НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

До захисту допущено:

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____2025 р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Інформаційні управляючі системи та технології»
спеціальності 126 «Інформаційні системи та технології»
на тему: «Рекомендаційна система підбору спортивних тренерів»

Виконав:
Студент IV курсу, групи ІС-11
Валіваха Андрій Андрійович _____

Керівник:
Кандидат технічних наук
Попенко Володимир Дмитрович _____

Рецензент:
Доцент кафедри ІПІ, к.т.н.,
Новінський Валерій Петрович _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2025 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 126 «Інформаційні системи та технології»

Освітньо-професійна програма «Інформаційні управляючі системи та технології»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____2025 р.

ЗАВДАННЯ
на дипломний проєкт студенту
Валівасі Андрію Андрійовичу

1. Тема проєкту «Рекомендаційна система підбору спортивних тренерів», керівник проєкту Попенко Володимир Дмитрович, кандидат технічних наук, затверджені наказом по університету від «23» травня 2025 р. № 1705-с

2. Термін подання студентом проєкту: «09» червня 2025 р.

3. Вихідні дані до проєкту: мова програмування Swift, фреймворк SwiftUI, зберігання даних у базу даних Realm, архітектурний шаблон MVVM.

4. Зміст пояснювальної записки: опис предметної області, аналіз існуючих рішень, формування вимог до системи, вибір технології розроблення, розроблення інформаційної системи, математичне забезпечення, тестування системи.

5. Перелік графічного матеріалу (із зазначенням обов’язкових креслеників, плакатів, презентацій тощо): Діаграма розгортання, діаграма діяльності, діаграма компонентів, діаграма класів.

6. Дата видачі завдання 03.03.2025

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Огляд предметної області	15.03.2025	
2	Аналіз існуючих рішень	20.03.2025	
3	Формування вимог до системи	25.03.2025	
4	Вибір технології розроблення	30.03.2025	
5	Вибір засобів та методів реалізації	05.04.2025	
6	Розроблення інформаційної системи	05.05.2025	
7	Розроблення математичного забезпечення	15.05.2025	
8	Тестування системи	20.05.2025	
9	Формування висновків	25.05.2025	
10	Оформлення пояснювальної записки	30.05.2025	
11	Подання готового проєкту	09.06.2025	

Студент Андрій ВАЛІВАХА

Керівник Володимир ПОПЕНКО

АНОТАЦІЯ

Рекомендаційна система підбору спортивних тренерів.

Проєкт містить 64 с. тексту, 17 рисунків, 3 таблиці, посилання на 15 літературних джерела, додатки та 4 конструкторських документи.

СПОРТИВНИЙ ТРЕНЕР, КЛІЄНТ, РЕКОМЕНДАЦІЙНА СИСТЕМА, КОСИНУСНА СХОЖІСТЬ, БАГАТОКРИТЕРІАЛЬНИЙ АНАЛІЗ.

Об’єктом дослідження є методи пошуку і відбору фахівців на основі багатокритеріального аналізу.

Метою проєкту є автоматизація процесу пошуку найбільш відповідного спортивного тренера за заданими критеріями користувача.

У дипломному проєкті розроблено рекомендаційний сервіс, що дозволяє здійснювати підбір тренера відповідно до індивідуальних побажань користувача. Система включає клієнтську частину для взаємодії користувача з інтерфейсом, а також серверну частину, відповідальну за обробку запитів і реалізацію бізнес-логіки. Для порівняння профілів користувача і тренера використано підхід на основі векторного представлення з обчисленням косинусної схожості. Реалізовано нормалізацію даних, урахування бінарних, числових та символьних характеристик, а також географічного розташування. Значну увагу приділено побудові математичної моделі та архітектурі системи.

Отриманий рекомендаційний сервіс може бути використаний як основа для створення повноцінних платформ з підбору фахівців у сфері спорту, фітнесу або інших послуг за запитом клієнтів.

SUMMARY

Recommendation system for selecting sports coaches.

The project contains 64 pages text, 17 figures, 3 tables, links to 15 literary sources, annexes and 4 design documents.

SPORTS COACH, CLIENT, RECOMMENDATION SYSTEM, COSINE SIMILARITY, MULTICRITERIA ANALYSIS.

The object of this work is the investigation of methods for searching and selecting specialists based on multicriteria analysis.

The aim of the project is to automate the process of finding the most suitable sports coach according to the user's preferences.

The graduation project presents a recommendation service that allows selecting a coach tailored to individual user needs. The system consists of a client-side interface for interaction and a server-side module that handles business logic and request processing. The recommendation mechanism is based on vector representation of profiles and uses cosine similarity to compare user and coach attributes. The solution includes normalization of data, consideration of binary, numerical, and categorical characteristics, as well as geographic proximity. Special attention was given to the construction of the mathematical model and system architecture.

The developed recommendation service can be used as a foundation for full-featured platforms aimed at selecting professionals in sports, fitness, or other user-requested services.

№ рядка	Формат	Позначення	Найменування	Кіл. аркушів	№ екз.	Примітка
1			Документація загальна			
2						
3			Знову розроблена			
4						
5	А4	ІС11.050БАК.005 ПЗ	Рекомендаційна система	64		
6			підбору спортивних тренерів.			
7			Пояснювальна записка			
8						
9	А3	ІС11.050БАК.005 Д1	Рекомендаційна система	1		
10			підбору спортивних тренерів.			
11			Діаграма розгортання			
12						
13	А3	ІС11.050БАК.005 Д2	Рекомендаційна система	1		
14			підбору спортивних тренерів.			
15			Діаграма діяльності			
16						
17	А3	ІС11.050БАК.005 Д3	Рекомендаційна система	1		
18			підбору спортивних тренерів.			
19			Діаграма компонентів			
20						
21	А3	ІС11.050БАК.005 Д4	Рекомендаційна система	1		
22			підбору спортивних тренерів.			
23			Діаграма класів			
24						
25						
26						
27						
28						

Зм.	Аркуш	№ докум.	Підпис	Дата	ІС11.050БАК.005 ТП		
Розроб.		Валіваха А.А.			Рекомендаційна система підбору спортивних тренерів. Відомість дипломного проєкту	Літ.	Аркуш 1 / Аркушів 1
Керівн.		Попенко В.Д.					
Реценз.						КПІ ім. Ігоря Сікорського Група ІС-11	
Н. Контр.							
Затверд.							

Пояснювальна записка
до дипломного проєкту
на тему: «Рекомендаційна система підбору спортивних тренерів»

Київ – 2025 року

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

Combine — фреймворк від Apple, який дозволяє обробляти асинхронні події (наприклад, введення тексту, зміни даних, мережеві запити) у вигляді потоків даних.

MongoDB Realm Sync — технологія, яка автоматично синхронізує дані між локальною базою даних Realm у мобільному застосунку та віддаленою базою MongoDB Atlas у хмарі.

MVVM (Model-View-ViewModel) — архітектурний шаблон, що забезпечує розділення відповідальності між логікою, інтерфейсом та представленням даних.

ObservableObject — протокол у SwiftUI, який дозволяє створювати класи, дані в яких можуть "спостерігатися" інтерфейсом.

Realm (Database) — легка та швидка база даних для мобільних застосунків, яка дозволяє зберігати об’єкти прямо у вигляді Swift- або Kotlin-класів. Вона працює офлайн, підтримує синхронізацію з сервером (через Realm Sync), і часто використовується замість SQLite або Core Data через простоту використання.

SwiftUI — фреймворк від Apple для створення інтерфейсів у застосунках на Swift. Він дозволяє описувати UI простим декларативним способом — тобто вказуєш, що має бути на екрані, а не як саме це реалізувати.

UAT (User Acceptance Testing) — фінальний етап тестування, коли реальні користувачі або замовники перевіряють, чи відповідає система їхнім вимогам і очікуванням.

UI (User Interface) — зовнішній вигляд і елементи, з якими взаємодіє користувач: кнопки, меню, шрифти, кольори тощо.

UX (User Experience) — загальний досвід користувача від взаємодії із застосунком: наскільки все зручно, зрозуміло та приємно працює.

ВСТУП

Індустрія спортивних послуг динамічно розвивається у відповідь на зростання попиту на персоналізовані тренування, фізичний розвиток і підтримку здорового способу життя. У зв’язку з цим все більшої актуальності набуває питання ефективного підбору спортивного тренера, який би відповідав індивідуальним потребам, можливостям та очікуванням клієнта. Наявність великої кількості тренерів на ринку, відмінності у кваліфікації, досвіді, вартості послуг і спеціалізаціях створюють складнощі при виборі найбільш підходящого кандидата.

У сучасних умовах розвитку цифрових технологій усе більш популярними стають онлайн-інструменти для автоматизації процесів вибору та прийняття рішень. Це відкриває нові можливості для впровадження рекомендаційних систем, які, базуючись на багатокритеріальному аналізі, здатні запропонувати найбільш релевантного тренера згідно з уподобаннями користувача. Важливо, щоб такий підбір був не лише точним, а й прозорим, обґрунтованим, простим у використанні. Завдяки інтеграції таких систем у мобільні та веб-застосунки, користувачі отримують персоналізований сервіс у зручному форматі. Крім того, автоматизовані підходи дозволяють враховувати змінні вподобання користувача в динаміці. Це підвищує рівень задоволеності клієнтів та сприяє довготривалому використанню платформи. Саме тому питання ефективності алгоритмів добору стає ключовим під час проєктування таких систем.

Процес вибору тренера зазвичай базується на кількох ключових параметрах: досвід, рейтинг, ціна, відгуки, місцезнаходження тощо. У ручному режимі врахування всіх цих характеристик займає багато часу та не завжди дає об’єктивний результат. Тому з’являється потреба в створенні інформаційної системи, яка дозволить структурувати дані, нормалізувати характеристики та здійснити ефективне порівняння кандидатів. Для досягнення цього важливо застосовувати математичні методи, здатні забезпечити об’єктивність та відтворюваність результатів. У результаті формується інтелектуальна система підтримки рішень, що поєднує гнучкість, точність і зручність.

Проблематика полягає в неоднорідності даних, які впливають на прийняття рішення, адже деякі з них є числовими (наприклад, рейтинг), інші — символьними (тип тренувань) чи географічними (місце проведення занять). Без системного підходу до аналізу таких характеристик неможливо досягти об’єктивності в рекомендаціях. Саме тому доцільним є застосування математичних моделей, які дозволяють зіставити вектори характеристик користувача та тренера та визначити ступінь відповідності між ними.

Об’єктом дослідження у межах дипломного проєкту є методи прийняття рішень за допомогою багатокритеріального аналізу.

Предметом дослідження виступають програмні засоби, що реалізують процес підбору спортивного тренера з використанням алгоритмів векторного порівняння на основі косинусної схожості.

Метою проєкту є створення системи, здатної автоматизувати процес підбору тренера та адаптуватися до потреб кожного окремого користувача. Для досягнення поставленої мети передбачено вирішення наступних задач:

– аналіз існуючих систем і підходів до персоналізованого підбору;

– формалізація критеріїв вибору спортивного тренера;

– побудова математичної моделі векторного представлення профілів;

– нормалізація характеристик різних типів та їх об’єднання в єдину систему оцінки;

– реалізація алгоритму обчислення косинусної схожості для порівняння профілів;

– створення клієнтської та серверної частин застосунку.

Запропонована система дозволяє оптимізувати підбір тренера, мінімізувати вплив суб’єктивного фактору, підвищити точність рекомендацій і зробити взаємодію між користувачем і тренером зручною та ефективною.

Дипломний проєкт складається з наступних розділів: вступ, основні розділи, список використаних джерел із 15 найменувань, 1 додаток. Графічна частина включає 4 кресленики формату А3. Загальний обсяг 64 сторінки.

1 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Загальні положення

У сучасних умовах цифровізації велика кількість соціальних, освітніх та оздоровчих процесів переноситься у віртуальне середовище. Серед них особливої актуальності набуває організація індивідуального тренінгу, адже зростає попит на послуги спортивних тренерів, здатних працювати як очно, так і онлайн. При цьому користувачі, які шукають тренера, стикаються з низкою труднощів: відсутністю централізованої платформи, недостатньою кількістю інформації про кандидатів, надлишком варіантів без можливості їх ефективного порівняння.

Чинна модель пошуку, яка спирається на соціальні мережі або загальні дошки оголошень, не гарантує відповідності очікуванням користувача. Навпаки, процес вибору часто є хаотичним, інтуїтивним і не завжди успішним. Нерідко виникає ситуація, коли клієнт і тренер витрачають час на взаємодію, не будучи сумісними за базовими критеріями: віком, місцем занять, досвідом, мовою, методикою роботи чи спеціалізацією.

Автоматизована система, яка здатна збирати профілі обох сторін, зіставляти їх на основі заданих характеристик і рекомендувати найбільш релевантні варіанти, дозволяє значно скоротити час на пошук, підвищити якість вибору та загальний комфорт взаємодії.

Завдяки використанню математичних методів аналізу подібності (зокрема косинусної схожості), така система здатна оцінювати відповідність профілів та автоматично сортувати варіанти для користувача.

Функціональні переваги:

– централізоване зберігання профілів;

– релевантний пошук на основі заданих критеріїв;

– зворотна комунікація через підтвердження запитів;

– захист персональних даних до моменту згоди;

– підвищення якості взаємодії між сторонами.

1.2 Об’єкт проектування

Ключова проблема полягає у відсутності спеціалізованої системи, яка забезпечує ефективний, автоматизований, персоналізований підбір спортивного тренера. Існуючі сервіси здебільшого надають лише каталог без глибокого аналізу відповідності. Це не дозволяє користувачеві швидко визначити найбільш підходящого кандидата.

Крім того, комунікація у багатьох випадках розпочинається без чіткої впевненості в релевантності сторін, що призводить до втрати часу, відмов після перших контактів та зниження ефективності пошуку. Аналіз реальних сценаріїв свідчить про необхідність впровадження алгоритмічного механізму, який виконує попередню обробку, порівняння та фільтрацію.

Для цього необхідно створити інструмент, який враховуватиме усі основні параметри, важливі для прийняття рішення користувачем, і зіставлятиме їх з характеристиками тренерів, що вже присутні в системі.

Основні задачі:

– створення структур профілів тренера та клієнта;

– впровадження алгоритму схожості;

– реалізація механізму запитів з підтвердженням;

– захист особистих даних до моменту згоди;

– інтеграція з базою даних.

1.3 Призначення системи

Система створюється як мобільний застосунок для macOS/iOS, призначений для організації ефективної взаємодії між двома категоріями користувачів: тренерами та клієнтами. На відміну від каталогів і оголошень, вона пропонує рекомендаційну модель, яка відбирає кандидатів автоматично на основі аналізу профілей користувачів та тренерів.

Принцип роботи застосунку нагадує системи підбору контенту у стримінгових сервісах або товарів в e-commerce, але адаптований до особливостей людських характеристик. Це дозволяє суттєво знизити поріг входу в процес, особливо для нових користувачів, які ще не мають досвіду у виборі тренера.

Система також дозволяє тренерам самостійно керувати своєю репутацією, публікацією особистої інформації та контролювати запити на співпрацю. Уся взаємодія відбувається в безпечному та зручному середовищі.

Цільова аудиторія:

– користувачі, які шукають тренера або хочуть змінити поточного;

– тренери, які прагнуть розширити базу клієнтів та отримати релевантні запити.

Основне призначення:

– автоматизований добір тренера за параметрами;

– підтримка приватної взаємодії до моменту підтвердження;

– зручний профіль і рейтингова система.

1.4 Цілі та задачі розробки

Головна мета полягає у створенні зручного, інтелектуального застосунку, який надає користувачеві можливість швидко і безпомилково підібрати тренера, що відповідає його критеріям. Система повинна бути інтуїтивно зрозумілою, захищеною та гнучкою в розширенні.

Розробка має забезпечити повний цикл:

– створення профілю;

– пошук тренера;

– фільтрація;

– ініціація контакту;

– прийняття рішення.

Особливий акцент робиться на алгоритмі підбору, який повинен показувати лише тих тренерів, які максимально відповідають очікуванням користувача — це значно підвищує конверсію в успішну співпрацю.

Ключові задачі:

– архітектура профілів і бази даних;

– інтеграція алгоритму косинусної схожості;

– проєктування взаємодії тренер-користувач;

– реалізація сучасного UI/UX;

– створення механізму запитів і відповідей;

– оптимізація збереження та тестування системи.

Висновок до розділу 1

У даному розділі було розглянуто предметну область, що охоплює процес підбору спортивних тренерів відповідно до індивідуальних потреб користувача. Було проаналізовано поточний стан ринку тренерських послуг, виявлено ключові труднощі в комунікації між клієнтами та тренерами, а також окреслено недоліки традиційних методів взаємодії.

Сформульовано основні задачі, що стоять перед розробником сучасної інформаційної системи: автоматизація процесу підбору, створення релевантного інтерфейсу, захист персональних даних, реалізація механізму запитів та впровадження ефективного алгоритму рекомендацій.

Також було визначено цільову аудиторію та функціональне призначення системи, яке включає в себе не лише підбір тренера, а й формування цифрової платформи для якісної взаємодії сторін. У результаті проведеного аналізу створено обґрунтовану основу для наступного етапу роботи — аналізу існуючих аналогів і рішень, що дозволить сформувати чіткі технічні вимоги до майбутнього застосунку.

2 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

Перед створенням нової інформаційної системи доцільно вивчити вже наявні рішення, які частково виконують аналогічні функції. Це дозволяє не лише врахувати вдалі реалізації в інтерфейсі чи архітектурі, а й усвідомити поточні недоліки присутніх на ринку продуктів, які можна покрити у власному проєкті. У межах цього розділу було проаналізовано три застосунки — Workfit, Mixsport та Apollo.Next. Всі вони мають різний фокус, функціональність та модель взаємодії з користувачами.

2.1 Програмне рішення «Workfit»

Workfit — це український мобільний застосунок, який позиціонується як платформа для пошуку фахівців у сфері здоров’я, спорту, фітнесу, масажу, реабілітації. Його мета — надати користувачеві швидкий доступ до спеціаліста за обраними параметрами. Сервіс реалізує базовий механізм сортування та відображення профілів тренерів і суміжних фахівців з можливістю прямого контакту. При цьому система не надає рекомендаційного механізму або логіки сумісності між профілями користувача і тренера — вибір здійснюється вручну на основі візуального перегляду карток.

Інтерфейс Workfit орієнтований на простоту: користувачі швидко отримують список доступних тренерів у місті, можуть переглянути рейтинг, фото, короткий опис та перейти до зв’язку. Проте за зовнішньою зручністю приховані функціональні обмеження, які суттєво знижують ефективність у разі, коли потрібен саме персоналізований підбір. Контактна інформація відкривається одразу, а сам пошук здійснюється без аналізу індивідуальних параметрів клієнта. Таким чином, Workfit більше нагадує "каталог спеціалістів", ніж інтелектуальну систему підбору.

Приклад тестового сценарію використання застосунку Workfit наведено на рисунку 2.1.

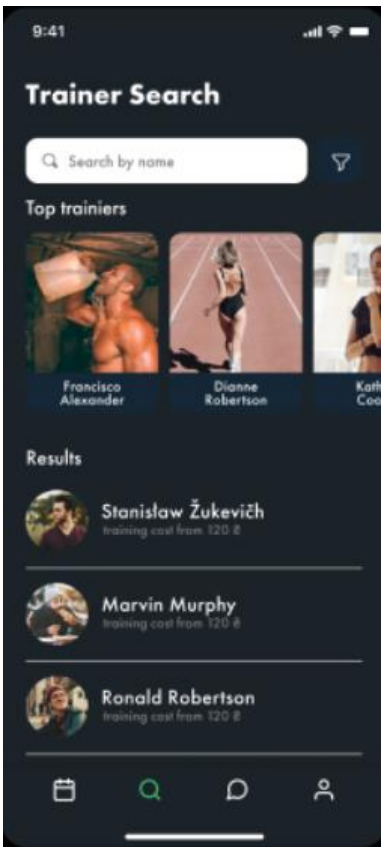

Рисунок 2.1 — Приклад використання застосунку Workfit

Переваги:

– велика кількість фахівців з різних напрямів;

– простий і доступний інтерфейс;

– система оцінювання й відгуків.

Недоліки:

– відсутність автоматизованого підбору тренера за характеристиками користувача;

– відкритість контактних даних без згоди тренера на їх розповсюдження;

– обмежена деталізація профілів, неможливість фільтрувати за складними критеріями.

2.2 Програмне рішення «Mixsport»

Mixsport — це українська онлайн-платформа, орієнтована на популяризацію спорту, фізичної активності та здорового способу життя. Вона містить інформацію

про спортивні події, секції, тренерів, клуби та спортивні об’єкти в містах України. Основна функція сервісу полягає не в підборі тренера як такого, а в інформаційному супроводі користувача, що прагне долучитися до спортивної активності — шляхом ознайомлення з можливими варіантами.

Mixsport працює як "спортивний гід" по місту: користувач може знайти спортивну подію, дізнатися про локацію, записатися в секцію або переглянути короткий профіль тренера, який веде певну групу. При цьому, система не здійснює персонального добору фахівця для користувача та не має механізмів порівняння чи зіставлення інтересів, очікувань і профілю клієнта. Застосунок також не передбачає закритої комунікації — уся взаємодія з тренером або клубом відбувається поза межами платформи.

Таким чином, Mixsport — це більше спортивний навігатор, ніж рекомендаційна система. Його роль — надати широкий контекст і зацікавити активністю, а не оптимізувати пошук персонального тренера. Приклад тестового сценарію використання застосунку Mixsport наведено на рисунку 2.2.

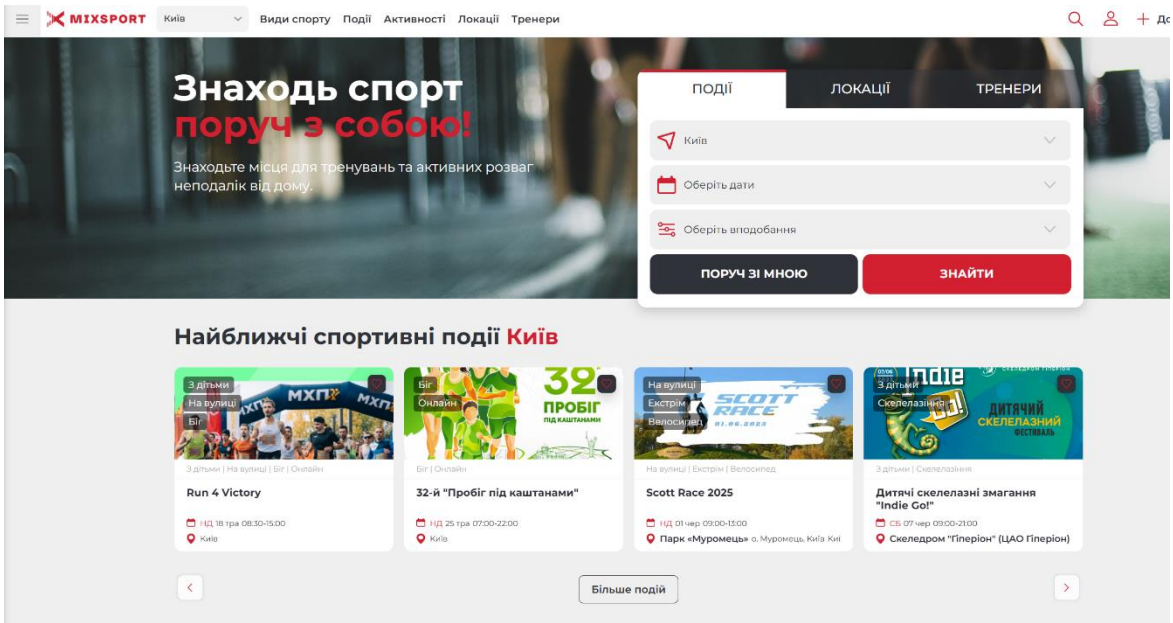

Рисунок 2.2 — Приклад основної сторінки застосунку Mixsport

На рисунку 2.3 зображено приклад карток тренерів у застосунку Mixsport, де користувач може ознайомитися з видом спорту, рейтингом, кількістю відгуків, місцем проведення тренувань та контактною інформацією кожного тренера.

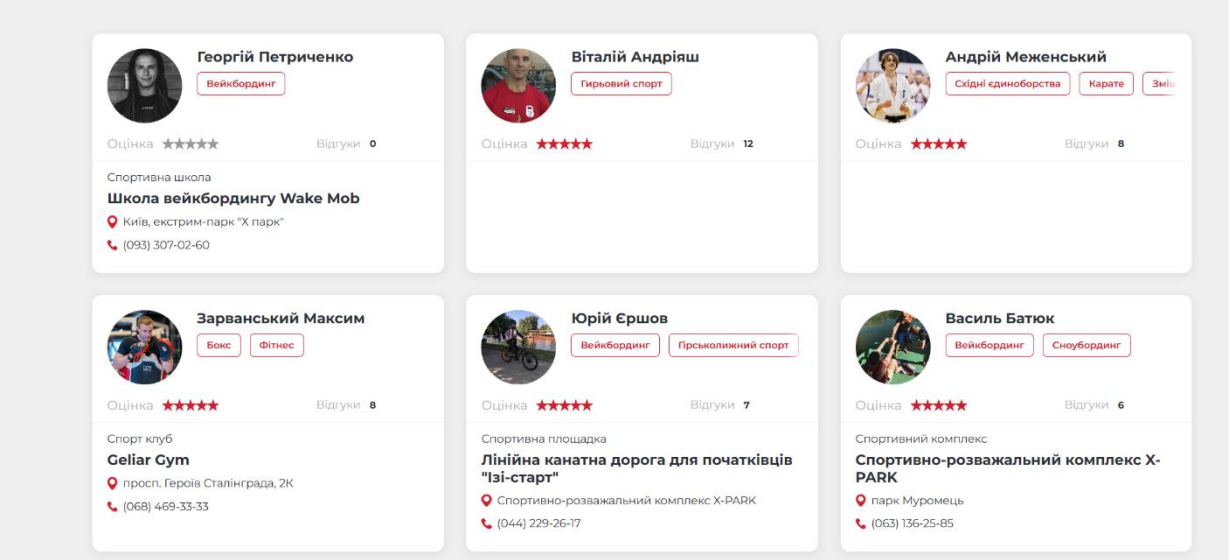

Рисунок 2.3 — Приклад карток тренерів у застосунку Mixsport

Переваги:

– інформаційне охоплення подій, клубів, тренерів і секцій;

– орієнтація на спортивні спільноти, а не тільки на індивідуальні тренування;

– календар подій і новини зі світу спорту.

Недоліки:

– відсутність підбору тренера за критеріями користувача;

– немає особистих рекомендацій чи рейтингових списків;

– профілі тренерів часто обмежені та не структуровані;

– комунікація з фахівцем відбувається поза межами платформи.

2.3 Програмне рішення «Apollo.Next»

Apollo.Next — це фірмовий мобільний застосунок, розроблений спеціально для мережі фітнес-клубів Apollo. Він є прикладом закритої екосистеми, у якій вся функціональність спрямована на полегшення взаємодії існуючих клієнтів клубу з

його сервісами: бронювання групових занять, персональні тренування, перегляд розкладу, оплата абонементів, отримання сповіщень тощо.

Ключовою перевагою Apollo.Next є глибока інтеграція з внутрішніми процесами клубу. Користувачі можуть у реальному часі переглядати наявність вільних тренувань, записуватись до конкретних тренерів, бачити зміни в розкладі, зберігати історію відвідувань, слідкувати за своїм прогресом. Усе це створює комфортну цифрову оболонку для клієнта, але водночас значно обмежує можливість пошуку тренера поза межами клубу.

Apollo.Next не орієнтований на вирішення задачі первинного підбору фахівця — він не здійснює аналіз профілю користувача, не фільтрує тренерів за індивідуальними критеріями та не реалізує алгоритм рекомендацій. Тренери вже «прив'язані» до локацій, а користувач самостійно обирає того тренера, хто працює у вибраному клубі. Сама комунікація є відкритою — персональні дані тренерів доступні одразу, без підтвердження з обох сторін. Таким чином, застосунок має перевагу у внутрішній організації, але не виконує функцій персонального добору чи конфіденційної взаємодії. Приклад тестового сценарію використання застосунку Apollo.Next наведено на рисунку 2.4.

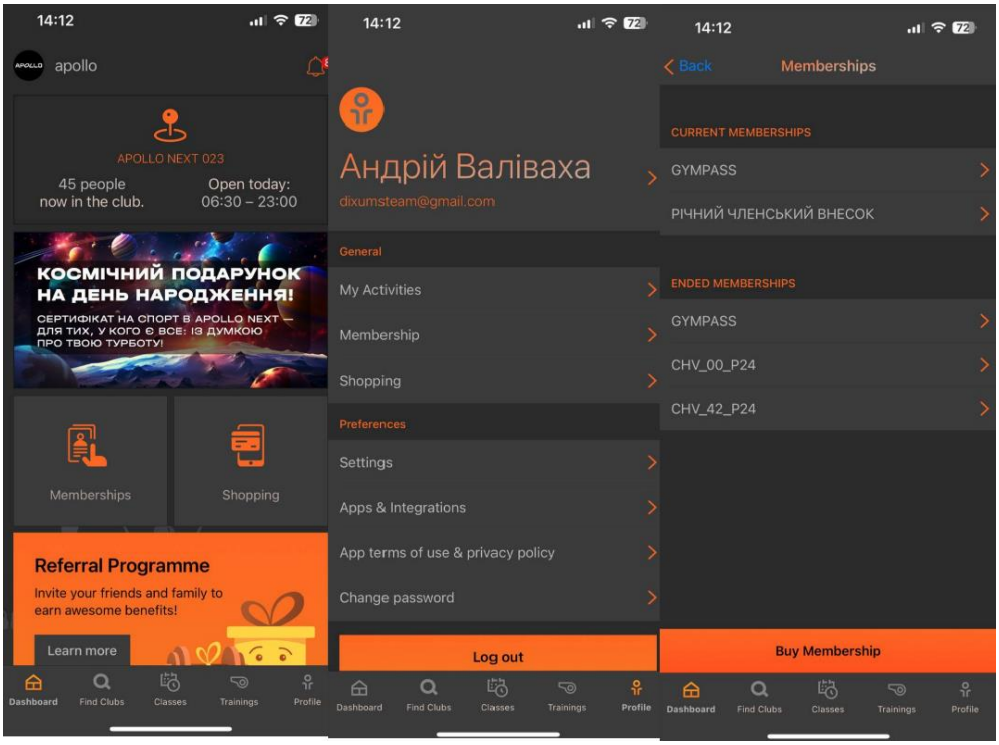

Рисунок 2.4 — Приклад використання застосунку Apollo.Next

Переваги:

– повна інтеграція з розкладом і послугами клубу Apollo;

– зручність запису на групові та індивідуальні тренування;

– ведення історії активності користувача;

– проста оплата й управління абонементами.

Недоліки:

– працює лише в межах мережі Apollo, не охоплює зовнішніх фахівців;

– відсутній персоналізований підбір тренера за параметрами користувача;

– контактна інформація тренерів відкрита для всіх клієнтів;

– система не підтримує рейтингів, фільтрів або рекомендацій.

У таблиці 2.1 представлено результати порівняльного аналізу функціональних можливостей даних застосунків для пошуку спортивних тренерів. Як видно з таблиці, жоден із розглянутих сервісів не реалізує повного набору функцій, необхідних для персоналізованого, безпечного та зручного підбору тренера. Натомість розроблювана система охоплює всі ключові критерії, включаючи аналіз профілю користувача, приховування персональних даних та двостороннє підтвердження контактів, що свідчить про її перевагу серед аналогів.

Таблиця 2.1 — Результати аналізу застосунків

Критерій/Застосунок	Workfit	Mixsport	Apollo.Next	Розроблювана система
Каталог фахівців	+	+	-	+
Рейтинг/відгуки	+	-	-	+
Геолокаційний пошук	+	+	+	+
Персоналізований підбір тренера	-	-	-	+
Аналіз профілю користувача	-	-	-	+
Приховання персональних даних	-	-	-	+

Критерій/Застосунок	Workfit	Mixsport	Apollo.Next	Розроблювана система
Запити з обопільним підтвердженням	-	-	-	+
Робота поза мережею	-	-	-	+
Прив’язка до фітнес-клубу	-	-	+	-
Підтримка зовнішніх тренерів	+	+	-	+

Висновок до розділу 2

Аналіз існуючих застосунків у сфері підбору спортивних тренерів засвідчив, що Workfit, Mixsport та Apollo.Next виконують окремі завдання в екосистемі фітнес-послуг, однак жоден із них не забезпечує повноцінного персоналізованого підбору тренера. Workfit слугує як довідник фахівців, але не враховує індивідуальні особливості користувача; Mixsport орієнтований переважно на інформування, а не на взаємодію між клієнтом і тренером; Apollo.Next обмежений у функціональності й пристосований лише до внутрішніх потреб клубу. У цьому контексті розроблювана система вирізняється комплексним підходом: вона поєднує персоналізований аналіз профілю, алгоритмічний добір на основі косинусної схожості, конфіденційність персональних даних до моменту згоди обох сторін і можливість роботи в офлайн-режимі. Така інтеграція функцій дозволяє створити рішення, яке заповнює прогалини наявних сервісів та відповідає актуальним потребам користувачів. Отже, запропонований застосунок не лише охоплює широкий спектр функцій, відсутніх у конкурентів, але й формує новий підхід до взаємодії між користувачем і тренером. Завдяки цьому система може адаптуватися до різних сценаріїв використання та забезпечувати високу точність рекомендацій. Такий функціональний набір дозволяє позиціонувати розроблену систему як інноваційне рішення на ринку цифрових фітнес-послуг.

3 ФОРМУВАННЯ ВИМОГ ДО СИСТЕМИ

Створення сучасного інформаційного застосунку потребує не лише ефективної реалізації програмного коду, а й попереднього глибокого аналізу вимог, які визначають функціональні, структурні, технічні й експлуатаційні особливості майбутньої системи. Вимоги слугують основою для формування архітектури, вибору технологій, структури бази даних, логіки взаємодії між користувачем і системою, а також впливають на якість кінцевого продукту.

Для рекомендаційної системи підбору спортивного тренера, яка має забезпечувати ефективну взаємодію між клієнтами та тренерами, особливо важливою є інтеграція алгоритмів персоналізованого підбору, конфіденційного обміну інформацією та адаптивного інтерфейсу. У цьому розділі сформовано три ключові блоки вимог: загальні вимоги до системи, функціональні характеристики та вимоги до забезпечення (програмного, технічного, інформаційного та організаційного).

3.1 Вимоги до системи в цілому

Загальні вимоги визначають фундаментальні принципи, яким має відповідати розроблюване програмне забезпечення, незалежно від його функціонального наповнення. Система повинна бути надійною, масштабованою, розширюваною та придатною до адаптації у майбутньому. Продукт розробляється як застосунок для macOS/iOS, з використанням мови програмування Swift [1] та фреймворку SwiftUI [2].

Ключовим є забезпечення зручного процесу взаємодії між двома категоріями користувачів: тренерами та клієнтами. Система має передбачати незалежну реєстрацію обох сторін, із подальшим розгалуженням функціональності в залежності від типу профілю.

Система повинна функціонувати стабільно за умови як наявності підключення до мережі Інтернет, так і в автономному режимі. Такий підхід

особливо актуальний у випадках, коли користувач перебуває у дорозі або в місцевості з обмеженим покриттям. Для цього всі основні дані повинні зберігатися у базі даних Realm [3] з можливістю масштабування у MongoDB Realm Sync [4]. Діаграму розгортання наведено на кресленику ІС11.050БАК.005 Д1.

Крім того, важливо забезпечити конфіденційність взаємодії. Всі персональні дані (ім’я, прізвище, email) мають бути приховані доти, доки обидві сторони не погодяться на відкриття контакту. Це підвищує безпеку користувача та відповідає вимогам етичної взаємодії у цифровому середовищі.

3.2 Вимоги до функціональних характеристик

Функціональні вимоги визначають набір дій, які система повинна виконувати для задоволення потреб користувача. Вони охоплюють ключову бізнес-логіку: від реєстрації й заповнення профілю до отримання рекомендацій та комунікації з тренером.

Одним із найважливіших компонентів системи є алгоритм підбору тренера. Він має використовувати принцип косинусної схожості для обчислення відповідності між параметрами користувача та тренера. Для цього характеристики обох сторін подаються у вигляді векторів, що включають такі ознаки:

– вік;

– очікуваний досвід тренера;

– рейтинг;

– ціна за заняття;

– категорії спорту;

– наявність сертифікатів;

– мови спілкування;

– район проживання або діяльності;

– специфіка тренувань (робота з дітьми, групові заняття тощо).

На основі зіставлення цих параметрів система формує рейтинг відповідності тренерів для кожного користувача. Результати подаються у вигляді списку, відсортованого за рівнем схожості.

Також повинна бути реалізована двостороння система контакт-запитів, що гарантує взаємну згоду перед початком відкритого спілкування. Користувач надсилає запит до тренера, після чого останній може або підтвердити, або відхилити його. Лише у разі підтвердження обидві сторони отримують доступ до особистої інформації.

Функціонально застосунок повинен підтримувати:

– реєстрацію та авторизацію;

– редагування профілю з фото;

– перегляд рекомендаційних списків;

– надсилання/отримання запитів;

– приховування/відкриття контактних даних;

– перегляд статусу запитів;

– зберігання історії взаємодій.

На кресленику ІС11.050БАК.005 Д2 представлено діаграму діяльності, яка відображає послідовність дій користувача в системі підбору спортивного тренера — від авторизації до отримання рекомендацій та встановлення контакту з обраним фахівцем.

3.3 Вимоги до видів забезпечення

Для забезпечення стабільної та безперебійної роботи системи необхідно передбачити відповідні ресурси, що охоплюють програмне, технічне, інформаційне та організаційне забезпечення.

Система має бути реалізована мовою Swift з використанням SwiftUI як основного фреймворку для побудови інтерфейсу. Обробка стану та взаємодії між екранами виконується через Combine [5] або аналогічні механізми. Збереження

даних реалізується через Realm — вбудовану мобільну базу даних, яка дозволяє зберігати об’єкти в офлайн-режимі.

У перспективі можлива інтеграція з хмарними технологіями — наприклад, MongoDB Realm Sync для аналітики, авторизації чи збереження резервних копій.

Застосунок має бути сумісним з пристроями Apple: iPhone, iPad, MacBook. Мінімальні вимоги:

– операційна система: macOS 12.0+ / iOS 15.0+;

– оперативна пам’ять: не менше 4 ГБ;

– вільна пам’ять: 100–150 МБ;

– підтримка камери (для завантаження фото профілю);

– інформаційне забезпечення.

До складу інформаційного забезпечення входять:

– база профілів користувачів і тренерів;

– параметри для алгоритму відповідності;

– вагові коефіцієнти для кожної ознаки;

– список запитів і статусів;

Організаційна структура системи передбачає розмежування прав доступу:

– клієнт має доступ лише до загальної інформації про тренера до моменту підтвердження зі сторони тренера при зацікавленості;

– тренер бачить запити, але сам вирішує — відкривати контакт чи ні.

Також має бути передбачено модулювання майбутнього масштабування — можливість додавання нових ролей, фільтрів, показників рейтингу, а також автоматичне оновлення застосунку.

Висновок до розділу 3

Таким чином, сформовані вимоги стали надійною основою для подальшого проєктування системи. Вони забезпечують цілісне бачення функціоналу та технічної реалізації застосунку.

4 ВИБІР ТЕХНОЛОГІЇ РОЗРОБЛЕННЯ

Одним із найважливіших етапів реалізації будь-якого програмного забезпечення є обґрунтований вибір технологій. Успішний проєкт неможливо реалізувати без продуманого підходу до архітектури, мови програмування, бібліотек, баз даних та інструментів, що забезпечують інтеграцію та масштабування. Особливо це актуально для застосунку, який виконує роль рекомендаційної системи, взаємодіє з користувачами, опрацьовує та аналізує множину параметрів у реальному часі.

У цьому розділі буде детально розглянуто набір технологій, які були обрані для реалізації системи підбору спортивного тренера. Кожне рішення підкріплене практичними міркуваннями щодо ефективності, відповідності предметній області, а також перспективи майбутнього розвитку продукту.

4.1 Вибір архітектурного шаблону програмного забезпечення

Застосунок реалізований із використанням архітектурного патерна MVVM (Model–View–ViewModel) [6], який є одним із найбільш рекомендованих підходів у розробці сучасних застосунків на SwiftUI. Його основна ідея полягає у відокремленні логіки представлення інтерфейсу (View) від бізнес-логіки (ViewModel) та моделей даних (Model). Це дозволяє зробити код більш структурованим, легшим для тестування та масштабування.

MVVM забезпечує логічне розділення відповідальностей між компонентами. View зосереджена виключно на відображенні даних, тоді як ViewModel відповідає за їх підготовку, обробку подій та взаємодію з локальною базою. Такий підхід дозволяє змінювати бізнес-логіку, не змінюючи інтерфейс, і навпаки.

Архітектурну побудову застосунку на основі шаблону MVVM представлено на кресленику ІС11.050БАК.005 Д3.

Архітектура MVVM також органічно підтримує реактивне програмування, яке реалізується через Combine — фреймворк, що дозволяє зв’язувати компоненти

даних і переглядів у реальному часі. Завдяки цьому система миттєво реагує на зміни у базі даних або вхідні події користувача, без ручного оновлення станів.

4.2 Технології інтерфейсу та логіки

Основною мовою програмування обрано Swift, яка є нативною для екосистеми Apple. Swift відзначається високою продуктивністю, статичною типізацією та безпечністю, що робить її оптимальним вибором для реалізації мобільного клієнта. Особливо це важливо для застосунку, який працює з персональними даними та потребує стабільної роботи на macOS та iOS.

Для реалізації інтерфейсу використовується SwiftUI — декларативний фреймворк, що дозволяє будувати інтерфейс за допомогою простих та гнучких конструкцій. Однією з основних переваг SwiftUI є автоматичне оновлення вмісту перегляду у відповідь на зміну стану.

SwiftUI дозволяє будувати адаптивні інтерфейси, які працюють на різних пристроях (iPhone, iPad, Mac), з мінімальними змінами в коді. Це особливо зручно для проєктів, які в майбутньому можуть отримати кросплатформене розширення.

4.3 Управління даними

Для зберігання даних користувачів, тренерів та запитів на контакт було обрано Realm — легку та високопродуктивну мобільну базу даних. На відміну від традиційних SQL-підходів, Realm працює з об’єктами, що дозволяє розробникам зосередитись на логіці замість синтаксису запитів. Такий підхід значно пришвидшує розробку та знижує ймовірність помилок при роботі з даними. База даних підтримує складені зв’язки між об’єктами, що дає змогу ефективно структурувати інформацію без надмірної складності. Це особливо зручно для реалізації взаємодії між користувачем і тренером через механізм контакт-запитів.

Realm забезпечує збереження всієї необхідної інформації в локальному середовищі, без потреби в підключенні до віддаленого сервера. Це особливо

важливо для забезпечення офлайн-доступу, який необхідний для застосунку, що працює в умовах непостійного з'єднання з мережею. Завдяки внутрішній системі транзакцій, база даних гарантує цілісність і надійність даних навіть у випадку раптового завершення сесії або збою живлення. У разі потреби синхронізації з хмарними сервісами Realm також пропонує можливості масштабування через Realm Sync, що може бути корисно при подальшому розвитку проєкту.

4.4 Перспективи масштабування і розширення

Архітектура і вибрані технології дозволяють легко інтегрувати сторонні сервіси, реалізовувати нові функції та адаптувати логіку для роботи з бекендом або веб-інтерфейсом.

Зокрема, можлива інтеграція з MongoDB Realm Sync для авторизації, аналітики, хмарного збереження профілів та push-сповіщень. Крім того, алгоритмічна частина (косинусна схожість [7]) реалізована як ізольований компонент, що легко переноситься у вигляді API-сервісу на Node.js чи Python.

Також передбачена можливість:

– реалізації чату між користувачами;

– розширення бази на інші сфери (наприклад, психологи, дієтологи);

– впровадження платної підписки, аналітики прогресу тощо.

Висновок до розділу 4

Обраний стек технологій дозволив досягти балансу між надійністю, продуктивністю, естетикою та перспективою масштабування. Swift, SwiftUI і Combine формують стабільне середовище для розробки клієнтської частини з приємним інтерфейсом. Архітектура MVVM забезпечує чисту логіку та структурованість, а Realm дозволяє організувати ефективну й гнучку роботу з локальними даними.

5 РОЗРОБЛЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

5.1 Структура системи

Інформаційна система, створена в рамках виробничої практики, є прикладом кросплатформеного застосунку, що функціонує на базі macOS та iOS. Основна мета системи полягає у забезпеченні користувачів можливістю зручно та ефективно знаходити спортивних тренерів відповідно до особистих потреб, а також взаємодіяти з ними у рамках безпечної, персоналізованої платформи. Структура системи побудована таким чином, щоб забезпечити максимальну гнучкість, масштабованість та зрозумілість архітектурних рішень.

Застосунок умовно можна поділити на декілька функціональних рівнів: рівень користувацького інтерфейсу, рівень обробки бізнес-логіки та рівень збереження даних. Всі ці компоненти реалізовано в межах однієї програмної оболонки, без використання зовнішнього серверного середовища, що значно спрощує реалізацію та оптимізує роботу в офлайн-режимі. Користувачі мають змогу створювати профілі, редагувати особисті дані, застосовувати фільтрацію для пошуку тренерів, надсилати запити на зв'язок, отримувати відповіді та залишати оцінки після співпраці.

Окрему увагу було приділено реалізації обробки особистих даних. Система побудована з урахуванням конфіденційності: ім’я, прізвище та контактні дані тренера не відображаються, поки обидві сторони не підтвердять запит на зв’язок. Такий підхід сприяє безпеці та формуванню довірливого середовища взаємодії.

Таким чином, структура інформаційної системи побудована за принципами простоти, зручності та безпеки, що відповідає сучасним вимогам до програмних продуктів персоналізованої взаємодії.

Крім основних функціональних можливостей, система реалізує інтелектуальний механізм рекомендації тренерів, що базується на алгоритмі косинусної схожості між вектором уподобань користувача та профілями тренерів. Такий підхід дозволяє враховувати індивідуальні переваги користувача, включаючи обрані спортивні напрямки, бажану локацію, бюджет, мову

спілкування та інші параметри. Результатом є персоналізована видача, яка підвищує ймовірність успішного підбору тренера з першого пошуку. Важливо, що всі параметри враховуються із відповідними ваговими коефіцієнтами, що дає змогу системі розуміти, які критерії є ключовими для конкретного користувача.

З технічної точки зору, система побудована з використанням мови програмування Swift та фреймворку SwiftUI, що забезпечує сучасний вигляд інтерфейсу та високу продуктивність. Для збереження даних використано локальну базу Realm, яка дозволяє працювати зі структурованими об'єктами, швидко здійснювати пошук та змінювати записи без потреби у складних SQL-запитах. У перспективі передбачена можливість інтеграції з хмарними сервісами для синхронізації даних між пристроями, що розширить функціональність платформи та зробить її ще більш зручною для користувачів.

5.2 Файлова структура застосунку

Файлова організація застосунку реалізована відповідно до принципів функціонального поділу, що дозволяє підтримувати чисту архітектуру, модульність і легку навігацію під час розробки. Нижче наведено деталізований опис основних груп файлів, що реалізують ключові частини системи.

Моделі даних (Models/) представляють сутності системи та використовуються для збереження інформації у локальній базі Realm. Сюди належать:

– User.swift, Trainer.swift — описують структури клієнтів і тренерів;

– ContactRequest.swift — моделює запити на контакт;

– TrainerRating.swift, Review.swift — відповідають за реалізацію системи оцінювання;

– District.swift, SportCategory.swift — допоміжні енумерації для категоризації.

Інтерфейс користувача (Views/) побудовано за принципами SwiftUI та логічно поділено на підрозділи:

– Auth/ — реєстрація та вхід (RegisterUserView, LoginView, AuthRootView);

– Profile/ — профілі тренерів та користувачів (TrainerProfileView, UserProfileView, TrainerDetailView);

– Edit/ — редагування персональних даних (EditUserProfileView, EditTrainerProfileView);

– Home/ — домашні екрани користувача і тренера, включно з рекомендаціями;

– List/, Row/ — відображення списків та елементів списків (рядків);

– Components/ — універсальні UI-елементи: кнопки, текстові поля, слайдери, картки, імідж-пікери;

– OnboardingPage.swift — реалізація екранів першого запуску (онбординг).

ViewModel-и (ViewModels/) відповідають за логіку, пов’язану з управлінням інтерфейсом:

– EditUserProfileViewModel, EditTrainerProfileViewModel — керування оновленням профілю;

– TrainerListViewModel — реалізація фільтрації тренерів;

– RecommendationViewModel — побудова векторів та обчислення косинусної схожості для рекомендацій.

Утиліти та сервіси:

– Utilities/ — включає FeatureNormalizer, FeatureWeights, RealmHelpers, які реалізують обробку векторів, нормалізацію та допоміжну логіку;

– Extensions/ — розширення моделей, наприклад, Trainer+Features.swift;

– Services/ — сервіси для взаємодії з базою (RealmService.swift) та логікою запитів (ContactService.swift).

Механізми міграції бази даних:

Migrations/RealmMigrationManager.swift, Migration_v9.swift — реалізують перевірку та оновлення структури Realm-бази у разі змін моделей.

Графічні ресурси (Assets.xcassets/):

– зображення для онбордингу;

– логотипи для різних розмірів;

– системна кольорова палітра.

Конфігураційні та службові файли:

– SpachApp.swift, ContentView.swift — вхідна точка застосунку;

– Spach.entitlements — визначає системні дозволи;

– *.xcodeproj, *.xcworkspace — службові файли Xcode-проєкту, схеми запуску, збереження сесій, параметри середовища розробки.

Тестові файли:

– SpachAppTests.swift;

– SpachAppUITests.swift;

– SpachTests.swift.

Юніт- та UI-тестування, готове до розширення відповідно до потреб.

Такий рівень деталізації у файловій структурі дозволяє підтримувати якісний рівень організації проєкту, швидко орієнтуватися в коді, розширювати його у майбутньому та забезпечує зручність у процесі командної або індивідуальної розробки.

5.3 Архітектура застосунку

Архітектура створеного застосунку базується на сучасному та ефективному патерні Model-View-ViewModel (MVVM), який забезпечує чітке розділення відповідальності між візуальним представленням, бізнес-логікою та структурою даних. Обрання саме цього архітектурного підходу зумовлене прагненням до підвищення масштабованості системи, покращення якості коду та зручності супроводу у майбутньому. Така структура дозволяє розробникам легко орієнтуватися в логіці застосунку, розмежовувати зони відповідальності та уникати дублювання коду.

Модельні класи (User, Trainer, ContactRequest, Rating) реалізовані як об'єкти Realm, що дозволяє ефективно зберігати та запитувати дані в межах локальної бази. Завдяки використанню @Persisted властивостей, модельні сутності автоматично оновлюються в базі, що мінімізує необхідність ручного керування збереженням.

Кожна модель має свій унікальний ідентифікатор, а також зв’язки з іншими сутностями, що відображає реальні взаємозв’язки між об’єктами системи. Наприклад, ContactRequest містить посилання як на клієнта, так і на тренера, що дозволяє швидко формувати логіку взаємодії.

ViewModel-складова застосунку відповідає за всю бізнес-логіку: створення нових записів, фільтрацію тренерів за заданими критеріями, обробку запитів на контакт, збереження редагованих даних, обчислення рейтингу тощо. Усі ViewModel реалізовані як класи, що дотримуються протоколу ObservableObject, що дає змогу інтерфейсу реагувати на зміни даних в режимі реального часу. Такий підхід особливо важливий у випадках, коли користувач змінює інформацію у своєму профілі або надсилає запит на контакт — зміни миттєво відображаються в інтерфейсі, не потребуючи перезапуску застосунку. Крім того, в рамках ViewModel застосовано окремі сервіси (наприклад, ContactService та RatingService), які додатково структурують бізнес-логіку за принципом інверсії залежностей.

Рівень представлення, реалізований за допомогою SwiftUI, забезпечує створення інтуїтивно зрозумілого, адаптивного та сучасного інтерфейсу. Завдяки декларативному стилю програмування, SwiftUI дозволяє будувати складні візуальні структури з мінімальним обсягом коду та високим рівнем підтримки анімацій і адаптивності. Компоненти, як-от TrainerRowView або UserDetailView повторно використовуються в різних частинах застосунку, що сприяє модульності та спрощенню обслуговування інтерфейсу. Також використання @State, @Binding та @Environment дозволяє ефективно передавати дані між екранами без надмірної складності.

Загалом архітектура системи забезпечує стабільну роботу в умовах локального середовища, відсутності інтернет-з’єднання та не потребує складної конфігурації зовнішнього сервера. Це значно спрощує розгортання застосунку та робить його зручним у використанні на різних пристроях Apple. Структурованість і модульність дозволяють масштабувати проєкт, у майбутньому додавати нові функції, наприклад, синхронізацію з хмарою або підтримку push-сповіщень.

5.4 Модулі системи

Застосунок структуровано за принципом модульності, що дозволяє ефективно організувати логіку програми, полегшити підтримку та масштабування проєкту. Кожен модуль реалізує окрему функціональну підсистему, логічно об’єднану навколо певного сценарію використання. Модулі взаємодіють через спільну локальну базу даних Realm, використовуючи моделі, сервіси та ViewModel-и, реалізовані у відповідних файлах.

Першим функціональним модулем є модуль реєстрації та авторизації. У ньому реалізовано два незалежні інтерфейси — для звичайного користувача та для тренера. Відповідні екрани (RegisterUserView.swift, RegisterTrainerView.swift, LoginView.swift) забезпечують зручне введення персональних даних, з подальшою перевіркою на наявність у базі. Центральний вузол логіки (AuthRootView.swift) керує переходами між станами автентифікації та реєстрації.

Другим важливим елементом є модулі профілів користувача і тренера. Ці модулі відповідають як за перегляд, так і за редагування персональних даних. Профілі виводяться на відповідних екранах (UserProfileView.swift, TrainerProfileView.swift), де відображається актуальна інформація про користувача або тренера. Для внесення змін передбачено окремі форми редагування (EditUserProfileView.swift, EditTrainerProfileView.swift), які тісно взаємодіють із ViewModel-ами (UserProfileViewModel.swift, TrainerProfileViewModel.swift). Саме через ці модулі здійснюється управління персональними характеристиками, які надалі впливають на фільтрацію та підбір.

Модуль пошуку та рекомендацій є основним механізмом персоналізованого добору тренерів. На основі характеристик, заданих користувачем (вік, очікуваний досвід, мови, райони тощо), формується вектор очікувань, який порівнюється із вектором кожного тренера через алгоритм косинусної схожості. Цей функціонал реалізується через комбінацію View (TrainerListView.swift, TrainerRowView.swift) та математичної логіки обробки даних у ViewModel (TrainerListViewModel.swift, RecommendationViewModel.swift). Для нормалізації даних і обчислення вагових

коефіцієнтів використовуються окремі утиліти (FeatureNormalizer.swift, FeatureWeights.swift, Trainer+Features.swift).

Наступним є модуль взаємодії між користувачем і тренером, що реалізується через систему запитів на контакт. Коли користувач знаходить потенційного тренера, він має можливість надіслати запит на зв’язок. Тренер у свою чергу бачить список вхідних запитів (UserListView.swift) та може їх підтвердити або відхилити (ClientRequestsView.swift). Для зручності реалізовано відповідні інтерфейсні компоненти (UserRowView.swift, ClientRowView.swift) та сервісну логіку (ContactService.swift). Інформація про запити зберігається у моделі ContactRequest.swift.

Модуль оцінювання тренерів надає користувачам змогу виставляти рейтинг після завершення співпраці. Реалізовано обчислення середнього балу та виведення кількості оцінок у профілі тренера (TrainerDetailView.swift). Виставлення оцінки здійснюється через окремий екран (RatingView.swift), а всі оцінки зберігаються у структурі моделей TrainerRating.swift та Review.swift.

Останнім функціональним компонентом є модуль онбордингу та головних екранів. При першому запуску застосунок показує користувачу серію ознайомчих слайдів (OnboardingPage.swift). Після цього користувач потрапляє на головний екран відповідно до своєї ролі — або UserHomeView.swift, або TrainerHomeView.swift. Ці екрани забезпечують доступ до всіх основних функцій, а їх організація здійснюється через контейнер навігації MainTabView.swift.

Кожен із модулів логічно ізольований, але пов’язаний із загальною архітектурою через моделі Realm, сервіси та ViewModel-и, що дозволяє системі залишатися цілісною, ефективною та гнучкою до розширення.

5.5 Інтерфейсна частина системи

Інтерфейс застосунку продумано таким чином, щоб він охоплював усі життєві сценарії користувача — від входу до завершення взаємодії з тренером. Кожна категорія користувачів — тренер або клієнт — має доступ до власного набору екранів, адаптованого до їхніх потреб. Такий розподіл дозволяє уникнути перевантаження інтерфейсу зайвими елементами та забезпечує інтуїтивно зрозумілу навігацію. Особливу увагу приділено зручності у користуванні — кожен екран реалізовано відповідно до сучасних принципів UI/UX-дизайну, включаючи використання великих інтерактивних елементів, контрастного тексту та адаптивної верстки.

Окрему увагу приділено доступності та інклюзивності інтерфейсу реєстрації. Всі елементи форми мають достатній контраст і чітке маркування, що дозволяє комфортно користуватись застосунком навіть на пристроях з невеликим екраном або за умов недостатнього освітлення. Крім того, форма реєстрації динамічно адаптується до обраної ролі: тренер бачить додаткові поля, пов’язані з досвідом, спеціалізацією та вартістю послуг, у той час як клієнт вказує очікування щодо тренера, що одразу формує основу для подальшого персоналізованого пошуку.

Інтерфейс починається з екранів входу та реєстрації. У залежності від ролі, користувач бачить відповідну форму: для тренера або клієнта. Ці екрани містять зручні поля введення, підказки, перевірку даних у режимі реального часу. У разі помилки користувач отримує негайне візуальне повідомлення. Передбачено також валідацію полів із контекстною підсвіткою та анімаціями, що покращують користувацький досвід. Реалізовано механізм автозаповнення поширених даних, наприклад email або імені, що прискорює процес входу. Реєстрація супроводжується мінімалістичним дизайном із достатнім простором, адаптованим як для мобільного, так і для десктопного середовища.

На рисунках 5.1 та 5.2 зображено екран реєстрації користувача, який включає введення особистих даних, вибір уподобань щодо тренера та фільтрацію за категоріями, досвідом і локацією.

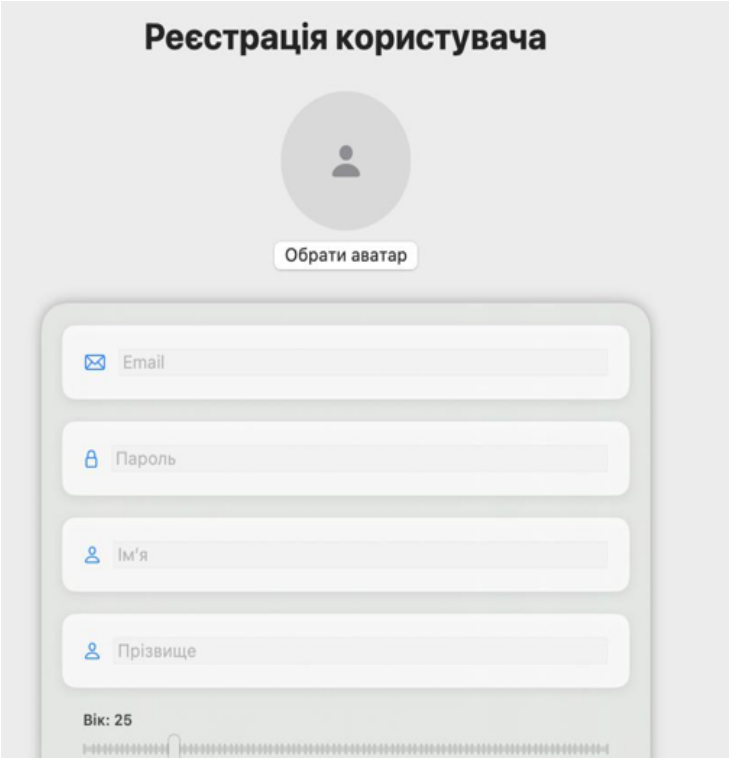

Рисунок 5.1 — Екран реєстрації користувача

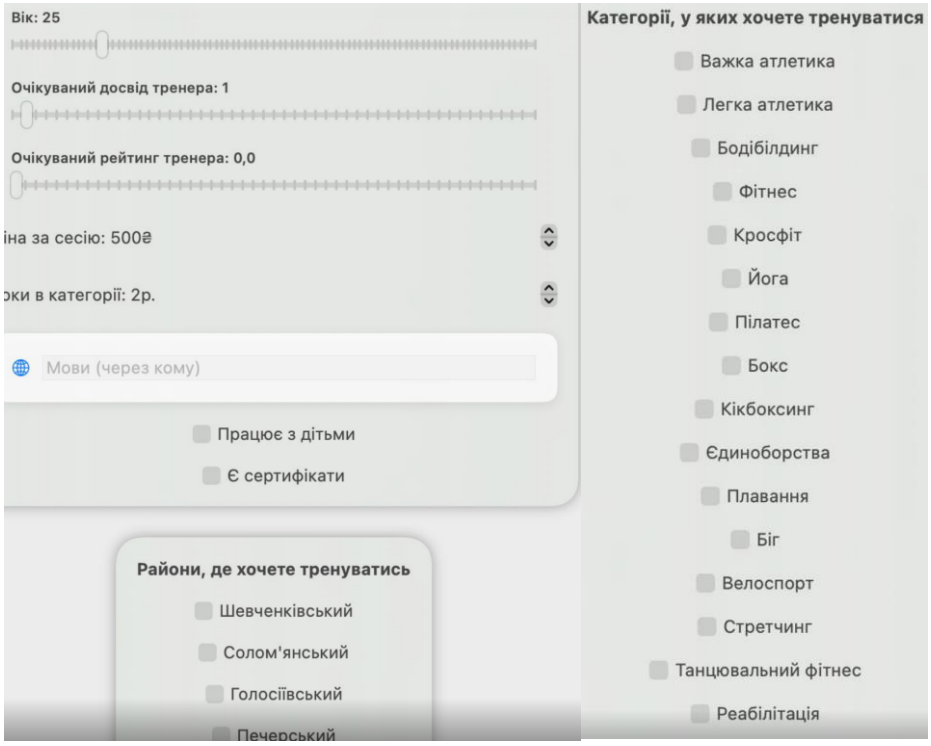

Рисунок 5.2 — Продовження екрану реєстрації користувача

На рисунках 5.3 та 5.4 представлено екран реєстрації тренера, який включає введення особистих даних, вказання досвіду, вартості послуг, категорій діяльності та районів, у яких проводяться тренування.

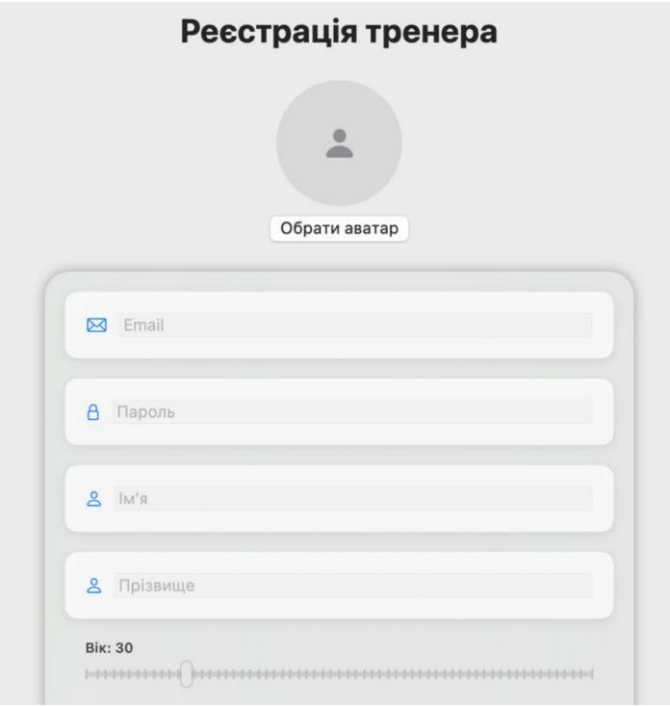

Рисунок 5.3 — Екран реєстрації тренера

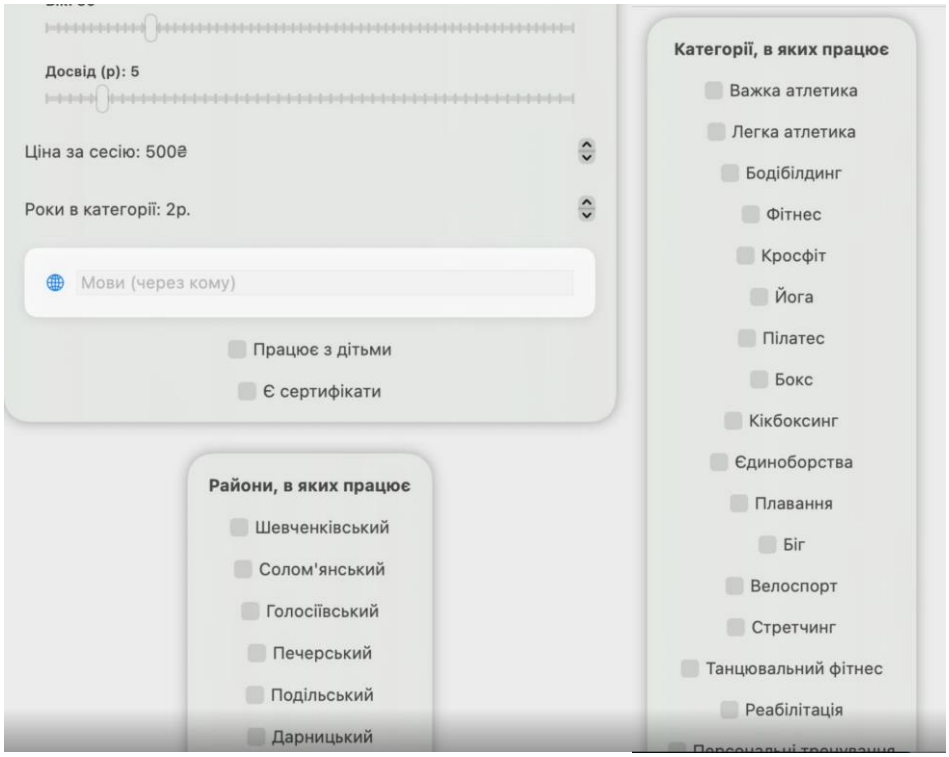

Рисунок 5.4 — Продовження екрану реєстрації тренера

На рисунку 5.5 представлено екран входу до системи, який забезпечує користувачам інтуїтивно зрозумілий інтерфейс авторизації з формами для введення електронної пошти та пароля, а також навігаційною панеллю для швидкого доступу до реєстраційних форм.

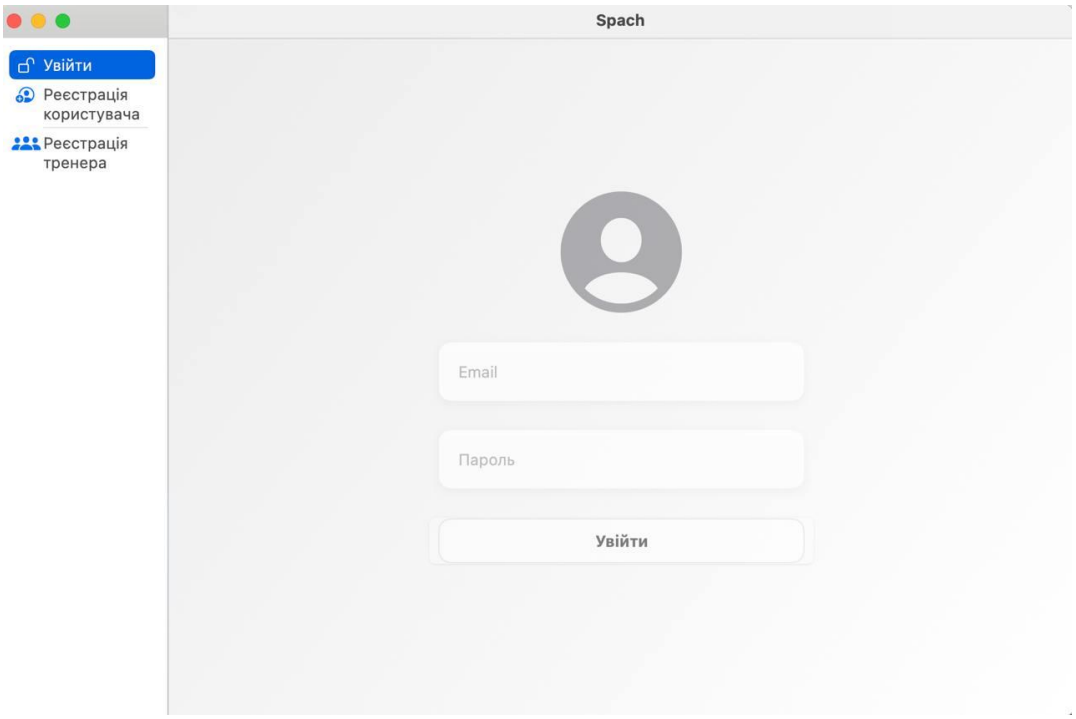

Рисунок 5.5 — Екран входу до системи

Користувачі можуть переглядати та оновлювати свої профілі. Для цього реалізовано зручні візуальні компоненти: аватар, іконки, повзунки, перемикачі, що дозволяє швидко редагувати основні параметри. Окремі блоки профілю чітко розділено на особисті дані та вподобання, що спрощує навігацію й підвищує зручність користування. Також передбачено кнопки для редагування та виходу з облікового запису, що дозволяє користувачу оперативно керувати своїм профілем.

На рисунку 5.6 зображено головний екран користувача з усіма персональними налаштуваннями, включаючи критерії для підбору тренера.

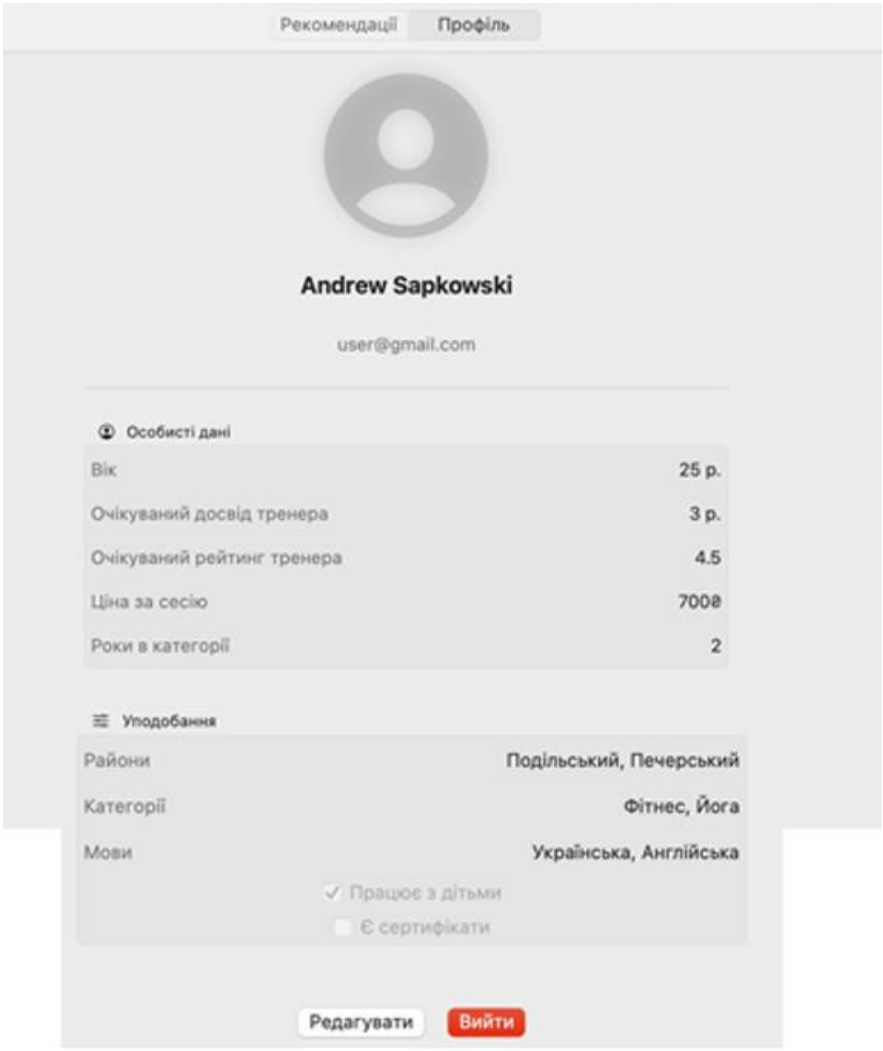

Рисунок 5.6 — Головний екран користувача

На рисунку 5.7 зображено екран редагування профілю тренера, де передбачено можливість оновлення особистих даних, вподобань, контактної інформації та аватару. Інтерфейс реалізовано у зручному формі з інтерактивними елементами для швидкого внесення змін. Застосовано компоненти з автоматичним збереженням стану, що дозволяє уникнути втрати даних при навігації між екранами. Всі поля адаптовані до введення відповідного типу даних, включаючи числа, електронну пошту та текст. Це забезпечує точність введення й покращує загальну якість користувацького досвіду.

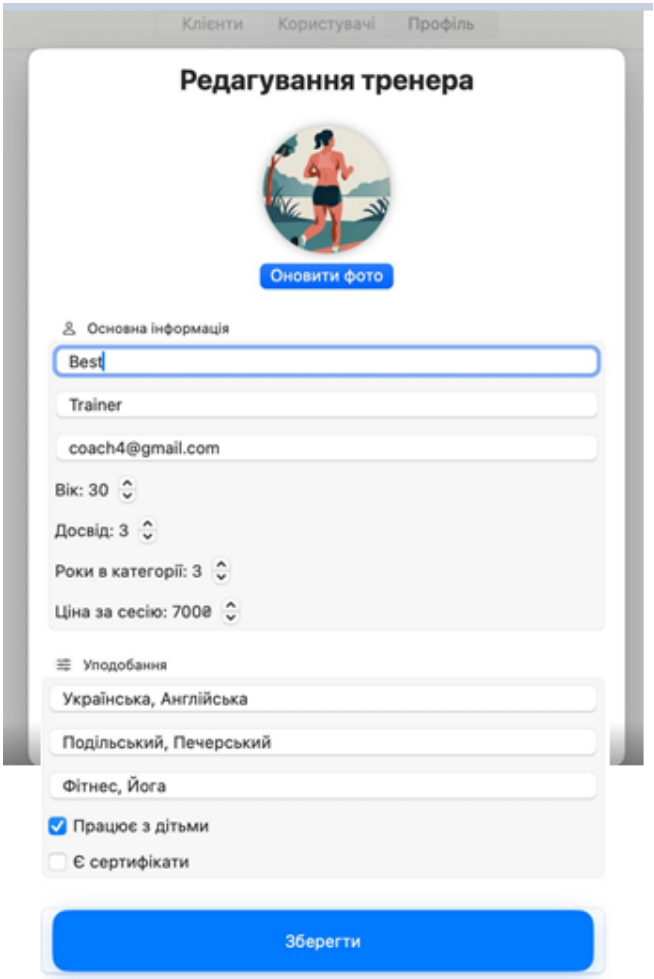

Рисунок 5.7 — Екран редагування профілю тренера або юзера

У профілі тренера доступний екран, де він бачить усіх користувачів, які надіслали запит на контакт. Кожен користувач має статус: «очікує», «підтверджено» або «відхилено». Інтерфейс побудований так, щоб тренер міг швидко приймати або відхиляти запити, натисканням однієї кнопки.

Інтерфейс відображення списку клієнтів тренера подано на рисунку 5.8.

Рисунок 5.8 — Екран із клієнтами тренера

Після підтвердження контакту користувач отримує доступ до повного профілю тренера — із контактними даними, рейтингом, досвідом, описом. Аналогічно, тренер може переглядати профіль користувача.

Інформацію про профіль тренера та клієнта, включно з основними характеристиками та додатковими параметрами, наведено на рисунку 5.9 — 5.10.

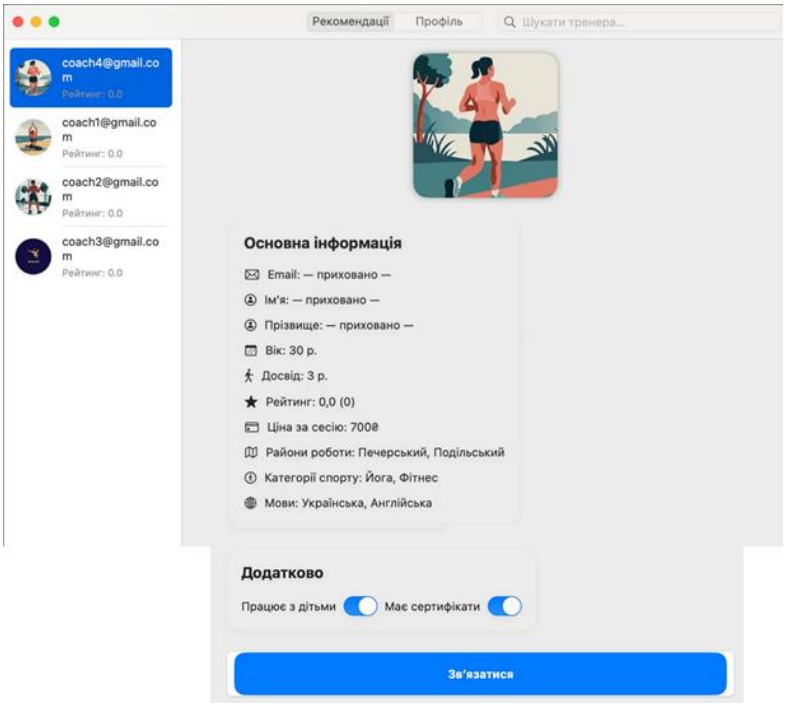

Рисунок 5.9 — Деталі профілю тренера

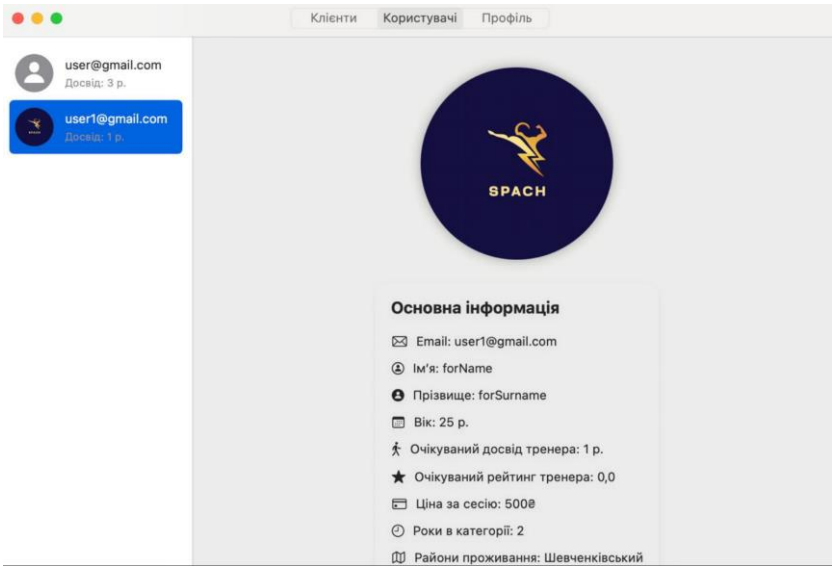

Рисунок 5.10 — Деталі профілю клієнта

5.6 Інформаційне забезпечення

Одним із критичних компонентів програмної системи є інформаційне забезпечення, яке охоплює не лише структуру збереження даних, але й логіку їх обробки, узгодженість між сутностями, контроль за актуальністю та збереженням цілісності. Саме від якості побудови інформаційної моделі залежать коректність роботи рекомендаційного механізму, швидкість реакції системи та її масштабованість.

Центральну роль в інформаційній архітектурі системи відіграє база даних Realm — високопродуктивний мобільний фреймворк, що забезпечує зберігання, оновлення та запит даних без потреби у зовнішньому сервері. На відміну від класичних реляційних баз даних, Realm побудований навколо об’єктної моделі, що ідеально підходить для використання в Swift-проєктах. Це дозволяє розробникам працювати з даними як з нативними об’єктами, забезпечуючи при цьому високу продуктивність, зниження складності коду та зручну інтеграцію з UI-компонентами.

Уся інформація у застосунку представлена у вигляді об'єктів класів-моделей, що наслідують Object із фреймворку Realm. Кожна така модель містить набір властивостей, які автоматично перетворюються у відповідні стовпці в базі даних, а також можуть зберігати зв’язки з іншими об’єктами через колекції (List<T>). Наприклад, один тренер може мати список пов’язаних контакт-запитів або рейтингів. Крім того, Realm підтримує реактивність — усі зміни, що вносяться до об’єктів, автоматично оновлюють пов’язані представлення у UI завдяки спостереженню за об'єктами (ObservedResults, @ObservedRealmObject тощо).

Це забезпечує не лише високу швидкодію та синхронізацію стану додатку, а й знижує кількість потенційних помилок при оновленні даних вручну. Наприклад, після підтвердження запиту на зв’язок між користувачем і тренером оновлення даних відбувається автоматично в усіх відповідних компонентах: статус запиту змінюється, особисті дані стають доступними, а рейтинг тренера стає відкритим для оцінювання. Такий підхід значно спрощує логіку додатку, робить його більш

стабільним і адаптивним до змін, а також полегшує підтримку та розвиток системи в майбутньому.

У системі реалізовано такі основні моделі:

– User (User.swift) — містить поля для імені, email, віку, очікуваного досвіду, списку мов, обраних районів, аватара, а також поля для expectedRating та зв’язку із запитами;

– Trainer (Trainer.swift) — описує профіль тренера з такими характеристиками: досвід, категорії спорту, сертифікати, робота з дітьми, мови, райони, рейтинг, аватар, контактна інформація;

– ContactRequest (ContactRequest.swift) — модель, що відображає запит на встановлення контакту між користувачем і тренером. Містить поля fromUserId, toTrainerId, status та timestamp;

– TrainerRating (TrainerRating.swift) — пов’язує користувача і тренера з полем value для збереження оцінки (від 1 до 5);

– Review (Review.swift) — текстові або числові відгуки з додатковими метаданими.

У Realm передбачено автоматичну підтримку зв’язків між сутностями.

Повну структуру основних сутностей системи, включаючи їх атрибути, зв’язки між об’єктами та ієрархію взаємодії між класами User, Trainer, ContactRequest та TrainerRating, відображено на кресленику ІС11.050БАК.005 Д4. Наприклад:

– кожен Trainer має список запитів, які можна фільтрувати за статусом;

– кожен User пов'язаний з оцінками, які він виставив (TrainerRating);

– при підтвердженні запиту між User і Trainer відбувається логічне оновлення обох об’єктів.

Це дозволяє ефективно виконувати запити, наприклад, отримати всіх користувачів, що вже встановили контакт із тренером, або перевірити, чи є чинний запит між двома сторонами.

Для збереження або оновлення об'єктів у базі використовується RealmService.swift, який реалізує CRUD-операції (створення, читання, оновлення,

видалення). Програмна реалізація методу додавання нового об'єкта зображена на рисунку 5.11.

```
func add<T: Object>(_ object: T) {
    try! realm.write { realm.add(object) }
}
```

Рисунок 5.11 — додавання нового об’єкту

Приклад оновлення даних користувача подано на рисунку 5.12.

```
func updateUser(_ user: User,
                block: (User) -> Void) {
    let realm = try! Realm()
    try! realm.write {
        block(user)
    }
}
```

Рисунок 5.12 — оновлення юзера

Усі моделі мають унікальні ідентифікатори (@Persisted(primaryKey: true) var _id = ObjectId.generate()), що забезпечує унікальність записів і дозволяє легко виконувати пошук за ключем.

Для гарантування цілісності інформації використовуються такі підходи:

– використання перевірки на унікальність email-адреси під час реєстрації;

– заборона дублювання оцінок тренера для пари user → trainer;

– автоматичне видалення запитів при видаленні користувача або тренера.

Міграція бази даних у разі змін структури моделей — за допомогою класів Migration_vn.swift, RealmMigrationManager.swift.

5.7 Конфігурації та налаштування

У процесі створення інформаційної системи особливу увагу було приділено налаштуванню середовища розробки, правильній організації збереження налаштувань користувача та забезпеченню захисту персональних даних. Конфігурація проєкту охоплює як середовище Xcode [8] і SwiftUI, так і внутрішню логіку управління доступами, станами автентифікації та приватності.

Проєкт реалізовано у середовищі Xcode 15, з використанням мови Swift 5.9 та фреймворку SwiftUI. Для збереження даних використано RealmSwift 10+, який інтегрується локально без налаштування зовнішнього серверу. Основною ціллю було створення повністю офлайн-здатного застосунку, придатного для роботи без підключення до мережі.

Усі конфігураційні файли (наприклад, Spach.entitlements) налаштовано таким чином, щоб застосунок мав доступ лише до локального збереження, що виключає витік або передавання персональної інформації назовні.

Для збереження простих налаштувань користувача, які не потребують структурованого зберігання у базі даних, використовується AppStorage [9] — високорівнева обгортка над UserDefaults. Наприклад:

– зберігання поточного email користувача (@AppStorage("currentEmail"));

– збереження статусу входу (@AppStorage("isLoggedIn"));

– відображення онбордингу лише при першому запуску (@AppStorage("hasSeenOnboarding")).

Такий підхід дозволяє ефективно управляти глобальними станами без ускладнення моделі даних Realm.

Оскільки застосунок оперує персональними даними (імена, email, фото), у ньому реалізовано механізм приховування чутливої інформації до моменту встановлення взаємного зв’язку. Зокрема:

– ім’я, прізвище, email тренера не відображаються до підтвердження запиту з обох сторін;

– аналогічно, тренер не бачить імен користувачів, які ще не пройшли підтвердження.

З метою підвищення безпеки взаємодії також реалізовано контрольовану логіку доступу до функціоналу залежно від ролі користувача та поточного контексту. Наприклад, лише після встановлення підтвердженого зв'язку активуються можливості оцінювання тренера, надсилання відгуку або перегляду додаткових деталей профілю. Завдяки такому підходу, застосунок не лише захищає персональні дані, але й забезпечує інтуїтивно зрозумілий сценарій використання: користувачі бачать лише те, що актуальне на поточному етапі взаємодії. Це створює відчуття довіри до платформи та підвищує її зручність, особливо в умовах конфіденційного характеру спортивного консультування.

Ця логіка реалізована у відповідних екранах через умовні перевірки статусу ContactRequest. Таким чином, навіть у разі несанкціонованого доступу до пристрою третя сторона не зможе отримати персональні дані інших користувачів.

Усі основні стани, які змінюються під час роботи застосунку, зберігаються у змінних середовища або спеціалізованих обгортках. Це дозволяє забезпечити гнучкість конфігурації без необхідності перезапуску програми. Наприклад, перемикання між роллю «тренер» або «користувач» впливає на структуру MainTabView, яка формується динамічно.

Висновок до розділу 5

У п'ятому розділі було детально розглянуто процес розробки застосунку для таких платформ Apple [10], як macOS та iOS, побудованого за архітектурою MVVM з використанням локальної бази даних Realm. Застосунок має модульну структуру, що охоплює основні функціональні блоки — автентифікацію, профілі, рекомендації, контакт-запити та інші, що забезпечує зручність розробки, тестування та масштабування. Особливу увагу приділено реалізації інтуїтивного інтерфейсу через SwiftUI та впровадженню механізмів захисту персональних даних.

6 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

6.1 Змістовна постановка задачі

У межах розробки системи підбору спортивних тренерів було виявлено потребу в ефективному алгоритмі, здатному здійснювати персоналізований добір тренерів для користувачів на основі множини характеристик. Користувачі мають індивідуальні очікування щодо типу спорту, географічного розташування тренера, володіння певними мовами, наявності сертифікатів, досвіду, вікової категорії тощо. Водночас кожен тренер характеризується власним вектором параметрів, що визначає його відповідність запиту.

Таким чином, математична модель базується на комбінуванні кількох важливих етапів: нормалізації значень, зважування за пріоритетами, інверсії окремих ознак та фінальному підрахунку косинусної міри схожості. Такий підхід дозволяє забезпечити об’єктивну й адаптивну логіку підбору спортивного тренера для кожного користувача.

Задача полягає в тому, щоб розробити математичний апарат, який дозволяє визначити ступінь відповідності кожного тренера індивідуальним очікуванням користувача та сформувати відсортований список рекомендацій. Урахування різних типів параметрів (категоріальні, числові, булеві) ускладнює просте порівняння, тому було вирішено застосувати метод косинусної схожості, адаптований під вагову модель [11] та з нормалізацією значень.

6.2 Математична постановка задачі

Рекомендаційна система підбору спортивних тренерів передбачає визначення ступеня відповідності між запитом користувача та характеристиками кожного тренера. Для цього було запропоновано математичну модель, що базується на поданні обох сторін (користувача й тренера) у вигляді багатовимірних векторів ознак.

Нехай маємо множину користувачів $U = \{u_1, u_2, \ldots, u_n\}$ та множину тренерів $T = \{t_1, t_2, \ldots, t_m\}$, де кожен користувач з бажаними критеріями про тренера u_i та тренер t_j описуються векторами ознак однакової довжини $V_{u_i} = [x_1, x_2, \ldots, x_k]$, $V_{t_i} = [y_1, y_2, \ldots, y_k]$.

Для урахування важливості окремих ознак вводиться вектор ваг $W = [w_1, w_2, \ldots, w_k]$.

Обчислення схожості між векторами реалізується за допомогою косинусної міри, що дозволяє оцінити схожість напрямків у багатовимірному просторі незалежно від довжини векторів:

$$\cos(\theta) = \frac{A * B}{||A|| * ||B||} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} * \sqrt{\sum_{i=1}^{n} B_i^2}} \quad (6.1)$$

де:

– A_i — і-та компонента вектора користувача після нормалізації та зважування;

– B_i — і-та компонента вектора тренера після нормалізації та зважування;

– n — кількість ознак (розмірність простору);

– $\cos(\theta)$ — значення косинусної схожості в діапазоні [0;1].

Перед обчисленням кожен вектор нормалізується за формулою:

$$x_i' = \frac{x_i - \min(x)}{\max(x) - \min(x)} \quad (6.2)$$

Після цього застосовується інверсія для тих ознак, де менше значення є кращим (наприклад, ціна за сесію):

$$x_i'' = 1 - x_i' \quad (6.3)$$

А потім проводиться зважування:

$$x_i''' = x_i'' * w_i \tag{6.4}$$

У результаті, і уявлення користувача про тренера, і реальні тренери представлені у вигляді нормалізованих та зважених векторів, а обчислення косинусної схожості між ними дозволяє отримати оцінку ступеня відповідності. На основі цього формується список тренерів, впорядкований за зменшенням значення косинуса.

6.3 Обґрунтування методу розв’язання

З огляду на поставлену задачу — побудову ефективної системи рекомендацій спортивних тренерів, було проаналізовано можливі методи розв’язання проблеми зіставлення векторів ознак. Враховуючи характер даних (суміш категоріальних, бінарних і числових ознак), а також потребу у зважуванні та нормалізації, було прийнято рішення про використання методу косинусної схожості.

На відміну від евклідової або мангеттенської метрик [12-13], косинусна міра дозволяє об’єктивно оцінювати схожість між векторами незалежно від їхньої абсолютної довжини. Це особливо актуально у контексті користувацьких вподобань, які можуть бути частково заповнені або містити дисбаланс у числових значеннях. Косинусна схожість дозволяє зосередитися на співпадінні напрямків векторів, а не їхніх абсолютних значень.

Крім того, ця метрика має низьку обчислювальну складність та добре масштабується. Її обчислення базується на скалярному добутку нормалізованих векторів, що дозволяє зменшити споживання ресурсів і забезпечити швидкий відгук користувачеві навіть на пристроях із обмеженою продуктивністю.

Особливу увагу було приділено питанню вагування ознак, оскільки деякі параметри (наприклад, спортивна категорія або район надання послуг) мають набагато більший вплив на вибір тренера, ніж інші (наприклад, кількість мов або

наявність сертифікатів). Саме тому застосовано підхід з побудовою вектора ваг, який визначає значущість кожного параметра. Вага розподіляється пропорційно до кількості значень у категоріях та заздалегідь визначених пріоритетів, що дозволяє більш точно врахувати індивідуальні переваги користувача.

Також, враховано необхідність нормалізації числових ознак, аби уникнути ситуації, коли значення на кшталт ціни або досвіду тренера непропорційно впливають на результат. Для цього було реалізовано механізм масштабування та інверсії значень (наприклад, нижча ціна вважається кращою), що дозволяє гармонізувати вектори перед їх порівнянням.

Таким чином, з-поміж альтернатив було обрано алгоритм на основі зваженої косинусної схожості з попередньою нормалізацією ознак, що дозволяє забезпечити високу точність рекомендацій за низьких обчислювальних витрат. Такий підхід не лише підвищує релевантність результатів, але й забезпечує масштабованість та адаптивність системи до змін у структурі даних або пріоритетах користувача, що робить його оптимальним для інтеграції в сучасні рекомендаційні мобільні застосунки.

6.4 Косинусна міра схожості

У межах побудови рекомендованої системи було обрано метод оцінювання схожості між профілями користувача та тренера, який ґрунтується на косинусній мірі. Косинусна міра подібності дозволяє визначити ступінь схожості між двома векторами, які представляють множину ознак. Відмінність цього підходу полягає в тому, що він не враховує абсолютні значення ознак, а зосереджується на напрямку векторів у багатовимірному просторі. Тобто, важливо не те, наскільки великі значення у векторі, а те, наскільки однаковим є їх розподіл.

Це особливо корисно в тих випадках, коли дані мають різні шкали або частково заповнені — наприклад, якщо користувач вибрав лише деякі категорії чи мови. Косинусна міра дозволяє порівнювати такі часткові профілі з повними профілями тренерів і коректно обчислювати ступінь відповідності.

Перевагою косинусної схожості є її незалежність від довжини вектора. Наприклад, два тренери можуть мати однакові види спорту, але відрізнятися за досвідом чи кількістю клієнтів. Косинусна міра в такому разі все одно покаже високу схожість, якщо співвідношення між ознаками зберігається. Це не властиво більшості інших метрик відстані, таких як евклідова або мангеттенська, які значною мірою залежать від абсолютних значень.

Ще однією суттєвою перевагою є стійкість до впливу "шумових" ознак. Якщо деякі характеристики заповнено некоректно або є крайні значення, косинусна міра не спотворюється настільки суттєво, як інші підходи. Це робить її особливо зручною у мобільних застосунках, де користувачі можуть вводити неповні або неточні дані.

У порівнянні з евклідовою відстанню, яка базується на різниці між відповідними компонентами векторів, косинусна міра не зважає на величини, а лише на їхні пропорції. Відповідно, навіть якщо один тренер має значно більший досвід, але решта ознак співпадає з профілем користувача, система визнає його релевантним. Це забезпечує гнучкість у роботі з даними, які мають різну природу: числову, категоріальну, бінарну тощо.

Також важливо відзначити, що косинусна схожість добре працює з розрідженими векторами. У нашому випадку, велика кількість ознак формується через перетворення категоріальних даних у one-hot [14] представлення, внаслідок чого більшість елементів вектора можуть мати нульове значення. Косинусна міра без проблем обробляє такі випадки, фокусуючись лише на значущих співпадіннях.

Окрім аналітичних переваг, метод має ще й практичні. Його обчислення не потребує складної математики або значних ресурсів, що дозволяє реалізувати алгоритм навіть на мобільному пристрої без підключення до серверної частини. Таким чином, це оптимальний вибір з огляду на ефективність, точність та продуктивність.

Зважаючи на зазначене, можна зробити висновок, що косинусна міра подібності є найбільш збалансованим підходом для реалізації персоналізованого підбору тренерів, оскільки вона забезпечує точність рекомендацій за мінімальних

витрат ресурсів, є стійкою до спотворених або неповних даних і чудово масштабується.

6.5 Опис методу розв'язання задачі

Для реалізації системи підбору тренерів було застосовано метод зваженої косинусної схожості. Алгоритм передбачає подання профілів користувача та тренера у вигляді векторів однакової довжини, які складаються з категоріальних, числових та бінарних ознак. Категоріальні ознаки, такі як вид спорту, район або мова, оцифровуються за допомогою one-hot кодування, тобто кожне можливе значення такої ознаки представляється окремою бінарною ознакою у векторі, де лише одне значення дорівнює 1, а решта — 0. Далі вектори нормалізуються, деякі ознаки інвертуються (наприклад, вартість), після чого кожне значення множиться на наперед задану вагу. Результатом є зважені вектори, які порівнюються між собою за допомогою косинусної міри подібності. Отримані коефіцієнти використовуються для ранжування тренерів відповідно до релевантності запиту користувача.

Такий підхід дозволяє досягти високої гнучкості: вага кожної ознаки може бути змінена відповідно до контексту або вподобань користувача. Наприклад, якщо для конкретного клієнта пріоритетом є район проведення тренувань або наявність сертифікатів, ці критерії можуть отримати вищу вагу при обчисленні схожості. Водночас менш значущі фактори, такі як мова спілкування або ціна, можуть мати нижчий вплив на кінцевий результат. Таким чином, система враховує як об’єктивні характеристики тренерів, так і суб’єктивні вподобання користувача.

Особливістю реалізації є те, що категоріальні ознаки, такі як типи спорту, райони чи мови, перетворюються на one-hot вектори, що дозволяє працювати з ними у рамках єдиної числової моделі. Бінарні параметри (наявність сертифікатів, робота з дітьми) представлені у вигляді 0 або 1, а числові значення (наприклад, досвід, ціна, рейтинг) — нормалізовані до одного масштабу. Завдяки цьому вектори можна ефективно порівнювати незалежно від природи початкових даних.

Застосування косинусної схожості як основного критерію порівняння обґрунтоване її інваріантністю до абсолютних значень ознак: вона оцінює саме напрямок вектора, тобто співвідношення характеристик, а не їхню абсолютну величину. Це дозволяє точно визначати, наскільки профіль тренера структурно «схожий» на очікування користувача навіть у разі, коли значення деяких параметрів суттєво відрізняються. Такий механізм забезпечує не лише точність, а й стабільність результатів, що є критично важливим для рекомендаційних систем.

Додатковою перевагою такого підходу є можливість динамічного оновлення критеріїв без необхідності повної перебудови моделі. У випадку розширення системи новими параметрами (наприклад, індивідуальний стиль тренувань, спеціалізація у певному виді спорту або тип локації), вони можуть бути інтегровані у векторну модель без порушення існуючої логіки обчислення. Це забезпечує масштабованість архітектури і дозволяє підтримувати актуальність рекомендацій у відповідь на зміни ринку чи потреб користувачів. Завдяки модульності реалізації, оновлення ваг, індексів інверсії або додавання нових ознак може відбуватися централізовано, без зміни базової логіки ранжування.

Система також може бути адаптована до персоналізованого навчання — наприклад, за допомогою зворотного зв’язку після взаємодії з тренером. У разі, якщо користувач залишає оцінку, ці дані можуть бути використані для тонкого налаштування ваг ознак, які виявилися більш або менш релевантними. Таким чином, реалізується механізм адаптивного самонавчання, який підвищує якість рекомендацій з кожною новою ітерацією використання. У перспективі це дозволяє реалізувати складніші сценарії, включаючи гібридні рекомендації або кластеризацію користувачів за стилем вибору тренерів, що додатково підсилює ефективність системи.

6.5.1 Структура та псевдокод алгоритму

Запропонований алгоритм реалізує метод добору тренерів на основі косинусної схожості між вектором ознак користувача та кожного окремого

тренера. Структура алгоритму передбачає декілька чітко визначених послідовних етапів обробки даних, що дозволяє забезпечити високу адаптивність системи до різних типів вхідної інформації.

На першому етапі формується вектор ознак користувача, який є основою для порівняння. Він включає інформацію про бажані категорії спорту, райони обслуговування, мови спілкування, а також числові параметри (наприклад, очікуваний рівень ціни або бажаний досвід тренера). Аналогічний вектор формується для кожного тренера на основі його профілю.

Далі виконується нормалізація числових даних, що дозволяє привести значення до єдиного масштабу. Це критично важливо для уникнення домінування окремих характеристик (наприклад, ціни) у процесі порівняння. Для певних ознак, таких як вартість послуг, проводиться інверсія значення — тобто, чим нижче значення, тим вища оцінка відповідності. Це дозволяє зберегти єдиний напрям інтерпретації: більше означає краще.

Далі застосовуються вагові коефіцієнти, які відображають пріоритетність різних ознак. Наприклад, збіг у категоріях спорту має більшу вагу, ніж спільна мова, що враховує очікування користувача та вплив кожної характеристики на якість тренування.

Після попередньої обробки проводиться обчислення косинусної подібності між користувацьким вектором та кожним тренерським вектором. Результати зберігаються у вигляді пар «тренер – рівень відповідності» та впорядковуються у зворотному порядку, що дозволяє сформувати ранжований список релевантних кандидатів.

Псевдокод алгоритму наведено в таблиці 6.1.

Таблиця 6.1 – Псевдокод алгоритму

Пункт	Опис кроку алгоритму
1	Для кожного тренера t у списку
2	Сформувати вектор уявного тренера для користувача u
3	Сформувати вектор реального тренера t
4	Нормалізувати числові значення обох векторів

Пункт	Опис кроку алгоритму
5	Інвертувати потрібні ознаки (наприклад, ціну, якщо менше — краще)
6	Застосувати вагові коефіцієнти до відповідних елементів векторів
7	Обчислити косинусну схожість між векторами u та t
8	Зберегти пару (t, similarity) до списку результатів
9	Відсортувати список результатів за спаданням значення similarity
10	Повернути перші N тренерів зі списку як найрелевантніших

Такий підхід дозволяє не лише гнучко адаптуватися до будь-якого профілю користувача, а й гарантує справедливість і логічність відбору завдяки уніфікованій структурі обробки ознак.

6.5.2 Опис програмної реалізації

Програмна реалізація алгоритму здійснена мовою Swift з урахуванням сучасних принципів модульного проєктування. Основна логіка алгоритму розбита на окремі компоненти, кожна з яких відповідає за свій етап обробки даних.

Ключовими елементами є дві структури: FeatureWeights та FeatureNormalizer.

FeatureWeights — надає доступ до вектора вагових коефіцієнтів, де загальна вага для кожної групи ознак рівномірно розподіляється між усіма категоріальними значеннями.

Структура також зберігає перелік індексів інверсних ознак, зокрема ціни за сесію, де менше значення є кращим. Розрахунок ваг поділено на групи ознак: категоріальні, бінарні та числові, що дозволяє гнучко масштабувати алгоритм при розширенні функціональності.

FeatureNormalizer — відповідає за нормалізацію, інверсію та зважування векторів. У методі normalize для кожного індексу ознаки враховується мінімальне та максимальне значення у вибірці тренерів, що дозволяє зберігати стабільність алгоритму навіть при значному розширенні бази даних. Далі застосовується умовна інверсія та множення на ваговий коефіцієнт.

Фінальний метод cosineSimilarity(a, b) реалізує обчислення скалярного добутку векторів та поділ на добуток їхніх норм, що відповідає класичній реалізації косинусної схожості.

Завдяки такій структурованій реалізації забезпечується високий рівень масштабованості та повторного використання коду. Наприклад, при додаванні нових ознак або розширення логіки зводиться до оновлення відповідного вектора ваг або нормалізації без потреби переписувати ядро алгоритму. Крім того, такий підхід дозволяє легко тестувати окремі компоненти, що є важливою умовою при розробці рекомендованих систем, орієнтованих на точність і надійність.

6.5.3 Опис сценаріїв роботи алгоритму

Розглянемо типові сценарії роботи алгоритму в межах застосунку, що ілюструють його поведінку у найпоширеніших випадках взаємодії користувача із системою.

Сценарій 1: Перегляд списку тренерів.

– користувач переходить до розділу з доступними тренерами;

– система формує вектор ознак уявлення користувача про тренера на основі його профілю;

– для кожного тренера формується вектор ознак з бази даних;

– застосовується нормалізація, інверсія та зважування;

– обчислюється косинусна схожість між уявленням користувача про тренера і кожним тренером;

– тренери впорядковуються за рівнем відповідності та відображаються у списку.

Сценарій 2: Зміна налаштувань профілю клієнта.

– користувач оновлює свої вподобання (наприклад, обирає нову категорію спорту або знижує бажану ціну);

– вектор ознак уявлення користувача про тренера автоматично оновлюється;

– алгоритм повторно виконує обчислення схожості з усіма тренерами;

– інтерфейс оновлюється з урахуванням нових релевантних кандидатів.

Сценарій 3: Додавання нового тренера до системи.

– тренер створює новий профіль або редагує існуючий;

– при наступному зверненні клієнта до списку тренерів, нові дані вже враховуються;

– система перераховує схожість для оновлених профілів автоматично;

– користувач бачить нові рекомендації без потреби у ручному оновленні.

Сценарій 4: Відображення рівня відповідності.

– для кожного тренера у списку додатково відображається коефіцієнт відповідності;

– клієнт може візуально оцінити, наскільки той чи інший тренер відповідає його очікуванням;

– це підвищує прозорість роботи системи та покращує взаємодію з користувачем.

Такий підхід до опису сценаріїв дозволяє продемонструвати гнучкість, динамічність та масштабованість алгоритму. Незалежно від контексту використання, система гарантує актуальність рекомендацій та ефективне реагування на зміни у даних.

Висновок до розділу 6

У розділі розглянуто математичне обґрунтування та реалізацію алгоритму підбору тренерів на основі косинусної схожості. Векторне представлення ознак, нормалізація, інверсія значень та вагові коефіцієнти забезпечили точне й масштабоване порівняння. Обчислення виконуються локально, що гарантує швидкодію та конфіденційність. Алгоритм відповідає вимогам до точності, надійності та ефективності.

7 ТЕСТУВАННЯ СИСТЕМИ

7.1 Мета випробувань

Тестування є невід'ємною частиною життєвого циклу створення програмного забезпечення, оскільки дозволяє переконатися у відповідності реалізованого функціоналу очікуванням користувачів, технічному завданню та загальним стандартам якості. У процесі розробки інформаційної системи для підбору спортивних тренерів метою тестування було підтвердження коректної роботи всіх основних функціональних модулів системи, перевірка її стабільності, безпеки та зручності користування.

Окремим завданням випробувань було переконатися у працездатності застосунку на різних платформах — macOS та iOS, включно з різними форм-факторами пристроїв. Також важливою метою було забезпечення правильності реалізації персоналізованого підбору тренерів за допомогою методу косинусної схожості, перевірка процесу автентифікації користувачів, логіки підтвердження контактів, а також надійної обробки профілів і рейтингових оцінок.

Важливим критерієм тестування була саме якість взаємодії з користувачем — наскільки швидко та без помилок працює система, чи виникають труднощі при виконанні базових сценаріїв і чи відчувається відповідність сучасним вимогам до дизайну та інтерфейсу мобільних застосунків.

7.2 Загальні положення

Тестування проводилось за допомогою комбінації методів ручного приймального тестування (UAT — User Acceptance Testing) [15] та функціонального тестування основних сценаріїв. Ручне тестування було обране як основна стратегія через необхідність безпосередньої перевірки UX-елементів, а також враховуючи гнучкість системи, що дозволяє швидко змінювати логіку без складної автоматизації.

Тестування здійснювалося у віртуальному середовищі симуляторів Xcode. Це дозволило переконатися в адаптивності інтерфейсу, правильному масштабуванні елементів та відсутності артефактів при взаємодії з UI.

Функціональною основою тестування стали ключові сценарії використання, такі як створення облікового запису, вхід у систему, редагування профілю, пошук тренера, відправлення запиту на контакт, підтвердження запиту з боку тренера, а також виставлення рейтингу після взаємодії. Ці дії охоплюють критично важливі компоненти бізнес-логіки та дозволяють перевірити не лише правильність реалізації, але і послідовність виконання, відповідність очікуванням, логічну завершеність кожного етапу.

Особливу увагу було приділено перевірці роботи рекомендаційної системи. У тестах перевірялося, чи коректно працює алгоритм, заснований на косинусній схожості, чи не виникають аномальні ситуації при зміні ваг ознак або крайніх значеннях вхідних параметрів. Проводилось тестування за кількома наборами даних, включно з ситуаціями, коли для користувача немає ідеально релевантного тренера.

Також важливим аспектом було перевірити, як система поводиться в умовах втрати мережевого з'єднання, чи коректно відображаються повідомлення про помилки, чи не відбувається краш застосунку та чи не зникають дані з БД.

Кожна перевірка супроводжувалася фіксацією вхідних даних, очікуваного та фактичного результату, а також висновком щодо проходження тесту. Основна частина тест-кейсів була виконана з позитивним результатом, що засвідчено у наступному підпункті.

7.3 Результати випробувань

У таблиці 7.1 наведено перелік тестових перевірок приймального тестування, які підтверджують коректну роботу основних функцій системи.

Таблиця 7.1 — Перелік тестових перевірок приймального тестування

№	Назва перевірки	Вхідні дані	Очікуваний результат	Статус перевірки
1	Реєстрація нового тренера та клієнта	Email, пароль	Успішна реєстрація, перехід до профілю	Пройдена
2	Валідація email при реєстрації	Email без “@” або з помилкою	Кнопка “Зареєструватися” стає недоступна	Пройдена
3	Вхід зареєстрованого користувача	Email, пароль	Перехід на головну сторінку	Пройдена
4	Відображення порожнього списку тренерів	Користувач вводить фільтр, який не дає збігів	Виводиться повідомлення “Нічого не знайдено” або порожній список	Пройдена
5	Наявність кнопки “Редагувати профіль”	Користувач авторизований	На екрані профілю присутня кнопка редагування	Пройдена
6	Оновлення особистих даних користувача	Ім’я, прізвище, мова, опис	Дані оновлюються та зберігаються, показується банер "Збережено успішно"	Пройдена
7	Збереження фільтрів після перезапуску застосунку	Обрані користувачем критерії	Після перезапуску відображаються попередні фільтри	Пройдена

№	Назва перевірки	Вхідні дані	Очікуваний результат	Статус перевірки
8	Відображення списку клієнтів тренеру	Тренер відкриває вкладку запитів	Виводиться список користувачів, які надіслали запит	Пройдена
9	Зміна аватару у профілі користувача	Фото у форматі PNG/JPEG	Новий аватар відображається у профілі	Пройдена
10	Відправка контакт-запиту тренеру	Користувач натискає “Зв’язатись”	Запит відображається у списку запитів тренера	Пройдена
11	Відсутність дублювання запитів	Користувач намагається двічі натиснути “Зв’язатись”	Повторний запит не надсилається, попередній зберігається	Пройдена
12	Підтвердження запиту тренером	Тренер натискає “Підтвердити”	Відкриваються особисті дані обох сторін	Пройдена
13	Відхилення запиту тренером	Тренер натискає “Відхилити”	Запит зникає зі списку, статус стає "Відхилено"	Пройдена
14	Перегляд профілю тренера після підтвердження	Контакт підтверджено обома сторонами	Відображаються всі дані тренера, включаючи ім’я, email, аватар	Пройдена

№	Назва перевірки	Вхідні дані	Очікуваний результат	Статус перевірки
15	Повторна спроба надіслати запит тренеру	Користувач уже надіслав запит	Кнопка “Зв’язатись” недоступна або з’являється повідомлення про вже надісланий запит	Пройдена
16	Відображення середнього рейтингу тренера	Тренер має кілька оцінок	Виводиться правильний середній бал і кількість оцінок	Пройдена
17	Оцінювання тренера користувачем	Кількість зірок (1–5)	Рейтинг оновлюється, тренеру додається відгук	Пройдена
18	Повернення на попередній екран	Користувач натискає кнопку “Назад”	Здійснюється перехід на попередній екран без втрати даних	Пройдена
29	Вихід із системи	Натискання кнопки “Вийти”	Повернення на екран входу	Пройдена
20	Некоректний вхід	Невірний email або пароль	Повідомлення про помилку	Пройдена
21	Коректне оцифрування категоріальних критеріїв	Користувач обирає спорт “Плавання”, мову “Англійська”,	Система зберігає відповідні числові вектори і відображає	Пройдена

№	Назва перевірки	Вхідні дані	Очікуваний результат	Статус перевірки
		район “Подільський“	релевантні результати	
22	Вплив категоріальних даних на фільтрацію	Уподобання користувача та тренера частково збігаються	Тренер потрапляє до списку рекомендованих із відповідним коефіцієнтом схожості	Пройдена
23	Валідація неправильного значення категорії	Категорія не з переліку (вручну підставлено у базу)	Значення ігнорується або викидається помилка при побудові векторів	Пройдена

Висновок до розділу 7

Результати тестування свідчать про те, що система здатна виконувати всі ключові функції без збоїв і відхилень від очікуваного результату. Жодна з перевірок не завершилася невдачею, що дозволяє стверджувати про відповідність системи технічним вимогам, її стабільність та готовність до використання. Окремо слід відзначити ефективність реалізації рекомендаційного алгоритму: у тестових сценаріях він демонстрував стабільно високі показники відповідності, обираючи тренерів, які дійсно відповідали критеріям користувача.

Також тестування показало, що інтерфейс застосунку відповідає сучасним стандартам — усі основні дії можна виконати за кілька кліків, навігація є логічною, а дизайн не перевантажений другорядними деталями.

ВИСНОВКИ

У межах дипломного проєкту на тему «Застосунок для рекомендації спортивних тренерів» було реалізовано повноцінну інформаційну систему, яка надає користувачам інструменти для зручного та персоналізованого пошуку тренерів, а також для взаємодії з ними в межах безпечної платформи. Результатом є кросплатформенний застосунок, що функціонує на базі macOS та iOS і відповідає сучасним вимогам до мобільного та десктопного програмного забезпечення.

На початковому етапі було здійснено глибокий аналіз предметної області, визначено основні потреби користувачів і сформовано вимоги до функціональності системи. Особливу увагу приділено моделюванню основних сутностей, логіці збереження зв’язків між користувачами і тренерами, а також механізмам приховування персональних даних до моменту взаємного підтвердження контакту.

Ключовим елементом системи стала рекомендаційна підсистема, що реалізує алгоритм на основі косинусної схожості. Кожен користувач задає власні вподобання щодо типу тренувань, мови, району та інших параметрів, які зіставляються з відповідними параметрами тренера. Алгоритм обраховує схожість векторів ознак і ранжує тренерів за релевантністю. У процесі реалізації було враховано нюанси нормалізації позитивних та інверсних ознак, а також визначено ваги для кожного критерію, що дозволило досягти високої точності відповідності.

Розробка здійснювалась із використанням мови Swift та фреймворку SwiftUI, що дало змогу забезпечити сучасний інтерфейс та зручну логіку відображення даних. Для локального зберігання інформації використано базу даних Realm, яка дозволяє організовувати об’єктно-орієнтовану структуру даних без використання зовнішнього серверу чи хмарної інфраструктури. Це рішення зробило систему повністю автономною та незалежною від підключення до інтернету у більшості сценаріїв.

Також було реалізовано механізм запитів між користувачами і тренерами: користувач надсилає запит на контакт, тренер його підтверджує або відхиляє, після

чого при підтвердженні відкриваються персональні дані. Така логіка забезпечує високий рівень приватності та контроль над розкриттям особистої інформації.

Застосунок містить окремі профілі для тренерів та користувачів, можливість редагування аватара, відображення кількості оцінок і рейтингу. Додатково реалізовано візуальні анімації для підтвердження дій користувача, такі як банер «Збережено успішно» та пульсація кнопок, що підвищує UX-рівень інтерфейсу.

У ході тестування було перевірено всі ключові сценарії: реєстрація, вхід, пошук тренерів, надсилання та обробка запитів, оновлення профілю, а також виставлення оцінок. Тестування проводилось вручну на фізичних пристроях і показало повну відповідність функціоналу заявленим вимогам. Жодних критичних помилок виявлено не було, а всі модулі працювали стабільно та ефективно.

Таким чином, реалізована система відповідає всім сформульованим цілям дипломного проєкту. Вона є функціонально завершеною, зручною у використанні, має сучасний дизайн і може бути розгорнута в реальних умовах або розширена до комерційного продукту. Надалі можливе доповнення системи онлайн-синхронізацією, аналітикою активності користувачів або інтеграцією з календарями для бронювання сесій, що зробить застосунок ще потужнішим і привабливішим для цільової аудиторії.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Swift programming language – Chris Ching courses. URL: https://codewithchris.com/ (дата звернення 03.03.2025).
2. SwiftUI – Apple Developer Documentation. URL: https://developer.apple.com/documentation/swiftui (дата звернення 10.03.2025).
3. Realm – MongoDB Realm Sync documentation. URL: https://www.mongodb.com/docs/realm/sync/ (дата звернення 17.03.2025).
4. MongoDB Realm SDK for Swift – official GitHub repository. URL: https://github.com/realm/realm-swift (дата звернення 31.03.2025).
5. Combine – Apple Developer Documentation. URL: https://developer.apple.com/documentation/combine (дата звернення 07.04.2025).
6. Introducing MVVM into your SwiftUI project. URL: https://www.hackingwithswift.com/books/ios-swiftui/introducing-mvvm-into-your-swiftui-project (дата звернення 10.04.2025).
7. Jure Leskovec, Anand Rajaraman, Jeffrey D. Ullman. Mining of Massive Datasets book. Palo Alto, CA: 2014. 496 с. (дата звернення 14.04.2025).
8. Xcode – Build, test, and submit your app with Apple’s integrated development environment. URL: https://developer.apple.com/documentation/xcode/ (дата звернення 21.04.2025).
9. SwiftUI, AppStorage – A property wrapper type that reflects a value from UserDefaults and invalidates a view on a change in value in that user default. URL: https://developer.apple.com/documentation/swiftui/appstorage (дата звернення 28.04.2025).
10. Building Cross-Platform SwiftUI Apps. URL: https://fatbobman.medium.com/building-adaptable-swiftui-applications-for-multiple-platforms-964624fa7b2 (дата звернення 05.05.2025).
11. How to weight values in a range with swift? URL : https://stackoverflow.com/questions/55003250/how-to-weight-values-in-a-range-with-swift (дата звернення 12.05.2025).

12. Що таке евклідова відстань і чому вона важлива в машинному навчанні? URL: https://uk.eitca.org/artificial-intelligence/eitc-ai-mlp-machine-learning-with-python/programming-machine-learning/euclidean-distance/examination-review-euclidean-distance/what-is-euclidean-distance-and-why-is-it-important-in-machine-learning/ (дата звернення 19.05.2025).

13. АНАЛІЗ МЕТРИК ДЛЯ ІНТЕЛЕКТУАЛЬНИХ ІНФОРМАЦІЙНИХ СИСТЕМ. URL: https://science.lpnu.ua/sites/default/files/journal-paper/2021/jun/23829/0210544infmaket-01-07-98-113_0.pdf (дата звернення 26.05.2025).

14. One Hot Encoding Explained. URL: https://medium.com/@heyamit10/one-hot-encoding-explained-0b0130ccd78e (дата звернення 02.06.2025).

15. User Acceptance Testing (UAT) – приймальне тестування та його цілі. URL: https://highload.tech/uk/user-acceptance-testing-uat-prijmalne-testuvannya-ta-jogo-tsili/ (дата звернення 09.06.2025).