою перевагою такого типу систем на відміну від контентно-орієнтованих є здатність включати у розрахунок не лише властивості об'єкта, але й додаткові фактори, наприклад, наявність товару на складі чи рейтинг виробника. Це може бути дуже корисним для інтернет-магазинів оскільки

використовуючи цю систему можна пропонувати лише ті товари, які є доступними на даний момент для покупки.

1.3.8 Knowledge-based Recommender system

Рекомендаційна система, що ґрунтується на знаннях, формує пропозиції не за допомогою аналізу попередніх вподобань користувача, а шляхом реагування на чітко сформульовані запити або вимоги. Користувач може вказати певні критерії чи описати бажані характеристики об'єкта або ж дати приклад такого об'єкта, після чого система здійснює пошук релевантних варіантів.

Такий підхід зазвичай застосовується у випадках коли користувач очікує знайти товари які відповідають конкретним запитам і мають певні властивості. Наприклад, під час пошуку житла на веб-сайті - користувач вказує параметри, як-от бюджет, поверх, тип будинку, стан ремонту, а система підбирає релевантні варіанти [8].

На перший погляд, це може здатися простим фільтром, проте система рекомендацій на знаннях використовує дещо складніші підходи - вона враховує зв'язки між властивостями і може адаптуватися до узагальнених запитів.

1.3.9 Hybrid Recommender system

Гібридні рекомендаційні системи - це, зазвичай, комбінація двох підходів, хоча іноді можуть поєднувати й більше, що дозволяє ефективніше використовувати сильні сторони кожного з них та нівелювати певні недоліки.

Найчастіше комбінуються методи спільної фільтрації (які аналізують взаємодію користувачів з об'єктами) та фільтрації на основі контенту (яка враховує характеристики елементів). Проте, можуть поєднуватися й інші методи – як от, демографічний або заснований на корисності. Основна ціль такого поєднання - створити більш гнучку та ефективну систему, яка може легко справлятися з викликами на кшталт проблеми холодного старту, а також покращити різноманітність і точність рекомендацій. Наприклад, такий підхід може об'єднувати переваги колаборативного методу у виявленні схожих моделей поведінки користувачів з точністю контентної фільтрації, яка враховує особливості об'єктів, що в результаті дає релевантніші рекомендації. Таким чином будуть усунуті головні проблеми цих типів фільтрації.

У своїй статті "Гібридні рекомендаційні системи: Дослідження та експерименти" [9] Берк виокремлює сім основних підходів до побудови гібридних рекомендаційних систем. Ці підходи відображають різні способи поєднання, що дає змогу адаптувати систему під конкретні цілі та контексти використання.

1. **Weighted** – у даній системі рекомендацій використовуються кілька моделей, кожна з яких добре підходить для аналізу даних. Результати цих моделей комбінуються із заздалегідь визначеними ваговими коефіцієнтами, які залишаються незмінними.

Наприклад, можна поєднати модель колаборативної фільтрації з контентною в рівному співвідношенні.

Тобто при такому підході ми об'єднуємо кілька стратегій у єдиний лінійний механізм, що забезпечує більш збалансовані рекомендації для різних типів даних.

2. Switching – у даній системі передбачено вибір конкретного методу рекомендацій залежно від контексту та умов. Для цього формується модель, яка аналізує дані на рівні окремих елементів і застосовує заздалегідь визначені правила або умови вибору - наприклад, з урахуванням профілю користувача або інших характеристик.

Такий підхід додає до архітектури рекомендаційної системи ще один шар - механізм прийняття рішень, який визначає, яка саме модель буде використана в кожному конкретному випадку. Завдяки цьому система може гнучко реагувати на обмеження та переваги кожного окремого методу, підвищуючи якість рекомендацій.

3. Mixed – у цій системі спочатку використовується профіль користувача та його характеристики для створення різних наборів даних. Після чого система рекомендацій інтегрує ці набори в модель і комбінує їхні прогнози для формування фінальних рекомендацій.

Такий підхід дозволяє змішаній гібридній системі одночасно видавати кілька варіантів рекомендацій, а також вибирати найбільш відповідний набір даних для конкретної моделі, що підвищує її продуктивність.

4. Feature Combination - особливістю цієї системи є додавання віртуальної моделі рекомендацій, що працює як функціональне перетворення для вихідного набору даних профілю користувача. До прикладу, можна

інтегрувати елементи моделі колаборативних рекомендацій з моделлю контентної.

5. Feature Augmentation - дозволяє підвищити ефективність основної системи без зміни базової моделі рекомендацій. Наприклад, за допомогою правил асоціації можна розширити набір даних акаунта користувача, що, в свою чергу, покращить результативність моделі рекомендацій, заснованої на контенті.
6. Meta-Level - досить подібна до підходу з аугментацією, оскільки одна з моделей формує розширений набір даних для використання в основній системі рекомендацій. Проте, в даному підході замість початкового набору даних застосовується результат навченої допоміжної моделі, який слугує вхідними даними для основної рекомендаційної моделі.
7. Cascade - передбачає побудову рекомендаційної системи у вигляді чіткої ієрархії, де перша модель генерує основні результати, а друга - уточнює їх, усуваючи певні недоліки, як-от однакові оцінки.

Оскільки більшість реальних датасетів є розрідженими, допоміжна модель у такій структурі може ефективно справлятися з проблемами, пов'язаними з рівнозначними оцінками або браком інформації.

1.4 Проблема холодного старту в рекомендаційних системах

Проблема холодного старту – це проблема, що виникає коли рекомендаційна система не має достатнього обсягу попередніх даних для формування персоналізованих рекомендацій. У таких умовах система не здатна одразу надавати точні й релевантні пропозиції новим користувачам або

ж рекомендувати нові об'єкти, оскільки необхідно зібрати достатньо інформації для створення якісних рекомендацій. Ця проблема притаманна не для усіх типів рекомендаційних систем.

За нормальних умов рекомендаційні системи використовують такі типи даних, як покупки, оцінки, коментарі або інша взаємодія з контентом - усе це є основними даними для навчання моделей. На основі цих характеристик система формує припущення про вподобання користувача та створює персоналізовані пропозиції.

Холодний старт є надзвичайно поширеною проблемою, з якою стикаються розробники, аналітики даних та ml-розробники під час проектування рекомендаційних систем. Коли на платформі з'являється новий користувач або новий об'єкт, система працює з обмеженим обсягом інформації або з її відсутністю, що унеможлиблює надання релевантних рекомендацій. Це може мати поганий вплив на бізнес оскільки користувачі можуть втратити інтерес до платформи під час першого користувацького досвіду [10].

Розділяють два типи даної проблеми: холодний старт продукту та холодний старт користувача.

Холодний старт користувача виникає тоді, коли він ще не здійснив жодної взаємодії з системою: не переглядав товари, не зробив покупку, не залишав відгуків, не додав улюблені товари в корзину. Через відсутність таких даних система не має змоги точно визначити інтереси цього користувача, що призводить до надання нерелевантних або випадкових рекомендацій.

Коли новий користувач отримує нецільові рекомендації, він може швидко втратити інтерес до сервісу ще до того, як система встигне зібрати потрібну інформацію для якісного прогнозування вподобань.

Проблема холодного старту для нового користувача особливо помітна в динамічних системах, де кількість нових відвідувачів стабільно зростає. У таких умовах звичайні методи рекомендацій не є надто ефективними.

Проблема холодного старту товару виникає в ситуаціях, коли з'являється новий об'єкт, про якого ще немає достатньо інформації. Через брак взаємодій система не буде включати цей об'єкт в рекомендації для користувачів.

Ця проблема стосується не лише нових товарів, але й тих, що вже деякий час були розміщені на платформі, однак залишаються малопомітними або непопулярними, наприклад якщо це якісь нішеві товари якими цікавиться мала кількість користувачів. Якщо об'єкт має обмежену кількість відгуків чи взаємодій, система не може зібрати достатньо даних, щоб визначити, кому саме варто його рекомендувати. Як наслідок - знижується частота появи таких продуктів у списках рекомендацій, що негативно впливає на охоплення об'єкта.

Існує багато методів для вирішення проблеми холодного старту. Одним з найрозповсюдженішим є збір початкових вподобань користувача під час реєстрації. Такий підхід передбачає заповнення короткої анкети новим користувачем, що дозволяє системі сформувати базовий профіль на основі його даних наданих користувачем під час реєстрації. Такий метод дає змогу

рекомендаційній системі одразу запропонувати щось релевантне, навіть без історії взаємодій.

Наприклад, популярна стримінгова платформа Apple Music застосовує подібну стратегію під час реєстрації, тобто при створенні акаунту користувачам пропонують обрати музичні жанри та улюблених виконавців (див. Рис. 2). Отримані результати служать початковим джерелом даних для алгоритму, який формує персоналізований плейлист з рекомендованими треками.



Рисунок 2. Екрани реєстрації акаунта в Apple Music

Проте, варто враховувати, що надмірна кількість запитань при реєстрації може знизити мотивацію користувача продовжити знайомство з платформою. Також цей метод є недоречним у випадках коли для базового використання сервісу не потрібна реєстрація. Наприклад, Prom - український маркетплейс,

надає можливість користувачам переглядати оголошення без попередньої реєстрації.

У таких випадках більш доречно використовувати одну з найрозповсюдженіших стратегій усунення проблеми холодного старту - використання підходу, що базується на популярності. У даному випадку новим користувачам демонструють найбільш затребувані товари, сформовані на основі загальних уподобань інших користувачів. Також доцільно окремо виділяти нові об'єкти - це дозволяє привернути до них увагу користувачів і стимулювати взаємодію. Після цього систему можна покращити використанням контекстних даних. Цей підхід ґрунтується на зборі й аналізі інформації про користувача, такої як його геолокація, демографічні характеристики, тип пристрою, налаштування браузера або поведінка під час сеансу. Дана інформація дозволяє системі адаптувати контент під конкретного користувача навіть за відсутності історичних даних.

Дану стратегію активно застосовують платформи для пошуку нерухомості на кшталт Лун. Наприклад, при першому відвідуванні сайту без реєстрації платформа автоматично визначає місце розташування користувача й використовує отримані дані для показу персоналізованих пропозицій – нерухомості у місті, де знаходиться користувач. Це значно підвищує релевантність контенту. Також не менш популярним способом усунення проблеми холодного старту є використання гібридної рекомендаційної системи. Наприклад, поєднавши контенту фільтрацію з колаборативною як це робить популярна платформа для прослуховування музики Spotify.

Висновок до першого розділу

У першому розділі було детально проаналізовано різні підходи до побудови рекомендаційних систем: колаборативну фільтрацію (user-based, item-based та model-based), content-based, demographic-based, utility-based, knowledge-based та гібридні моделі. Також окремо було розглянуто проблему холодного старту та способи вирішення.

Кожен із розглянутих методів має свої переваги та недоліки:

User-based та item-based методи добре працюють для малих обсягів даних, але погано масштабуються.

Content-based пропонує схожі за своїми характеристиками об'єкти, таким чином маючи досить одноманітні рекомендації, адже даний підхід не враховує вподобання інших користувачів.

Demographic-based не має проблеми холодного старту проте потребує певної інфортації про користувача та якісний аналіз кожного сегмента аудиторії.

Utility-based є дещо складним через необхідність створення функції корисності в кожному окремому випадку.

Knowledge-based вимагає багато вхідних даних від користувача, тому більше підходить у випадку коли користувач знає що йому потрібно.

Гібридні рекомендаційні системи мають великий потенціал, адже за рахунок поєднання кількох методів можуть легко усунути недоліки кожного, проте вони дещо складні в реалізації і зазвичай використовуються як покращення для вже наявної рекомендаційної системи.

На основі проведеного дослідження було прийнято рішення використовувати модельно-орієнтовану колаборативну фільтрацію. Model-based добре підходить для масштабування та працює значно швидше за

memory-based підхід, проте може бути дещо важчим для розуміння. Також даний метод забезпечує високу точність у рекомендаціях та добре працює з розрідженими даними на відміну від memory-based методів. До того ж, model-based підхід широко використовується у популярних платформах, таких як Netflix та Spotify. Що є ще одним доказом ефективності даного методу побудови рекомендаційної системи.

Розділ 2: Технології та інструменти, використані під час розробки

2.1 Core ML

Core ML є відносно новим фреймворком від компанії Apple, за допомогою цього фреймворка можна використовувати модель машинного навчання відразу на пристрої. Core ML доступний на таких платформах: iOS, macOS, watchOS і tvOS. Важливою перевагою даного інструменту є його автономність - тобто можливість працювати без підключення до інтернету, адже усі розрахунки відбуваються на пристрої що також позитивно впливає на швидкість роботи [11].

Окрім того Core ML було спеціально оптимізовано Apple для мінімального використання пам'яті та енергоспоживання, виконуючи обчислення безпосередньо на центральному та графічному процесорах. Також з важливих переваг цього фреймворку варто відзначити безпеку. Даний інструмент забезпечує конфіденційність і мінімізує ризик викрадення особистих даних користувача оскільки дані користувачів, наприклад взаємодії з контентом нікуди не передаються, а відразу опрацьовуються на пристрої. Core ML працює на основі низькорівневих фреймворків від компанії Apple: Accelerate, Metal і BNNS.

Для спрощення процесу тренування моделей компанія у 2018 році представила Create ML.

Create ML - інструмент, що дозволяє створювати моделі машинного навчання у форматі Core ML за допомогою зручного інтерфейсу в Xcode (див. Рис. 3). Create ML має велику кількість моделей (див. Рис. 4), серед яких є класифікація зображень, рухів, звуків, виявлення об'єктів, таблична регресія, рекомендаційна модель тощо.



Рисунок 3. Тренування моделі за допомогою Create ML

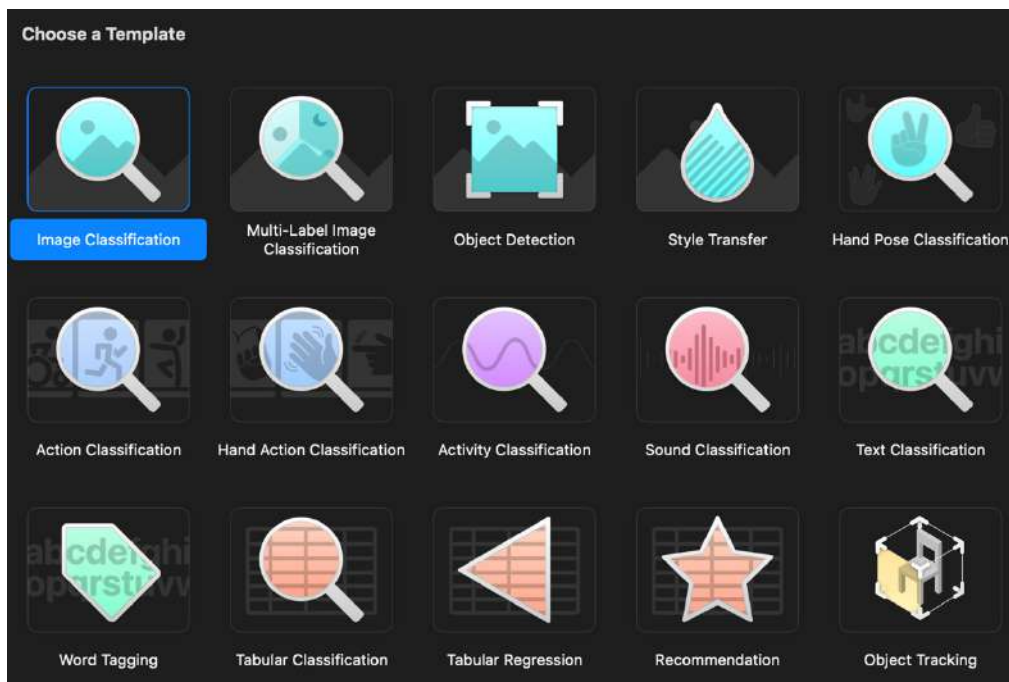


Рисунок 4. Моделі Create ML

2.2 Core Data

Одним з найпопулярніших фреймворків для роботи з даними на iOS є Core Data. Цей фреймворк дозволяє інкапсулювати дані з бази даних у об'єкти і працювати з ними як зі звичайними класами у Swift. Він добре інтегрований у Xcode та має зручний інтерфейс. Стандартно Core Data працює на базі SQLite, проте також може працювати на основі XML та бінарного формату [12].

Основною перевагою Core Data у порівнянні з іншими схожими фреймворками є його зрілість, адже інструмент існує вже понад дванадцять років та нативність – для його використання не потрібно додавати в проект якість залежності. Головним з недоліків даного фреймворку є неможливість використання його на інших платформах, а також дещо повільніша робота

порівняно, наприклад, з Realm. Проте, Core Data добре підходить для проектів зі складними моделями даних, які постійно змінюються.

2.3 SwiftUI

SwiftUI - це фреймворк від компанії Apple для створення користувацького інтерфесу на iOS, iPadOS, watchOS, tvOS та macOS. SwiftUI був представлений у 2019 році як альтернатива UIKit [13]. Кожен з цих фреймворків має свої переваги і недоліки (див. Таблиця 2.1), проте зараз все частіше використовується SwiftUI для створення нових застосунків. Також, за потреби ці фреймворки можна поєднувати в одному проекті.

Таблиця 2.1. Порівняння SwiftUI та UIKit

Характеристика	SwiftUI	UIKit
Парадигма програмування	Декларативний підхід	Імперативний підхід
Live preview	Так	Ні
Платформи	iOS, macOS, watchOS, tvOS	iOS
Interface Builder	Ні	Storyboard
Версії iOS	Підтримує усі версії від iOS 13	Підтримує усі версії від iOS 9
Управління станом UI	Вбудоване через @State, @ObservedObject	Ручне оновлення, зазвичай реалізується через патерн Делегат
Підтримка темної теми	Автоматично	Вручну
Поріг входу	Простий для освоєння, проте, можуть бути складнощі з кастомізацією	Відносно складний для початківців, але надає більше можливості для кастомізації

Висновок до другого розділу

У другому розділі було розглянуто основні технології, використані під час розробки застосунку: Core ML, Create ML, Core Data та SwiftUI. Усі перераховані інструменти є частиною екосистеми Apple, що забезпечує високу продуктивність, стабільність та зручність роботи з ними.

Core ML надає можливість виконання обчислень безпосередньо на пристрої користувача, що дозволяє системі рекомендацій працювати автономно, швидко й без потреби в інтернет-з'єднанні. Це також гарантує конфіденційність даних, адже особиста інформація обробляється на пристрої. Завдяки Create ML модель можна швидко і просто натренувати без використання сторонніх інструментів.

Core Data є чудовим фреймворком для роботи з даними, оскільки він інтегрований в Xcode, не потребує додаткових залежностей та є досить зрілим. Головним недоліком Core Data є відсутність кросплатформеності та відносно високий поріг входу.

SwiftUI – це хороше рішення для створення сучасного, адаптивного та візуально привабливого інтерфейсу з використанням декларативного підходу. Підтримка live preview, тобто оновлення користувацького інтерфейсу автоматично після змінення коду, робить розробку більш простою на швидкою.

Використання цих інструментів дозволить реалізувати застосунок, який є продуктивним, безпечним, адаптивним до сучасних стандартів, а також готовим до масштабування та подальшого розвитку. Нативність, сумісність та оптимізація під роботу на iPhone є ключовими перевагами.

Розділ 3: Практична реалізація: створення застосунку з системою рекомендацій

Для демонстрації ефективності інтеграції локальної моделі у створення системи рекомендацій було розроблено мобільний застосунок Dish, який надає користувачам персоналізовані рекомендації щодо рецептів на основі їхніх вподобань та попередніх взаємодій.

3.1 Вибір датасету та його підготовка

Для реалізації застосунку було обрано датасет Food.com - Recipes and Reviews [14].

Головні переваги даного датасету:

- велика кількість даних (початково датасет містив понад пів мільйона рецептів та понад мільйон відгуків користувачів)
- потрібний формат (датасет містить в собі два окремі файли у форматі csv)
- повнота даних (кожен рецепт містить усі необхідні дані)
- висока якість зображень

Підготовка датасету:

Уся обробка датасету здійснювалася за допомогою мови програмування Python та бібліотеки Pandas.

На початку роботи було здійснено фільтрацію даних, таким чином спершу було видалено всі рецепти що мали порожні значення. Після чого було відфільтровано рецепти за критерієм корисності: тобто обрано лише ті рецепти що мали серед ключових слів значення “Healthy” (див. Рис. 5).

```

columns_to_keep = [
    "RecipeId", "Name", "RecipeCategory", "Images", "AggregatedRating", "Description", "Calories",
    "RecipeIngredientParts", "RecipeInstructions", "Keywords"
]
filtered_df = recipes[columns_to_keep].dropna()
filtered_df = filtered_df[filtered_df["Images"] != "character(0)"]
healthy_recipes = filtered_df[filtered_df["Keywords"].str.contains("Healthy", na=False, case=False)]
healthy_recipes = healthy_recipes.drop(columns=["Keywords"])

```

Рисунок 5. Обробка датасету на мові програмування Python

Наступним кроком у роботі з датасетом була нормалізація даних. Зокрема було очищено текстові поля від зайвих символів (наприклад, с(...), лапки).

Останнім етапом був процес створення нового датасету лише з потрібними для роботи даними. Таким чином було залишено лише необхідні колонки – такі як id, назва, опис, інгредієнти, інструкції, категорія, калорійність, рейтинг та зображення.

Після усіх етапів обробки, розмір датасету скоротився до 10 869 рецептів та 39 531 відгуків. Цього обсягу достатньо для навчання, тестування та побудови якісної рекомендаційної системи.

3.2 Створення рекомендаційної системи

Для реалізації персоналізованих рекомендацій рецептів була натренована модель машинного навчання на основі файлу з відгуками користувачів. Даний файл у форматі csv містить три поля: RecipeId, AuthorId, Rating.

Для тренування моделі був обраний метод колаборативної фільтрації, що аналізує уподобання користувачів і на основі цього рекомендує рецепти, які можуть їх зацікавити. Модель було натреновано за допомогою дуже зручного інструменту від компанії Apple – Create ML, що дозволяє

створювати та тренувати моделі машинного навчання безпосередньо в середовищі Xcode. Цей підхід дозволив ефективно та швидко обробити великі обсяги даних та створити модель машинного навчання у форматі `.mlmodel`.

Після чого модель було інтегровано в iOS-додаток за допомогою фреймворка Core ML. Для зручної роботи з моделлю був реалізований спеціальний Swift-клас, який відповідає за взаємодію з моделлю і генерує рекомендації для користувачів на основі їх оцінок і вподобань (див. Рис. 6).

```
func getRecommendedRecipeIDs(userRatings: [Int64: Double], exclude: [Int64]) -> [Int64]{
    do{
        let recommender = try RecipeRecommender(configuration: MLModelConfiguration())
        let input = RecipeRecommenderInput(items: userRatings, k: 200, restrict_: [], exclude: exclude)
        let result = try recommender.prediction(input: input)
        return result.recommendations
    }catch(let error){
        print(error.localizedDescription)
        return []
    }
}
```

Рисунок 6. Метод для створення рекомендацій

Загалом, логіка роботи рекомендаційної системи досить проста та інтуїтивно зрозуміла. Основним джерелом даних для рекомендацій є оцінки, які користувач виставляє рецептам у вигляді зірочок (див. Рис. 7). Система також враховує дії користувача для покращення персоналізованих рекомендацій. Зокрема, якщо користувач додає інгредієнти до списку покупок або зберігає рецепт, це автоматично інтерпретується системою як позитивна взаємодія. У такому випадку рецепту присвоюється максимальна оцінка – 5, що означає високу ймовірність того, що рецепт сподобався користувачу. Такий підхід дозволяє покращити якість рекомендацій навіть без прямого оцінювання.



Рисунок 7. Функціонал оцінювання рецепта

Оцінки користувача зберігаються в базі даних що дозволяють формувати набір рекомендованих рецептів. При відкритті головного екрана застосунку, якщо у користувача вже є дані про вподобання, то відображаються рекомендовані рецепти. Якщо ж у користувача ще не було взаємодій з рецептами то система показує список найпопулярніших рецептів - тобто тих, що отримали найбільше позитивних оцінок від інших користувачів.

Такий підхід дозволяє створити базову, проте дієву персоналізацію, яка з часом покращується завдяки активній взаємодії користувача з застосунком.

3.3 Реалізація збереження даних

Збереження даних у мобільному застосунку Dish реалізовано за допомогою технології Core Data, що є об'єктно-орієнтованою обгорткою над SQLite. Такий підхід дозволяє зручно працювати з об'єктною моделлю даних, забезпечуючи гнучке управління записами, зв'язками між сутностями, створенням, оновленням та видаленням даних.

Для зручного збереження даних було створено відповідні сутності:

- RecipeData - сутність для збереження рецептів, (id, name, image, rating, descriptionText, calories, ingredients, instructions), має зв'язок один до багатьох з RecipeCategoryData.
- RecipeCategoryData - сутність для збереження категорій рецептів (id, name), має зв'язок багато до одного з RecipeData.

- UserRatingData - сутність для збереження оцінок користувача (recipeId, rating)
- SavedRecipeData - рецепти, які зберіг користувач (id).
- ShoppingItemData - сутність для збереження продуктів, які користувач додав до свого списку покупок (id, name, isDone).

Для зручної роботи з базою даних було створено клас DataManager, що реалізований з використанням патерна Singleton. Даний клас містить методи для зчитування, збереження, оновлення та видалення даних, що охоплюють різні аспекти функціональності додатку.

3.4 MVVM – як обрана архітектура

MVVM (Model-View-ViewModel) є одним із популярних підходів до побудови архітектури застосунків, особливо в середовищі iOS. Його використання забезпечує чітку модуляризацію, що сприяє підвищенню перевикористання коду, спрощує процес автоматичного тестування та полегшує подальшу підтримку програмного продукту [15].

Складові MVVM архітектури (див. Рис. 8):

Model - відповідає за бізнес-логіку, збереження даних, різноманітні обчислення, взаємодію з сервером.

ViewModel – реалізовує логіку взаємодії View з Model. Головна задача даної структури – це опрацювання подій від користувача, наприклад натискання на кнопку чи ввід тексту, та збереження даних які мають відобразитися на екрані.

ViewModel не повинна мати доступ до елементів інтерфейсу.

View – відповідає за відображення користувацького інтерфейсу, а також фіксує отримані жести від користувача і передає їх у ViewModel. У View не повинно бути прямої взаємодії з Model.

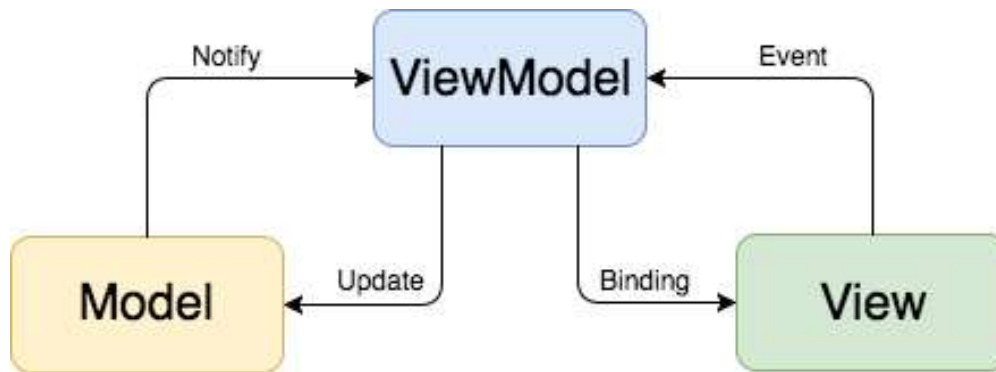


Рисунок 8. Схема роботи MVVM

Завдяки MVVM архітектурі досягається необхідна гнучкість у розробці, що дозволяє легко масштабувати проект, а також легко покривати необхідний функціонал тестами.

3.5 Реалізація користувацького інтерфейсу

3.5.1 Головний екран застосунку

Першим екраном, який бачить користувач після запуску застосунку, є головний екран (див. додаток А). У верхній частині розміщене поле для пошуку, за допомогою якого користувач може швидко знайти необхідний рецепт за його назвою.

Найбільшу частину екрану займає блок з рецептами. Якщо рекомендаційна система вже має зібрані дані про вподобання користувача, вона формує індивідуальні рекомендації на основі цих даних. У випадку, коли користувач новий і дані ще не зібрані, застосунок відображає найпопулярніші рецепти. Під час взаємодії користувача з рецептами список рекомендацій буде змінюватися, таким чином рецепти на головній сторінці будуть регулярно оновлюватися.

Список рецептів представлено за допомогою елемента SwiftUI LazyVGrid, який дозволяє оптимізовано відображати сітку з картинками, тобто дані підвантажуються поступово під час скролу екрану, що позитивно впливає на продуктивність застосунку.

Для зручної навігації у нижній частині екрану розташовано панель з кнопками, реалізовану за допомогою TabView. Вона забезпечує зручне та швидке перемикання між основними екранами застосунку, такими як головна сторінка, список покупок та збережені рецепти.

3.5.2 Екран рецепта

Екран рецепта містить усю інформацію про обрану страву (див. додаток Б). У верхній частині відображається зображення, короткий опис, категорія та назва. Також трохи нижче показано калорійність страви та її загальний рейтинг. Найбільшу частину екрану займає список інгредієнтів та інструкції для приготування. На даному екрані користувач має можливість зберегти рецепт, додати інгредієнти до списку покупок, а також оцінити рецепт щоб покращити рекомендації. У випадку якщо користувач зберігає рецепт і/або додає його інгредієнти у свій список продуктів - це сприймається як позитивна взаємодія і система автоматично виставляє відповідному рецепту максимальну оцінку.

3.5.3 Екран зі списком покупок

Екран представляє собою інтерактивний список з продуктами де користувач може позначити продукт як куплений, видалити непотрібний продукт або ж додати новий натиснувши на “+” (див. додаток В). Інтерфейс реалізовано таким чином, щоб забезпечити максимально просте та зрозуміле управління списком покупок під час реального походу до магазину.

3.5.4 Екран зі збереженими рецептами

Екран зі збереженими рецептами за структурою та зовнішнім виглядом подібний до головного екрана. Основна відмінність полягає в тому, що тут відображаються не рекомендовані рецепти, а ті, що користувач самостійно зберіг. Цей екран є дуже зручним щоб швидко знайти страву яка сподобалася. Користувачу не потрібно щоразу шукати рецепт, який він хоче приготувати - адже всі збережені рецепти доступні в одному місці. Це значно економить час і дозволяє легко повернутися до варіантів що вже сподобалася, аби приготувати страву.

Висновок до третього розділу

У третьому розділі розписано реалізацію мобільного застосунку Dish, що демонструє можливості машинного навчання для створення рекомендацій. Застосунок демонструє ефективність використання персоналізованих рекомендацій, сформованих за допомогою колаборативної фільтрації, що ґрунтується на оцінках та діях користувачів.

Було обґрунтовано вибір та процес обробки датасету, натреновано модель за допомогою Create ML, інтегровано її в iOS-додаток за допомогою фреймворка Core ML. Збереження даних забезпечено за допомогою Core Data, а загальна структура застосунку побудована на основі архітектури MVVM, що підвищує модульність, підтримуваність та масштабованість. Також, у розділі описано реалізацію користувацького інтерфейсу, який забезпечує зручну взаємодію з рекомендаційною системою.

Таким чином, розроблений застосунок Dish наочно демонструє можливості створення ефективної персоналізованої платформи з рецептами на основі локальних ML-моделей у мобільному середовищі iOS.

Висновок

У ході виконання дипломної роботи було досліджено теоретичні аспекти побудови рекомендаційних систем, їх класифікацію, принципи роботи та поширені проблеми, серед яких особливу увагу приділено проблемі холодного старту. Детально розглянуто основні типи рекомендаційних систем, зокрема: Collaborative, Content-based, Demographic-based, Utility-based, Knowledge-based, Hybrid. Також досліджено популярні сервіси: Spotify, Netflix, Amazon, що успішно використовують рекомендаційні системи.

На основі проведеного аналізу було реалізовано мобільний HealthTech-застосунок Dish для платформи iOS. Основною функціональністю застосунку є надання персоналізованих рекомендацій рецептів з урахуванням вподобань користувача. Для реалізації цієї можливості була використана локальна ML-модель на базі Core ML, що дозволяє здійснювати обробку даних безпосередньо на пристрої користувача, зберігаючи конфіденційність, забезпечуючи високу швидкість та роботу в офлайн режимі.

Під час розробки було використано сучасні інструменти від компанії Apple, а саме SwiftUI, Core ML, Create ML та Core Data. Застосунок було реалізовано з архітектурою MVVM, що забезпечує високу модульність, легкість до тестування та простоту підтримки коду в майбутньому.

Практичне значення отриманих результатів полягає у створенні функціонального та зручного мобільного застосунку, що може бути використаний як основа для подальших досліджень у галузі рекомендаційних систем. Також, результати даної роботи демонструють ефективність інтеграції

локальних моделей машинного навчання у мобільні застосунки, що є перспективним напрямом розвитку сучасних цифрових рішень.

У результаті виконання дипломної роботи поставлена мета була досягнута: розроблено мобільний застосунок із вбудованою системою персоналізованих рекомендацій, що відповідає сучасним вимогам до продуктивності, конфіденційності та зручності використання.

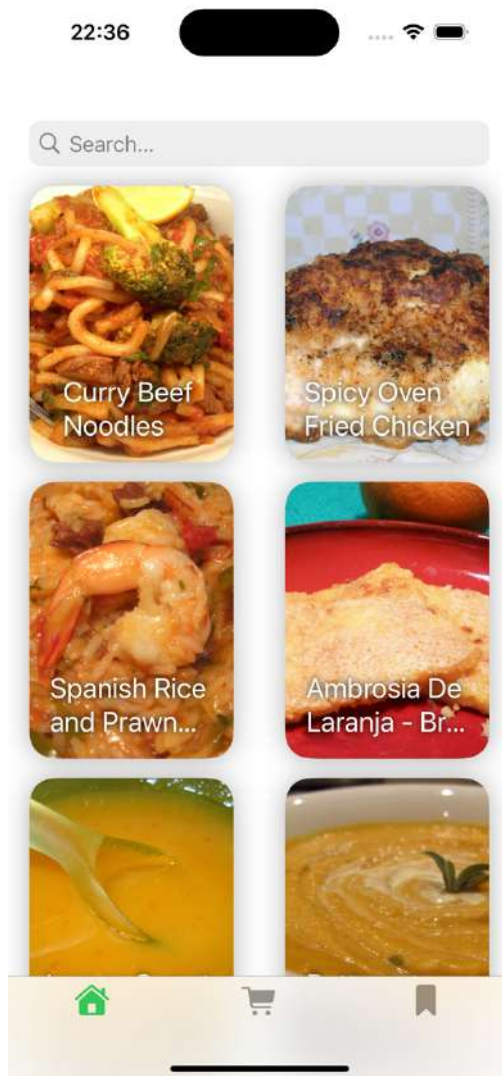
Джерела

1. Nvidia: Recommendation System - Посилання на електронний ресурс: <https://www.nvidia.com/en-us/glossary/recommendation-system/>
2. Analytics Steps: What is Collaborative Filtering? - Посилання на електронний ресурс: <https://www.analyticssteps.com/blogs/what-collaborative-filtering-types-working-and-case-study>
3. *Aditya, P. H., Budi, I., & Munajat, Q.* A Comparative Analysis of Memory-based and Model-based Collaborative Filtering on the Implementation of Recommender System for Ecommerce in Indonesia: A Case Study PT X. Faculty of Computer Science, University of Indonesia. 2021. – С. 308.
4. Medium: Memory-Based vs. Model-Based Collaborative Filtering Techniques - Посилання на електронний ресурс: <https://medium.com/@niitwork0921/memory-based-vs-model-based-collaborative-filtering-techniques-c0a7f6ec4f5f>
5. Medium: Recommender Systems - User-Based and Item-Based Collaborative Filtering - Посилання на електронний ресурс: <https://medium.com/@cfpinela/recommender-systems-user-based-and-item-based-collaborative-filtering-5d5f375a127f>
6. Geeksforgeeks: What are Recommender Systems? - Посилання на електронний ресурс: <https://www.geeksforgeeks.org/what-are-recommender-systems/>
7. The app solutions: Five Types of Recommender Systems and Their Benefits - Посилання на електронний ресурс: <https://theappsolutions.com/blog/development/recommender-systems-guide/>

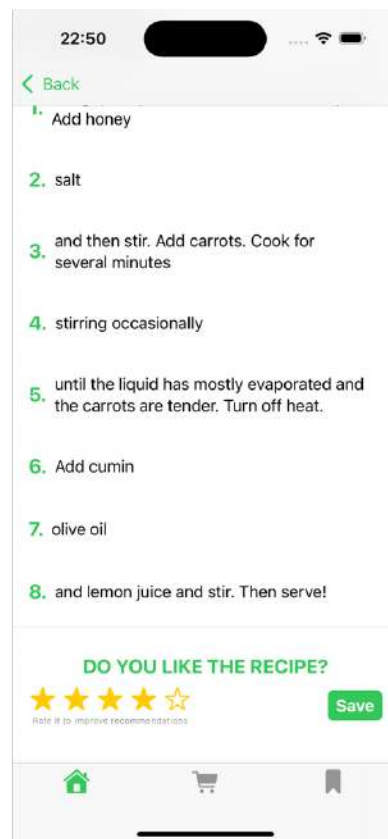
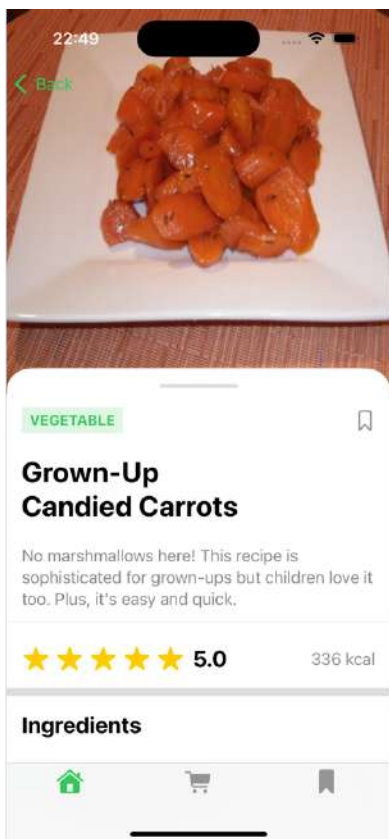
8. Medium: Knowledge-Based Recommender Systems - Посилання на електронний ресурс: <https://medium.com/@jwu2/knowledge-based-recommender-systems-an-overview-536b63721dba>
9. Берк Р. Гібридні рекомендаційні системи: Дослідження і експерименти 2002. № 12(4). С. 331–370
10. FreeCodeCamp: What is the Cold Start Problem in Recommender Systems - Посилання на електронний ресурс: <https://www.freecodecamp.org/news/cold-start-problem-in-recommender-systems/>
11. Apple Documentation: Core ML - Посилання на електронний ресурс: <https://developer.apple.com/documentation/coreml/>
12. Apple Documentation: Core Data - Посилання на електронний ресурс: <https://developer.apple.com/documentation/coredata>
13. Swift by Sundell: Discover SwiftUI - Посилання на електронний ресурс: <https://www.swiftbysundell.com/discover/swiftui/>
14. Kaggle: Food.com - Recipes and Reviews dataset - Посилання на електронний ресурс: <https://www.kaggle.com/datasets/irkaal/foodcom-recipes-and-reviews?select=reviews.csv>
15. Aubergine: MVVM Architecture and Modular Patterns for Building Scalable iOS Apps - Посилання на електронний ресурс: <https://www.aubergine.co/insights/mvvm-architecture-and-modular-pattern-for-building-scalable-ios-apps>

Додатки

Додаток А. Головний екран застосунку



Додаток Б. Екран рецепта



Додаток В. Екран зі списком покупок

