Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБКА МУЛЬТИСЕРВІСНОЇ ПЛАТФОРМИ ДЛЯ АНАЛІТИКИ УСПІШНОСТІ СПОРТСМЕНІВ
(тема)

Виконав:
здобувач 4 року навчання,
групи ІТІНФ-21-2
Побізінський О. М.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник асист. Кобилін І. О.
(посада, прізвище, ініціали)

Допускається до захисту

Завідувач кафедри інформатики _____ Кобилін О. А.
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту

Кафедра Інформатики

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Побізінському Олександру Миколайовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка мультисервісної платформи для аналітики успішності спортсменів

затверджена наказом університету від 19 травня 2025 року № 381Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 21 травня 2025 р.
3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, результати аналізу ринку аналітично-спортивних систем. JavaScript, фрейморки React, Spring Boot, база даних PostgreSQL, брокер повідомлень RabbitMQ, авторизаційний сервіс Supabase.

4. Перелік питань, що потрібно опрацювати в роботі
1. Аналіз предметної області у галузі спортивно-аналітичних систем.
2. Розробка інтерфейсу для збору та аналітики даних.
3. Реалізація вебінтерфейсу для збору та аналітики спортивних даних.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри)актуальність спортивно-аналітичних систем, огляд існуючих аналітичних-систем, постановка задачі, проектування та розробка мультисервісної платформи для аналітики успішності, зображення інтерфейсу.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	07.04.2025	
2	Аналіз завдання, підбір літератури	08.04.25-10.04.25	
3	Аналіз літератури з досліджуваної проблеми	11.04.25-14.04.25	
4	Аналіз технічних засобів	15.04.25-20.04.25	
5	Розробка методу	21.04.25-27.04.25	
6	Програмна реалізація	28.04.25-11.05.25	
7	Оформлення пояснювальної записки	12.05.25-20.05.25	
8	Перевірка на нормоконтроль	21.05.25-01.06.25	
9	Перевірка на плагіат	21.05.25-01.06.25	
10	Рецензування	21.05.25-01.06.25	
11	Підготовка презентації та доповіді	21.05.25-18.06.25	
12	Занесення роботи в електронний архів	02.06.25-18.06.25	
13	Попередній захист кваліфікаційної роботи	02.06.25-18.06.25	

Дата видачі завдання 7 квітня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ _____ асист. Кобилін І. О._____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 62 с., 9 табл., 37 рис., 1 дод., 30 джерел.

ВЕБПЛАТФОРМА, МІКРОСЕРВІСИ, SPRING, REACT.

Об’єктом роботи є спортивні дані, що включають результати виступів спортсменів у змаганнях з підводного синхронного плавання.

Метою роботи є розробка мультисервісної платформи для збору, обробки та аналітики спортивних результатів, що дозволяє оцінювати динаміку успішності спортсменів у різних видах спорту.

У роботі проведено аналіз існуючих підходів до зберігання та візуалізації спортивних даних, розроблено архітектуру мультисервісної системи, яка включає 2 окремі сервіси для обробки даних з підводного синхронного плавання. Обидва сервіси забезпечують збір статистики, та виведення ключових метрик як для спортсменів, так і для команд.

У результаті роботи реалізовано платформу, яка дозволяє спортсменам, та тренерам отримувати деталізовану аналітику виступів спортсменів, виявляти прогрес, або зниження форми, і також розуміти, куди треба рухатися далі на основі отриманих даних.

WEB PLATFORM, MICROSERVICES, SPRING, REACT.

The object of this work is sports data, including the performance results of athletes in underwater synchronized swimming and volleyball competitions. The aim of the work is to develop a microservices-based platform for collecting, processing, and analyzing sports results, which allows tracking and evaluating the performance dynamics of athletes across different sports.

The work includes an analysis of existing approaches to storing and visualizing sports data, and the development of a microservices architecture consisting of two separate services: one for underwater synchronized swimming and one for volleyball. Both services handle data collection and provide key performance metrics for both individual athletes and teams.

As a result of the work, a platform has been implemented that allows athletes, coaches, and sports analysts to access detailed analytics on athlete performance, identify improvements or declines in form, and understand how to move forward based on data-driven insights.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API – Application Programming Interface (інтерфейс програмування застосунків)

UI – User Interface (користувацький інтерфейс)

DB – Database (база даних)

JSON – JavaScript Object Notation (формат обміну даними)

REST – Representational State Transfer (архітектурний стиль взаємодії компонентів у мережі)

DTO – Data Transfer Object (об’єкт передачі даних)

MQ – Message Queue (черга повідомлень)

TS – Team Statistics (командна статистика)

UI – User Interface (користувацький інтерфейс)

XSS – Cross Site Scripting (міжсайтовий скриптинг)

CSRF – Cross Site Request Forgery (міжсайтова підробка запиту)

ВСТУП

У сучасному світі цифрових технологій аналітика даних відіграє ключову роль у прийнятті зважених рішень у різних сферах, а саме: бізнес, логістика, і зокрема у спорті. З розвитком професійного спорту та підвищенням конкуренції серед команд і окремих спортсменів, зростає потреба глибокій, інформативній та своєчасній оцінці. Особливе значення набуває систематизація результатів змагань, виявлення динаміки розвитку спортсменів, наприклад із своїми минулими результатами.

Наразі більшість спортивних організацій стикаються із відсутністю наявності вебзастосунку, який міг би повністю, не фрагментовано подати статистику. Це стосується усіх видів спорту.

Актуальність розробки мультисервісної вебплатформи для аналітики спортивних досягнень полягає у створенні універсального інструменту, який дозволить збирати, обробляти та візуалізувати дані змагань у зручній формі. (Наприклад самі судді додають дані через застосунок, або надається excel файл у потрібному форматі, який зчитує данні по потрібному спорту.) Така система має забезпечувати гнучкість у роботі з різними видами спорту, мати гарну архітектуру для подальшого розвитку застосунку, а також підтримку сучасних інструментів, технологій.

Метою цього проєкту є розробка сучасної платформи, яка дозволяє тренерам, спортсменам та суддям отримувати(або додавати) структуровану інформацію про результати виступів, виявляти тенденції у зміні форми спортсменів, оцінювати ефективність тренувань, та приймати рішення щодо подальшого розвитку на основі даних. В основі платформи – мікросервісна архітектура з окремими сервісами для кожного виду спорту, реалізована з використанням технологій Spring та React, що забезпечує надійність, архітектурну масштабованість і гарний UI.

1 ОГЛЯД СТАНУ ПРОБЛЕМИ І ПОСТАНОВКА ЗАДАЧІ

1.1 Потреби спортивної галузі та аналітичних користувачів

Сучасний спорт стрімко розвивається, а збір та аналіз даних стає невіддільною частиною тренувального процесу, щоб було на що опиратися при підготовці до подальших змагань. Тренери, та самі спортсмени, дедалі більше покладаються на об’єктивні дані для побудови стратегії, щодо тренування своїх підопічних [1].

Одна з ключових проблем полягає, в майже повністю відсутньому доступу до способів, та інструментів збирання уніфікованої інформації, опираючись на яку можно проводити подальший аналіз. Різні види спорту, як-от наприклад підводне синхронне плавання, мають свої специфічні метрики ефективності, і це вимагає гнучких аналітичних рішень, що здатні адаптуватися під індивідуальні потреби кожного виду спорту [2].

Для тренерів важливо виявляти швидкість зростання спортсмену, і також виявляти слабкі місця, та робити аналіз динаміки змін фізичних і технічних показників.

У зв’язку з цим, зростає попит на створення сучасної вебплатформи, що поєднує можливість аналітики, інтерактивної візуалізації із гарним UI, та можливості додавати, та оновлювати дані. Розробка мультисервісної системи, яка об’єднує спортивні метрики з різних дисциплін в одному середовищі, є надзвичайно актуальною задачею, що відповідає викликам цифрової епохи у спорті.

1.2 Аналіз застосування технологій обробки даних у спортивній аналітиці

Розвиток спорту, зростання кількості змагань, інших видів спорту, та індивідуальний підхід до тренувального процесу кожного із спортсменів, або команди, та інтерес який невпинно зростає до діджіталізації, вимагають нових підходів до обробки та аналізу спортивних даних. Особливої актуальності це набуває у майже у всіх видах спорту, (у нашому разі це синхронне підводне плавання). Саме тому, ефективна обробка даних стає основою для прийняття рішень для тренерського штабу, який зацікавлений у подальшому розвитку своїх спортсменів.

Одним з ключових завдань є автоматизований збір результатів та статистичних показників спортсменів. Часто така інформація може міститися у цифрових звітах, наприклад у CSV форматі. Це створює потребу у єдиному механізмі перетворення структурованої інформації у цифровий формат, із яким далі можна зручно працювати [3].

Важливою частиною сучасної аналітики є зручна візуалізація спортивних показників, яка дає змогу наочно представляти динаміку результатів спортсменів або команд у вигляді графіків, діаграм, інтерактивних таблиць. Такі інструменти особливо зручні для тренерів, яким необхідно розробляти стратегії для тренування, визначення слабких зон, над якими треба попрацювати. Реалізація візуалізації може здійснюватися за допомогою таких фронтенд-технологій, як React, у поєднанні з бібліотеками на кшталт MUI/NIVO-Charts, що дозволяють створювати гнучкі й адаптивні графіки.

Серверна частина платформи повинна бути достатньо гнучкою із можливістю подальшого масштабування, да повинна надавати змогу оброблювати великі обсяги даних, що особливо актуально в умовах багатокористувацької системи. Тут доцільним є використання мікросервісної архітектури, яка передбачає поділ функціональності платформи на окремі

сервіси. Наприклад, окремий сервіс може відповідати за обробку даних з підводного плавання тощо. За допомогою такого розподілення, ми зможемо розвантажити нашу платформу, для її подальшого стабільного функціонування. Для створення такої архітектури ідеально підходить фреймворк Spring, що дозволяє створити масштабовану, мікроскервісну, та надійну систему [4].

Також, дуже важливе забезпечення зручності введення нових даних, можливість редагування показників ручним методом, а також функціональність для перегляду історичних даних. Усе це дозволяє створити універсальну платформу, яка буде ефективною для любих видів спорту, та спортсменів, надаючи їм можливість бачити свій розвиток на інтерактивних графіках.

1.3 Аналіз актуальності та доцільності розробки онлайн сервісу

У сучасному світі більшість сфер життя автоматизуються. Спорт не є винятком – все більше спортсменів, тренерів та спортивних організацій користуються цифровими інструментами для зручності та покращення результатів. В будь-який час, і з будь-якої точки світу, спортсмени які активно приймають участь у замаганнях, зможуть отримати доступ до онлайн сервісу, і скористатися ним.

Як індивідуально для спортсменів, так і для команд, аналітика у є дуже важливою частиною тренувального процесу, яка сприяє подальшому розвитку. Але, як і більшість сфер у нашому житті, спортивна галузь залишається майже не автоматизованою. Наприклад, тренери постійно змушені самостійно збирати дані про виступи спортсменів, вести таблиці в Excel або нотатках, що займає багато часу, і є не дуже зручним.

Мало відомих видів спорту, наприклад як підводного синхронного плавання, це стосується найбільше, оскільки це не є загально поширеним видом спорту. Саме тому, такі види спорту, часто залишаються без якісних

цифрових сервісів, де можна було б зберігати результати змагань, і основуючись на цих даних проводити зручний аналіз.

Завдяки розробці спеціалізованого онлайн сервісу, ми могли би значно спростити життя як для спортсменів, так і для тренерів. Наприклад, за його допомогою можна було б:

- швидко додавати результати виступів спортсменів;
- переглядати статистику у зручній формі;
- бачити прогрес кожного спортсмена або команди;
- автоматично формувати графіки та таблиці.

В сучасних реаліях розвитку цифрових технологій потреба у зручному цифровому інструменті, який завжди буде під рукою, зростає. Перевагою таких інструментів є зручність та швидкість у використанні. Таким чином, створення подібного сервісу являється актуальною задачею. Окрім зручності, це спосіб зробити спорт більш доступним і сучасним. Онлайн сервіс допоможе об’єднати всі необхідні функції в одному місці та дозволити користувачам звертатися до даних надійно, та безпечно.

Таким чином, розробка мультисервісної онлайн платформи для аналітики спортсменів є логічним і доцільним рішенням, яке відповідає сучасним викликам і потребам у сфері спорту. Це не тільки спрощує життя спортсменам, але й допомагає приймати правильні і зважені рішення.

1.4 Аналіз існуючих платформ та сервісів аналітики у спорті

У світі спорту, де кожна дія може стати вирішальною, аналітика стала важливою складовою ефективного управління тренувальним і змагальним процесом. Напротязі останніх років на цифровому ринку з’явилася немала кількість інструментів і платформ, які спеціалізуються на роботі із спортивними даними. Проте більша частина орієнтована на всесвітньо популярні види спорту (футбол, хокей, тощо). Таким чином, поза увагою залишаються специфічні види спорту, як підводне синхронне плавання у

нашому випадку. Саме тому, перед створенням вебзастосунку слід проаналізувати конкурентів у цьому напрямку.

1.4.1 Hudl – аналіз платформи

Hudl – це одна з найпоширеніших платформ, що надає послуги для відеоаналізу виступів спортсменів, тактичного розбору матчів та збереження спортивної статистики. Починаючи шкільними секціями та закінчуючи професійними клубами, hudl набула широкого попиту серед своїх користувачів.

Сервіс дозволяє завантажувати відео матчів, автоматично чи вручну розмічати важливі моменти (гол, помилка, пас, подача), ділитися уривками з гравцями та аналізувати дії кожного члена команди. Переваги Hudl:

– Hudl має інтегрований інтсрумент відеоналізу, який надає користувачам можливість створювати відеокліпи ключових моментів. Опираючись на них, можна провести детальну аналітику, додати коментарі, і все це завдяки інтерактивним інструментам, які надає веб-платформа. Тренери можуть позначати дії гравців, графічні елементи (стрілки, схеми), що значно спрощує пояснення різних ситуацій;

– зручна командна взаємодія. Гравці можуть отримати персональний доступ до своїх ключових моментів під час гри, переглядати зауваження тренера, робити висновки і також переглядати матчі суперників;

– аналітична статистика. У залежності від обраної підписки, Hudl дозволяє формувати статистику на основі відео. Завдяки такому потужному інструменту, основна частина тренерської роботи автоматизується;

– мобільність і хмарне зберігання. Hudl крос-платформений застосунок. Додатком можна скористатися як через браузер так і мобільні платформи, що дає користувачам значну мобільність у використанні застосунку. Варто зазначити, всі дані зберігаються в хмарі. Таким чином веб-платформа буде займати незначну кількість місця на вашому смартфоні [5];

– масштабованість. Платформа підходить як для шкільних команд, так і для професійних клубів. Hudl має різні тарифні плани, адаптовані під обсяг потреб користувачів;

Недоліки Hudl:

– обмеження за видами спорту. Hudl набув широкого використання у популярних видах спорту, таких як футбол і баскетбол. Наприклад, для менш відомих видів спорту, застосунок не передбачає шаблонів аналітики, або візуального розмічення;

– висока вартість для непрофесійних команд. Хоча Hudl пропонує базовий безкоштовний пакет, але найголовніша частина функціоналу із відео нарізками, інтерактивністю, та необмеженістю завантажень відео – потребує іншого тарифного плану, який може потребувати значних матеріальних витрат. Саме тому, наша платформа може бути поганим вибором для шкільних гуртків або невеликих клубів;

– залежність від швидкого. Інтернету Оскільки платформа працює у хмарі, для її ефективного використання необхідно стабільне і швидке з’єднання. Завантаження відео може займати час, а перегляд - "підвисати" при слабкому Інтернеті;

– складність UI для новачків. Для користувачів без технічної підготовки або досвіду з аналітичними платформами, інтерфейс Hudl може здаватися дещо перевантаженим;

– обмежена кастомізація. Hudl має фіксовану структуру певних шаблонів, що не завжди дозволяє їх змінювати під певну специфіку заданого виду спорту.

1.4.2 Аналіз платформи Dartfish

Dartfish – це одна з найстаріших і найбільш відомих платформ для відеоаналізу у спорті, яка вже багато років використовується національними збірними, професійними клубами, олімпійськими командами та навіть

освітніми установами. Головна особливість Dartfish полягає в акценті на детальній роботі з відео – розкадруванні, повільному повторі, порівнянні рухів, накладанні графічних елементів. Такий підхід особливо ефективний у видах спорту, де потрібно звертати увагу на технічні деталі рухів спортсмена: гімнастиці, плаванні, легкій атлетиці, боротьбі, тенісі.

Платформа має глибокі можливості з анотації відео, що дозволяє не лише відмічати моменти виступу, а й створювати персоналізовані кліпи для кожного спортсмена. Наприклад, у випадку аналізу синхронного плавання можна обрати фрагмент виступу, прокоментувати його, накласти траєкторії руху або фіксувати відхилення від технічного стандарту. Це особливо корисно для тренерів, які хочуть продемонструвати спортсмену помилки безпосередньо на візуальному прикладі.

Разом з тим, Dartfish не обмежується лише відеоаналізом. Сервіс підтримує роботу зі статистичними даними, зокрема ручний і частково автоматизований підрахунок показників. Це дозволяє поєднувати кількісні метрики з візуальним розбором. У волейболі, наприклад, можна підраховувати точність передач, успішність атак, ефективність блокування, а потім зв’язувати ці показники з відеоепізодами. Така гнучкість – один із головних плюсів платформи.

Проте Dartfish має і свої обмеження. Вона, насамперед, розрахована на користувачів із технічним досвідом та глибоким розумінням процесу аналізу. Інтерфейс доволі складний, а процес навчання платформі може зайняти чимало часу. Крім того, Dartfish є комерційним програмним продуктом із закритим вихідним кодом, що унеможливлює глибоку адаптацію або кастомізацію під потреби конкретної команди або школи. Також вона не передбачає гнучкої багатосервісної архітектури – усі функції зосереджені в одному застосунку, що ускладнює масштабування чи інтеграцію з іншими цифровими системами, наприклад, вебінтерфейсами або мобільними рішеннями.

Ще однією проблемою є слабка підтримка автоматичного імпорту даних з інших джерел – таблиць Excel, баз змагань, електронних форм. У випадку, коли потрібно об’єднати відео з результатами змагань, синхронізувати їх із розкладом або історією тренувань – користувачеві доводиться робити це вручну, що знижує ефективність у порівнянні з платформами, що мають модульну структуру та API.

Підсумовуючи, Dartfish є потужним, але досить спеціалізованим інструментом для аналізу техніки рухів і візуального розбору дій спортсмена. Проте в контексті створення універсальної мультисервісної платформи, як описано в темі даної кваліфікаційної роботи, така система не забезпечує необхідної гнучкості, розширюваності та відкритості, що обмежує її придатність для широкого кола користувачів – особливо в умовах аматорського спорту, навчальних закладів або секцій, які потребують простих, адаптивних, економічно досяжних рішень.

1.4.3 Аналіз платформи Wyscout

Wyscout – це одна з найпотужніших і найвідоміших платформ у сфері спортивної аналітики, що спеціалізується на футболі. Вона надає користувачам доступ до гігантської бази даних відеозаписів матчів, статистики гравців, тактичних схем та тренерських розборів. Платформою активно користуються футбольні клуби, скаути, агенти, тренери та навіть журналісти. Її головна особливість полягає в тому, що вона поєднує у собі одразу три напрямки: аналітику, пошук талантів і глибокий відеоархів.

Wyscout дає змогу користувачам не лише переглядати матчі та дії гравців у розрізі – передачі, удари, перехоплення тощо – але й порівнювати їх між собою за багатьма параметрами. Для скаутів це надзвичайно зручний інструмент: можна знайти гравця в конкретному віці, з певними характеристиками і переглянути лише його дії за весь сезон – без потреби дивитися повні матчі. Для тренерів і аналітиків це можливість глибше

аналізувати гру суперників, моделі поведінки команд, індивідуальні сильні та слабкі сторони гравців.

Проте, незважаючи на всю технологічну складність, Wyscout фактично є закритою системою, що працює лише з власною базою даних. Це означає, що користувачі не можуть завантажити відео своїх матчів або створити аналітику на основі локальних змагань, якщо ці дані не входять у глобальну базу Wyscout. Тобто для невеликих клубів, аматорських ліг або нетипових видів спорту (наприклад, синхронного плавання чи навіть волейболу), платформа виявляється непридатною.

Ще одним фактором є висока вартість доступу до повного функціоналу – підписка коштує доволі дорого, і це робить її недоступною для шкіл, студентських команд чи спортивних гуртків. Більшість клубів, які працюють з Wyscout, мають власні бюджети, аналітичні штаби та досвід користування подібними сервісами. Це створює ситуацію, коли аналітика високого рівня стає привілеєм виключно професіоналів.

Також варто зазначити, що Wyscout, попри свої розширені можливості, не дозволяє гнучкої персоналізації. Наприклад, неможливо створити власну структуру оцінки гравців або змінити набір метрик відповідно до конкретної тактики. Усі параметри фіксовані та стандартизовані під загальноєвропейський футбольний контекст.

Таким чином, Wyscout – це приклад високотехнологічного, але закритого та обмежено доступного рішення, яке чудово працює у масштабі великого професійного спорту, проте не підходить для індивідуальних, малопоширених або аматорських видів спорту. Цей факт ще раз підтверджує актуальність створення гнучкої мультисервісної платформи, яка дозволяє працювати з власними даними, адаптується під потреби конкретної дисципліни (зокрема підводного синхронного плавання чи волейболу), і є доступною для звичайних користувачів без значних фінансових чи технічних ресурсів.

1.5 Постановка задачі

З урахуванням стрімкого розвитку вебтехнологій у всіх сферах, розробка спеціалізованого рішення для автоматизування збіру, зберігання та аналізування даних про виступи спортсменів наразі являється актуальною. Сучасні сервіси аналітики здебільшого орієнтовані на найпоширеніші командні види спорту, не враховуючи особливостей менш популярних дисциплін, таких як підводне синхронне плавання. Крім того, існуючі платформи або занадто дорогі для широкого кола користувачів, або мають закриту архітектуру, яка не дозволяє адаптувати функціонал під специфіку конкретного виду спорту чи потреб користувача.

У рамках роботи передбачається створення універсальної вебплатформи, що об’єднує можливості аналітики даних з підводного синхронного плавання та волейболу. Система повинна забезпечувати зручний інтерфейс для введення інформації про змагання, результати виступів, індивідуальні й командні показники, а також формувати наочні звіти, які допомагають відслідковувати успішність спортсменів у динаміці. Важливим аспектом розробки є побудова архітектури платформи на основі мікросервісного підходу, що дозволяє гнучко керувати окремими складовими системи, масштабувати її в майбутньому та забезпечити незалежність у роботі кожного модуля.

Ключовим завданням є не лише створення технічно функціонального програмного продукту, а й врахування потреб різних категорій користувачів - від тренера до спортсмена, які часто бувають далекі від світу цифрових технологій, але потребують зручний та дружелюбний застосунок для для введення власних статистик. Крім того, у процесі реалізації необхідно передбачити можливість візуалізації даних у вигляді таблиць та графіків, що зробить результати аналітики зрозумілими навіть для непідготовленого користувача. Інтерфейс який буде реалізовано таким чином, що все буде інтуїтивно зрозуміло, збір та обробка статистики ефективним шляхом,

дозволить платформі не лише фіксувати виступи команд та спортсменів, але й реально впливати на тренувальний процес та покращення спортивних даних.

Основними завданнями, які потрібно вирішити, є наступні:

– розробка зручного інтерфейсу: створення головної сторінки із детальною інформацією щодо досягнень спортсмену. Можливість перегляду своїх результатів, та порівняння із іншими користувачами. Сторінка для додавання результатів виступів, створення нових змагань;

– розробка бізнес-логіки: перевірка правильності введених користувачами даних, створення нових записів у базі даних при успішній операції;

– розробка зв’язку з базою даних: актуалізація та створення потрібних таблиць. Розробка процедур для переіврки цілісності інформації та її збереження у базі даних;

– подальша оцінка якості розроблених рішень на основі тестування функціювання системи.

2 АНАЛІЗ ТА РОЗРОБКА МУЛЬТИСЕРВІСНОЇ ПЛАТФОРМИ

2.1 Проектування системи

Правильне проектування системи є невід’ємною частиною розробки будь-якого вебзастосунку. Визначення, які підлягають під термін успішне проектування:

– чітке формулювання функціональних та нефункціональних вимог. Цей етап розробки забезпечує відповідність очікуванням замовника, і гарантує технічну обґрунтованість обраних технологій;

– архітектура системи. Який підхід використовувати: мікросервісний, монолітний;

– масштабованість. Це здатність проєкту впоратися із більшим навантаженням на платформу, і можливістю системи адаптуватися до нових функціональних вимог;

– проєктування бази даних. Моделювання та оптимізація сутностей. Визначення правильних зв’язків між ними. Оптимізація запитів, та індексація таблиць;

– інтерфейс користувача (UI). Простий та інтуїтивний інтерфейс, під час використання якого не виникне жодних труднощів. Адаптивність дизайну під мобільні пристрої, та якісне тестування застосунку. Усі ці пункти являються невід’ємною частиною для побудови позитивного користувацького досвіду, під час використання платформи;

– веббезпека. Https протоколи для більш надійного шифрування. Захист від XSS та CSRF атак. Єдина точка аутентифікації та авторизації із використанням Oauth2 [6];

– тестування. Unit (тестування окремих модулів коду програми) та інтеграційне тестування. Налаштування CI/CD(continuous integration та continuous deployment) для автоматизованих перевірок;

– впорядковане введення документації. Наявність технічної документації, як правильно використовувати існуюче API. Опис структури та архітектури системи. Наявність README файлів для розробників;

– моніторинг. Правильне використання сервісів моніторингу, наприклад такі як Kibana, забезпечить зручний інтерфейс для відстеження логів, алертів, помилок які будуть надходити із нашого застосунку.

Наступний крок – уточнення функціональних вимог. Чіткий перелік будує основу для подальшого проектування логіки, інтерфейсу та баз даних для кодових модулів [7].

Функціональні вимоги:

– можливість бути авторизованим на веб-платформі, і мати відповідні права до своєї ролі. У нашому застосунку буде фігурувати 2 різні ролі: спортсмен та тренер;

– базовий функціонал для перегляду своєї статистики. Перегляд останніх змагань у яких людина приймала участь, графіки для введення власної статистики (кількість виграних змагань, порівняння із іншими людьми);

– пошук та перегляд детальної інформації щодо усіх зареєстрованих спортсменів. На сторінці користувача має відображатися інформація пов’язана із його спортивною кар’єрою та належність до існуючої команди;

– можливість зчитування CSV/EXCEL документів із конкретною структурою записів. Наша платформа має надавати можливість швидкого зчитування, та правильного збереження будь-якої інформації пов’язаної із спортивними змаганнями у базу даних [8];

– формування pdf файлів для перегляду статистики. У тренерів має бути змога формування pdf документів, які несуть у собі інформацію щодо поточного змагання;

– спроможність створювати нові змагання конкретного типу, і додавати до них особисті результати спортсменів;

– створення тренерами нових команд, та редагування існуючих [9].

Технології для розробки:

– мови програмування: Java/TypeScript;

– бази даних: Postgres;

– комунікація між модулями за допомогою брокеру месседжей RabbitMQ;

– серверна частина: Spring Boot;

– UI частина: React;

– автоматизація та розгортання застосунку: Docker;

– середовище розробки:IntelliJ IDEA.

2.2 Огляд архітектурних підходів

Архітектура вебзастосунку визначає спосіб організації та взаємодії наявних компонентів. У світі існує багато архітектурних паттернів, але у цьому розділі ми розглянемо і порівняємо 2 основних види, це монолітний та мікросервісний підхід.

Монолітна програма будується єдиним модулем, де уся бізнес логіка та інтерфейси щільно пов’язані між собою. Із плюсів варто відзначити, що під час розробки монолітний застосунок як правило розроблюється набагато легше при старті самого проєкту. Також, можемо відокремити легкість підтримки одного зібраного контейнеру і його подальшого розгортання. Але є інша сторона медалі, масштабування програми. По-перше набагато складніше робити будь-які зміни у надзвичайно великому модулі із складною і наплутаною бізнес-логікою. Наприклад, додавши якісь зміни до існуючого монолітного застосунку, розробник без наміру може зламати існуючий функціонал. І звичайно надалі, масштабування буде збільшувати кількість часу потрібного для тестування нового функціоналу[10].

Розглядаючи мікросервісний підхід, можемо виділити легкість та гнучкість у подальшому розширенні програмного коду. Набагато легше

тестувати невеличкі модулі застосунку. Із таким підходом, можливість зачепити існуючий функціонал під час розробки нових фічей становиться майже неможливою. І варто зазначити, що розбиття архітектури на невеличкі модулі розгружує системне навантаження, що є вагомим досягненням [11].

Більш детальніше порівняння архітектурних паттернів за допомогою таблиці(табл. 2.1).

Таблиця 2.1 – Порівняльний аналіз архітектурних підходів

Критерій	Монолітна Архітектура	Мікросервісна архітектура
Структура	Уся логіка в одному додатку	Кілька незалежних сервісів, кожен виконує свою роль
Розгортання (деплой)	Цілісний застосунок, зібраний в один артефакт	Кожен сервіс розгортається окремо
Розробка	Простіше на старті, єдиний код для всієї логіки	Кожен сервіс - окремий репозиторій, стек, команда
Тестування	Тестується як єдине ціле	Окремо для кожного сервісу + інтеграційні тести
Комунікація	Через методи/класи всередині програми	HTTP/REST, gRPC, черги (RabbitMQ, Kafka)
Швидкість розробки	Швидко на початку, складніше при масштабуванні	Складніше на старті, але краще масштабуються
Масштабування	Масштабується вся система одразу	Масштабується лише потрібний сервіс
Надійність	Один баг може вплинути на всю систему	Падіння одного сервісу не є падінням всієї системи
Розгортання командою	Часто працює одна команда над всім	Кілька команд можуть працювати паралельно

2.3 Огляд існуючих мікросервісів

У системі наявні 2 мікросервіси:

– сервіс із головним функціоналом як для спортсменів, так і для тренерів;

– сервіс який відповідає за зчитування/формування CSV/EXCEL файлів.

Основна задача головного сервісу: забезпечення зручного перегляду та пошуку інформації, статистики. У ньому імплементована вся бізнес-логіка, із якою можуть взаємодіяти користувачі. Задача сервісу для роботи із CSV/EXCEL файлу полягає лише у зчитуванні/формуванні даних і їх відправці до користувача/брокеру месседжей (рис. 2.1).

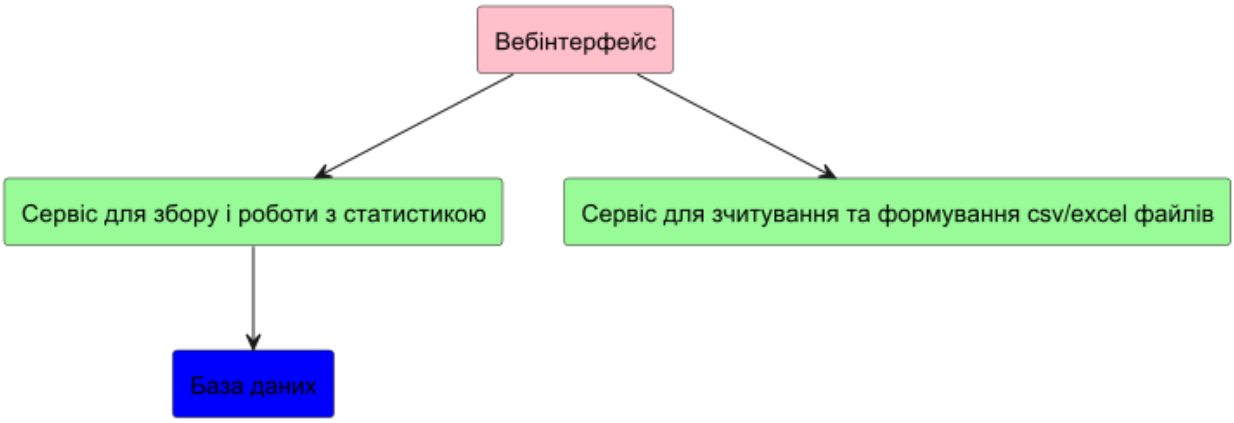


Рисунок 2.1 – Початкова версія мікросервісної архітектури

2.4 Огляд та проектування бази даних

Бази даних – це організовані колекції даних, які дозволяють зберігати, керувати та отримувати інформацію. У кожній системі правильна організація бази даних є надзвичайно важливим фактором. Гнучкість і масштабованість застосунку, напряму залежить від структури і зв’язків наших сутностей [12].

Бази даних поділяються на два види реляційні (SQL) та нереляційні (NOSQL). Нижче у таблиці проведений аналіз та порівняння цих двох підходів (табл. 2.2).

Таблиця 2.2 – Порівняльний аналіз типів баз даних

Критерій	SQL (реляційні БД)	NoSQL (нереляційні БД)
Структура даних	Строга, фіксована схема (таблиці, стовпці)	Гнучка або відсутня схема (документи, ключ-значення)
Мова запитів	SQL (Structured Query Language)	Залежить від БД (MongoDB – власний API, тощо)
Транзакції / ACID	Підтримка ACID (надійні транзакції)	Може бути eventual consistency, не завжди ACID
Масштабування	Вертикальне (потужніший сервер)	Горизонтальне (додавання серверів)
Продуктивність	Висока при складних зв’язках, аналітиці	Висока при великих обсягах, простих запитах
Ідеальні для	Фінансові системи, ERP, CRM, звітність	API, IoT, real-time apps, документи, Big Data
Складність змін	Зміна схеми вимагає міграцій	Дані можуть змінюватися без міграцій

У зв’язку з тим, що у нас є чітка і складна схема формату змагань і вона ніколи не змінюється, і завдяки цим даним тренера повинні мати змогу робити звіти, у розробці даного застосунку було передано перевагу реляційній базі даних Postgres.

Postgres – це потужна реляційна система управління базами даних яка в свою чергу має багато переваг, а саме:

– безкоштовний із відкритими вихідниками код, над яким постійно працюють, оновлюють та підтримують;

– велика спільнота;

– повна підтримка SQL стандартів;

– підтримка JSON/JSONB формату;

– наявність системи індексації таблиць(B-tree, Hash, GiST, GIN, BRIN).

Розробка структури бази даних для інформаційної системи управління спортивними змаганнями потребує чіткого визначення сутностей, атрибутів та зв’язків між ними. Нижче наведено детальний опис класів, атрибутів та зв’язків, що представлені в UML-діаграмі [13].

Основною сутністю є клас User, що представляє користувача (спортсмена) системи. Його атрибути включають id, firstName, lastName, birthDate. Атрибут id є первинним ключем і унікально ідентифікує кожного користувача. Ім’я (firstName), прізвище (lastName) та електрона пошта (email) є текстовими значеннями. Вони перевіряються за встановленими шаблонами і вимогами до довжини, а електрона адреса є унікальною для кожного користувача яка у свою чергу ідентифікує користувачів. BirthDate зберігає дату народження користувача. Зв’язок User з Gender описується як асоціація, що показує, що кожен користувач має одну стать – або MALE, або FEMALE (enum). Більш детальніше у таблиці 2.3.

Таблиця 2.3 – Схема таблиці користувачів (User)

Зміст поля	Назва поля	Тип даних	Нотатки
1	2	3	4
Ідентифікатор спортсмена	id	BIGINT	
Ім’я користувача	firstName	VARCHAR	
Електрона пошта	email	VARCHAR	Завдяки ній користувач може увійти у свій аккаунт на порталі
Прізвище спортсмена	lastName	VARCHAR	

Продовження таблиці 2.3

1	2	3	4
День народження спортсмена	birthDate	DATE	
Розряд спортсмена	rank	VARCHAR	Розряд в залежності від рівня професіоналізму спортсмену
Стать	gender	VARCHAR	

Клас CompetitionResult зберігає інформацію про кожен виступ спортсмена. Атрибути включають id, entryTime (початковий заявлений час), result (результат змагання), lane (номер доріжки) і heat (номер запливу). Додатково, кожен результат має категорію (Category enum A, B, C, D, E), яка описує вікову або спортивну групу, та спортивний розряд (Rank), який також є enum (від JUNIOR до HMS). Має 2 поля які посилаються на наступні таблиці: User та Competition. Зв’язок із таблицею User (багато до одного) утворений задля присвоєння результату конкретному спортсмену. Посилання на таблицю Competition (багато до одного) відповідає за ідентифікацію результату у конкретному змаганні. Більш детальніше у таблиці 2.4.

Таблиця 2.4 – Схема таблиці результатів змагань (CompetitionResult)

Зміст поля	Назва поля	Тип даних	Нотатки
1	2	3	4
Ідентифікатор результату змагання	id	BIGINT	
Вхідний час	entryTime	TIME	Час, на пропливання дистанції

Продовження таблиці 2.4

1	2	3	4
Результат	result	TIME	Час, за який спортсмен проплив задану дистанцію
Смуга	lane	INTEGER	Смуга, на якій зафіксований результат
Заплив	heat	INTEGER	Номер запливу, на якому був зафіксований результат
Розряд спортсмена	rank	VARCHAR	Розряд який був у спортсмена на момент виступу
Категорія	category	VARCHAR	
Ідентифікатор користувача	user	BIGINT	
Ідентифікатор змагання	competition	BIGINT	

Сутність Competition представляє спортивне змагання. Вона має атрибути: id, name, date, де id – унікальний ідентифікатор змагання, name – назва події, date – дата проведення. Змагання має тип, який визначається через enum CompetitionType, де можуть бути такі значення як SWIMMING_WITH_FINS або SCUBA_DIVING. Більш детальніше у таблиці 2.5.

Таблиця 2.5 – Схема таблиці змагань (Competition)

Зміст поля	Назва поля	Тип даних	Нотатки
Ідентифікатор змагання	id	BIGINT	
Назва змагання	name	VARCHAR	
Дата проведення	date	DATE	
Тип змагань	competitionType	VARCHAR	У нашому додатку існує 2 види плавання з ластами і підводне без ласт

Клас Team описує команду спортсменів, має атрибути id та name. Користувачі прив’язуються до команди через проміжну таблицю TeamUser, яка має атрибути id, userId та teamId. Це реалізація зв’язку багато до багато між User та Team. Таким чином, один користувач може бути у кількох командах, і одна команда може мати багато користувачів (табл 2.7 та 2.8).

Таблиця 2.6 – Схема таблиці команди (Team)

Зміст поля	Назва поля	Тип даних	Нотатки
Ідентифікатор команди	id	BIGINT	
Назва команди	name	VARCHAR	
Ідентифікатор тренера	trainer_id	BIGINT	Тренер, до якого відноситься команда

Таблиця 2.7 – Схема зв’язуючої таблиці команди (TeamUser)

Зміст поля	Назва поля	Тип даних
Ідентифікатор запису	id	BIGINT
Ідентифікатор спортсмена	userId	BIGINT
Ідентифікатор команди	teamId	BIGINT

Зв’язок між Trainer та Team є зв’язком один до одного: кожна команда має одного тренера, а кожен тренер закріплений за однією командою. Клас Trainer має атрибути id, uid (унікальний ідентифікатор), name. Більш детальніше у таблиці 2.8.

Таблиця 2.8 – Схема таблиці тренерів (Trainer)

Зміст поля	Назва поля	Тип даних	Нотатки
Ідентифікатор запису	id		
Ідентифікатор тренеру	uid		Унікальний ідентифікатор за допомогою якого тренер може залогінитись у додаток
Ім’я	name		

Загалом, система враховує складну ієрархію та множинність зв’язків, що забезпечує гнучкість та масштабованість моделі. Для реалізації складних зв’язків використано проміжну таблицю TeamUser. Наявність enum-типів дозволяє стандартизувати категорії, розряди, стать та тип змагання.

Система може бути використана для ведення обліку спортсменів, організації змагань, аналізу результатів та створення звітів по командам та тренерам. В майбутньому модель можна розширити, додавши нові типи змагань, підтримку мультиспортивних заходів та історію змін розрядів спортсменів [14].

Детальніше про всі зв’язки та елементи таблиць представлено на класовій діаграмі сутностей яка наведена на рисунку 2.2.

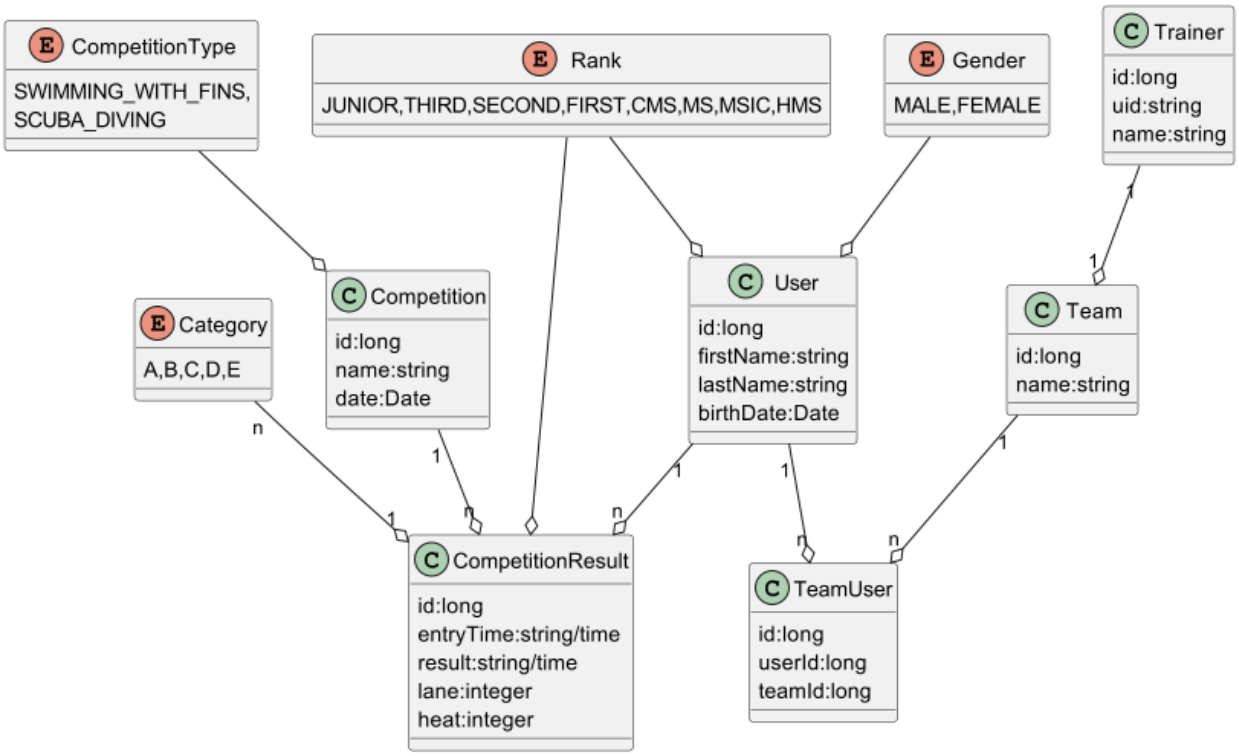


Рисунок 2.2 – Класова діаграма сутностей

2.5 Огляд авторизаційних сервісів

Supabase і Auth0 – це два популярні сервіси, які виконують різні, але дотичні функції в екосистемі сучасної веброзробки. Supabase позиціонує себе як відкритий аналог Firebase і забезпечує повний набір інструментів для розробки бекенду: базу даних, аутентифікацію, зберігання файлів, функції тощо. Auth0, з іншого боку, – це вузькоспеціалізований сервіс для ідентифікації та управління доступом, який фокусується виключно на безпечній аутентифікації, авторизації та керуванні користувачами [15].

Запропонована архітектурна філософія має наступний вигляд. Supabase – це повноцінний бекенд-платформний стек, побудований навколо PostgreSQL. Він орієнтований на розробників, яким потрібне централізоване, самодостатнє рішення з можливістю управління базами даних, запитами, функціями та автентифікацією в одному середовищі. Auth0 ж, навпаки, є вузьконаправленим сервісом, що надає авторитетний механізм авторизації OAuth2, OpenID Connect та інших протоколів, і ідеально підходить для

інтеграції з великими ентерпрайзними системами, хмарними додатками та зовнішніми провайдерами автентифікації [16].

Supabase надає Auth-сервіс на базі GoTrue (від Netlify), який підтримує email/password, magic link, OAuth2-провайдерів (Google, GitHub, Facebook тощо), та інтегрується з Postgres для керування сесіями. Це рішення добре підходить для додатків із невисокими вимогами до кастомізації механізмів авторизації, і його основною перевагою є тісна інтеграція з самою базою даних. Наприклад, у Supabase користувачі одразу мають відповідний запис у таблиці auth.users, і цю інформацію можна легко використовувати в SQL-політиках (Row Level Security). Auth0 натомість орієнтований на корпоративне використання з великою гнучкістю у правилах, тригерах (rules, actions), багатоетапній автентифікації (MFA), підтримкою десятків зовнішніх ідентифікаторів та керуванням правами через roles/permissions [17].

У плані безпеки Auth0 має очевидну перевагу завдяки розвиненій інфраструктурі захисту: автоматичне виявлення аномалій, MFA, підтримка enterprise SSO (SAML, LDAP, ADFS), логін через корпоративні облікові записи. Supabase же покладається на простішу модель безпеки, в якій безпека в основному реалізована на рівні доступу до таблиць, політик і JWT токенів. Це працює добре в більшості стандартних додатків, але може бути недостатньо для систем, які потребують жорстких правил доступу, зовнішньої автентифікації або федерації [18].

У плані керування користувачами Supabase забезпечує базовий та гнучкий інтерфейс: реєстрація, підтвердження email, відновлення паролю, OAuth. Панель адміністратора дозволяє переглядати користувачів і керувати їхніми даними. Auth0 же має розширену систему керування користувачами, включно з налаштуваннями блокування акаунтів, правилами rate limiting, логами аутентифікації, webhook-нотифікаціями, та деталізованим журналом дій користувачів (рис. 2.3 та рис. 2.4) [19].

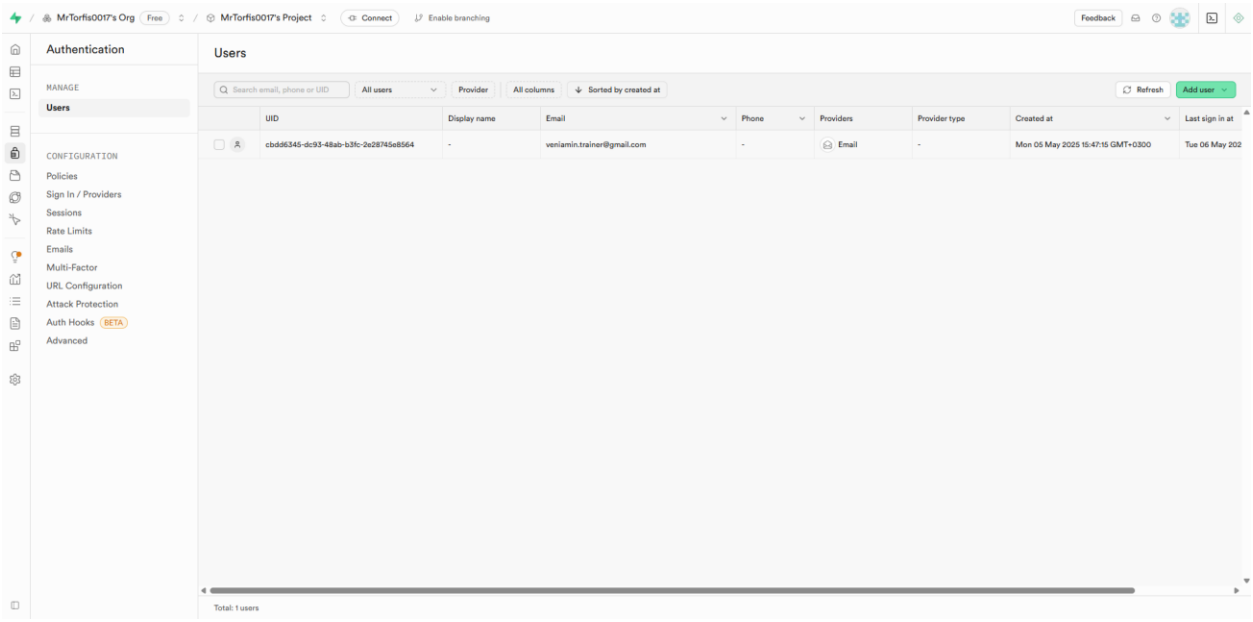


Рисунок 2.3 – Postgres база даних на стороні Supabase із усіма існуючими користувачами

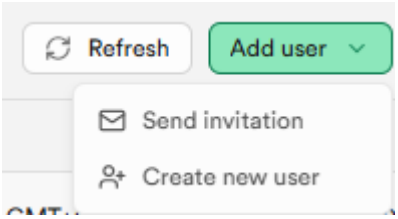


Рисунок 2.4 – Popup меню із можливими варіантами додавання користувачів

Далі розглянемо інтерфейс форми створення нового користувача. Ось опис його елементів:

– email address. текстове поле для введення електронної пошти користувача. У полі за замовчуванням показано приклад user.example@com;

– user Password. Поле для введення пароля користувача, при цьому введені символи приховані (замінені крапками);

– чекбокс Auto Confirm User. Якщо активований (як на скріншоті), користувач буде автоматично підтверджений, і йому не буде надіслано листа з підтвердженням (рис. 2.5).

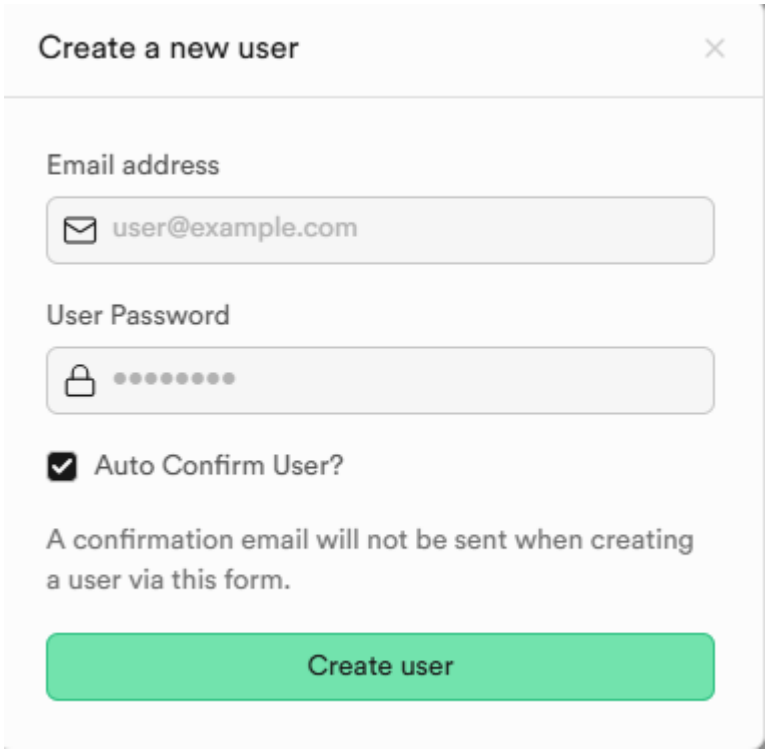


Рисунок 2.5 – Ручне додавання користувача

Щодо цінової політики: Supabase має щедрий безкоштовний план з включеним хостингом бази даних, файлового сховища, edge-функцій та auth сервісу. Платні плани орієнтовані на масштабованість, і ціноутворення базується на кількості запитів, обсязі зберігання та функціональних викликах. Auth0 має теж безкоштовний план, але при зростанні кількості користувачів і потребі в розширених можливостях (SSO, enterprise federation, MFA) ціни стрімко зростають, що може стати обмеженням для стартапів або малого бізнесу [20].

З погляду інтеграції Supabase – це частина цілісного підходу до бекенду, тому логін/реєстрація/авторизація – це лише один із компонентів, які розробник має у своєму арсеналі. Водночас, Supabase покладається на PostgREST і RLS для безпечного доступу до API, тому розробники повинні мати базові знання SQL для ефективної побудови системи доступу. Auth0 розроблений таким чином, щоб ізолювати авторизацію як окремий сервіс - із власним API, UI та логікою. Його легше підключити до стороннього бекенду (Spring, Node.js, .NET) і керувати без потреби змінювати бізнес-логіку [21].

Щодо кастомізації: Supabase обмежений в кастомізації аутентифікаційного UI, логіки логіну або обробки подій, хоча останнім часом вони додають edge-functions, які частково розв’язують це. У Auth0 кастомізація винесена у фронтові скрипти, rules та UI templates [22], що дозволяє побудувати складні сценарії логіну, реєстрації, кастомні email-шаблони та поведінку під час входу користувача в систему.

Щодо зберігання метаданих користувачів, Supabase дозволяє додавати додаткові атрибути через user_metadata або app_metadata, що зберігаються у JWT токені і базі даних. Auth0 має схожу модель, але з кращими API для редагування, пошуку та політик доступу до цих даних.

Підсумовуючи, якщо розробник створює проект із нуля, який потребує повноцінного бекенду і хоче мінімізувати кількість зовнішніх залежностей, Supabase – це чудовий вибір завдяки інтегрованому підходу та гнучкому SQL-контролю доступу [23]. Якщо ж мова йде про застосунки, де головне – безпечна, масштабована авторизація з високим рівнем контролю над користувачами, підтримкою enterprise-інтеграцій, багатофакторної автентифікації – Auth0 забезпечить вищу гнучкість і надійність, хоча й за більшу ціну.

3 РЕАЛІЗАЦІЯ МУЛЬТИСЕРВІСНОЇ ПЛАТФОРМИ ДЛЯ АНАЛІТИКИ УСПІШНОСТІ СПОРТСМЕНІВ

3.1 Розробка клієнтської частини системи

Основною метою програмного забезпечення є оптимізація процесу введення статистики спортсменів, та змагань для тренерського складу. Для ефективної роботи з системою вона потребує інтуїтивно зрозумілий та дружелюбний інтерфейс, саме тому розробка має починатися з клієнтської частини. Для створення клієнтської частини використовується мова JavaScript та бібліотека React, яка дозволяє побудувати вебінтерфейс [24].

Клієнтська частина платформи включає в себе наступні екрани (для спортсменів):

– сторінки авторизації та аутентифікації користувача;

– домашньої сторінки, де відображається уся корисна інформація щодо побудови порівняльних графіків, особистих досягнень;

– сторінка із усіма існуючими спортсменами та командами у базі;

– детальна інформація щодо конкретного змагання;

– детальна інформація щодо користувача. Зазвичай це особиста інформація, та належність до якоїсь команди.

Доступні екрани для тренерів:

– екран з детальною інформацією про керовану команду;

– модальне вікно для створення команди;

– модальне вікно для створення змагання;

– модальне вікно для додавання результату.

Кожна сторінка виконує свої індивідуальні задачі. Вони повинні правильно та доступно подавати інформацію. Розберемо більш детальніше наявні компоненти у застосунку, та за що вони відповідають.

Почнемо із самого початку, а саме сторінки для входу у систему.

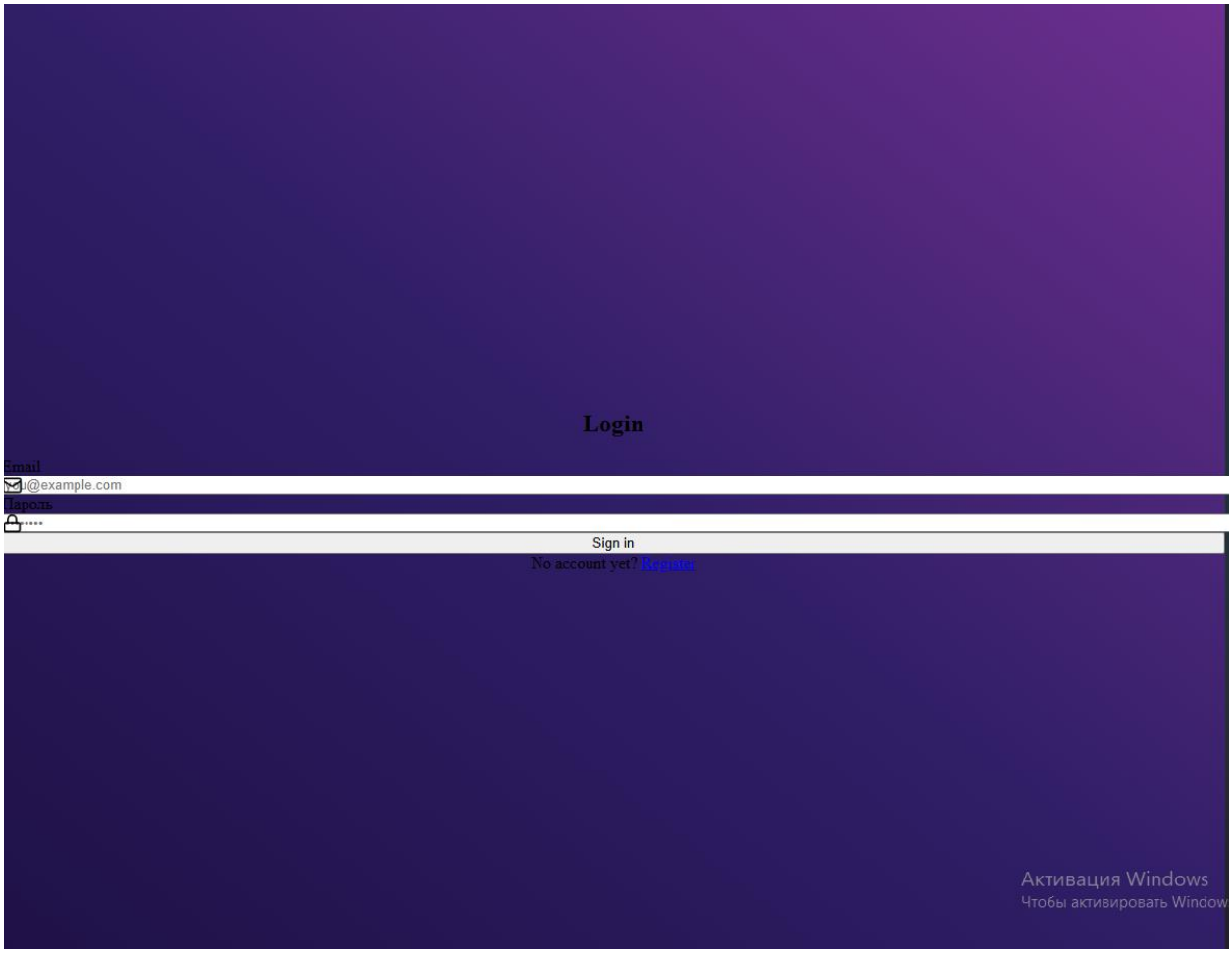


Рисунок 3.1 – Сторінка для логіну у систему

Детальніше про сторінку для авторизації (рис. 3.1)

– текстове поле із значком листа та підказкою. Слугує для вводу пошти;

– наступне текстове поле із значком замочку. Слугує для вводу паролю;

– кнопка увійти. Підтвердження введення даних, та запуск процесу авторизації;

– лінк-текст для реєстрації. Якщо користувач не має аккаунта, то завдяки надпису «Register» він має змогу перейти на відповідну сторінку.

Сторінка має влаштовану валідацію. При введені даних про неіснуючий запис, користувач отримає наступну помилку(рис. 3.2).

Рисунок 3.2 – Помилка при введені даних

Тепер розглянемо не менш важливий аспект – сторінка реєстрації. Являється невід’ємною частиною застосунку.

Рисунок 3.3 – Сторінка для реєстрації у систему

Детальніше про сторінку для авторизації (рис. 3.3)

– текстове поле для вводу пошти. У подальшому, буде використовуватися для ідентифікації користувача;

– наступні 2 поля – для вводу ім’я користувача та його фамілії;

– і останні два поля для вводу паролю, та його повторення;

– кнопка «Register». Підтвердження введених даних, та запуск процесу реєстрації користувача;

– лінка «Already have an account». Перехід до логін форми.

Залежачи від ролі авторизованого, після входу до нашого застосунку у нас відобразиться конкретний інтерфейс.

3.1.1 Роль «Спортсмен»

В застосунку присутні 2 головні ролі, а саме, тренер та спортсмен. Функціональні можливості спортсмену(рис. 3.4).

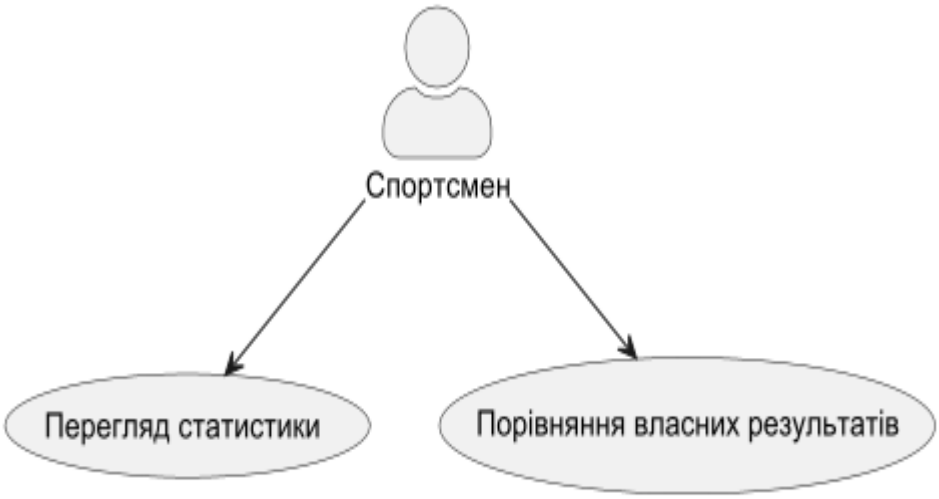


Рисунок 3.4 – Функціональні можливості користувача з роллю «Спортсмен»

Залогінившись як спортсмен, перша сторінка із якою стикається спортсмен, це домашня сторінка, на якій відображаються основні елементи щодо персональної статистики(рис. 3.5 та 3.6).

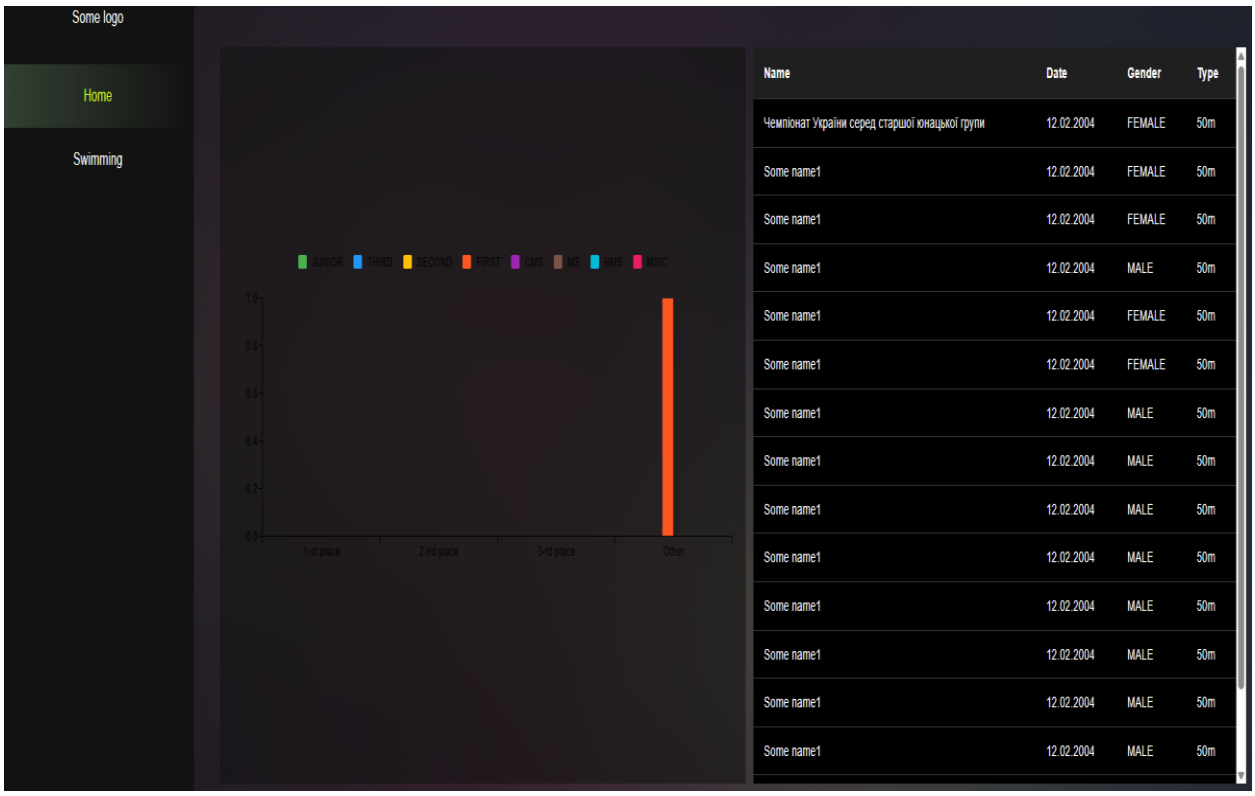


Рисунок 3.5 – Перша частина домашньої сторінки

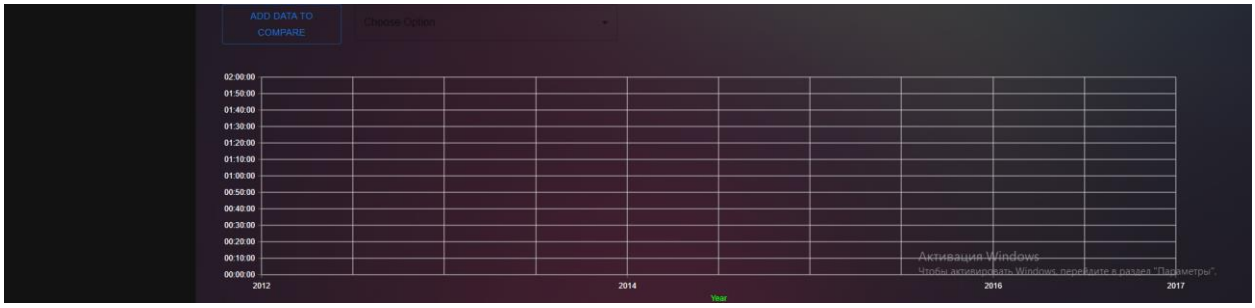


Рисунок 3.6 – Друга частина домашньої сторінки

Більш детальніше про домашню сторінку:

– зліва ми бачимо бокову панель, на якій відображаються усі доступні сторінки і жовтим фоном виділяється на якій ми зараз знаходимося;

– перший графік відображає кількість зайнятих місць під конкретним званням;

– наступна таблиця відображає останні змагання, у яких спортсмен приймав участь;

– далі (опираючись на рис 3.6) у нас є кнопка «Add data to compare», і меню вибору «Choose option»;

– порівняльний графік, на якому можна порівнювати свої найкращі результати по рокам, із обраним спортсменом або спортсменами.

При натисканні на кнопочку «Add data to compare» у користувача має відкритися модальне вікно (рис. 3.7).

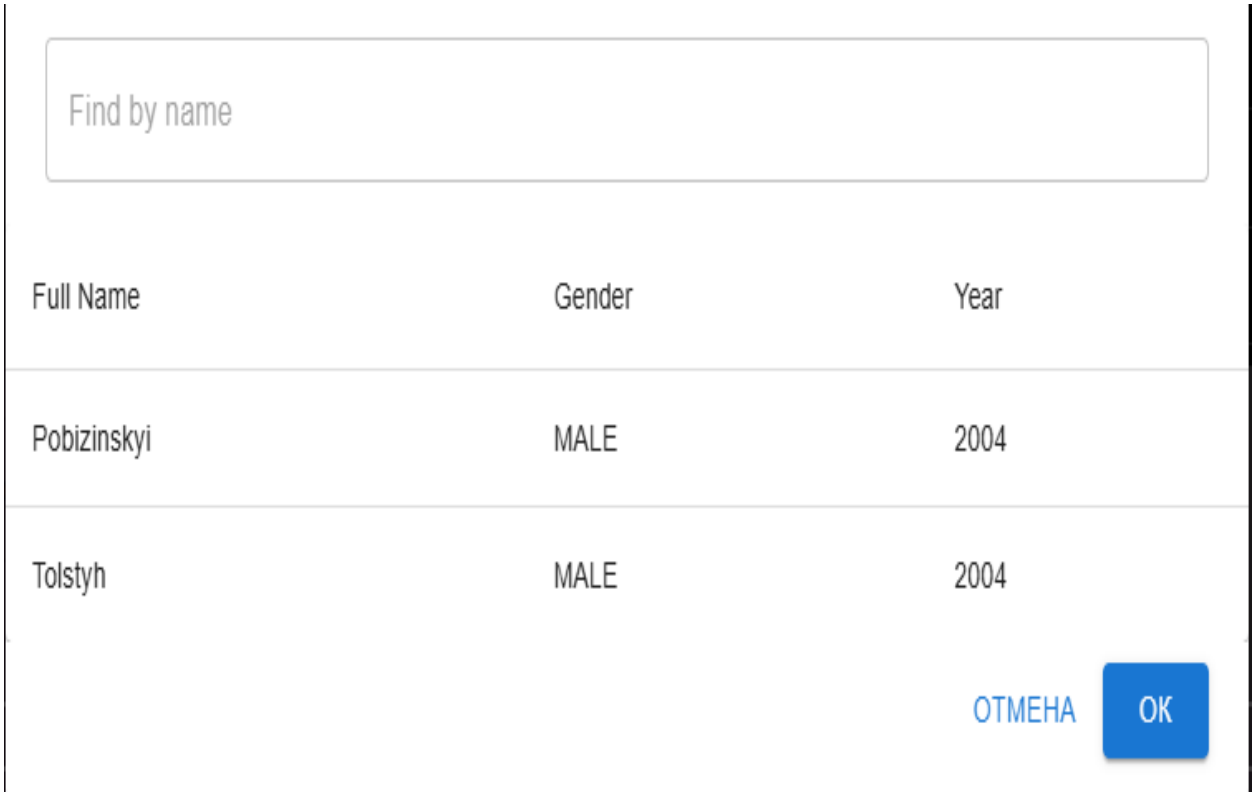


Рисунок 3.7 – Модальне вікно для вибору порівняльних даних

Із побаченого на рисунку 3.7 ми можемо помітити, що у нас є елементи інтерфейсу, а саме:

– текстове поле для пошуку спортсменів по імені;

– основні дані про спортсмена;

– кнопочки «Cancel» та «Ok». При натисканні першої, скинуться усі наші дії у діалоговому віконці. При натисканні другої, обрані спортсмени будуть додані на графік для порівняння(рис. 3.8).

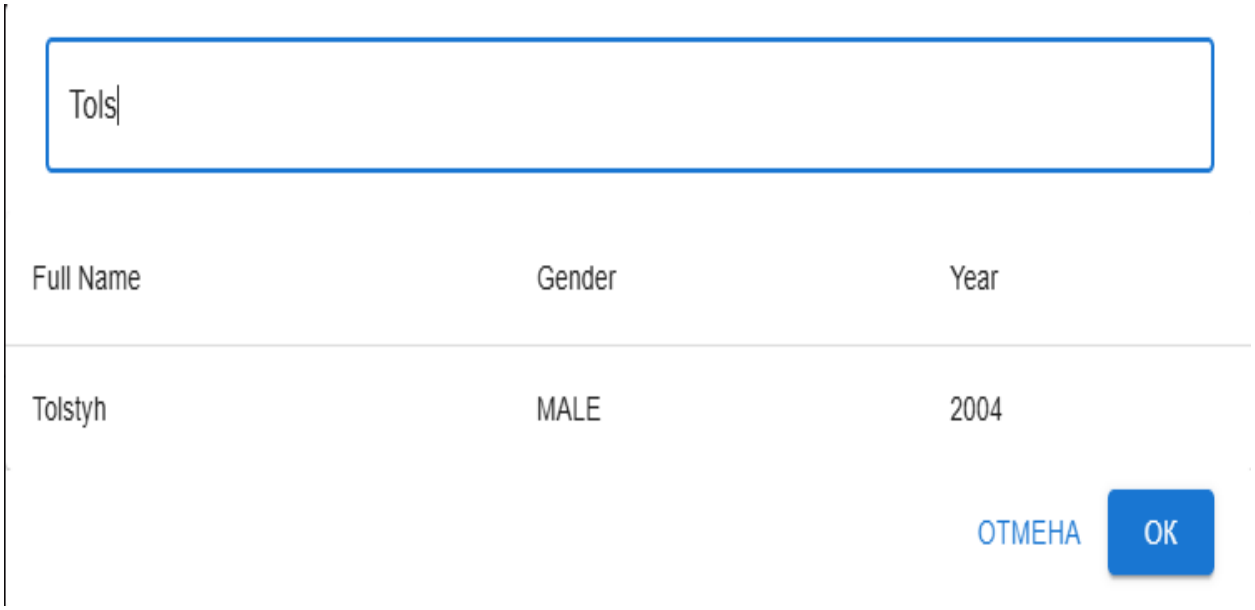


Рисунок 3.8 – Вдалий пошук спортсмена по імені та фамілії

При невдалому пошуку, користувач просто отримає пусту таблицю із користувачами. Варто зазначити, що пошук відпрацьовує тільки по імені та фамілії (рис. 3.9).

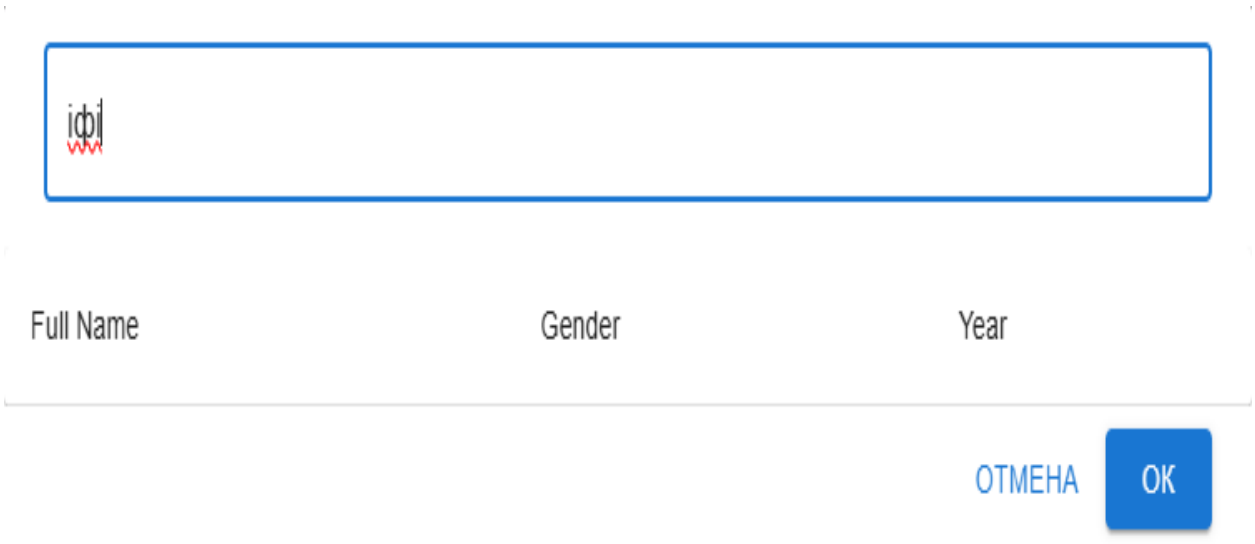


Рисунок 3.9 – Невдалий пошук спортсмена по імені та фамілії

Щоб застосувати потрібний набір даних до графіку, користувач має обрати декілька спортсменів, і натиснути кнопку «Ok» (рис. 3.10).

Рисунок 3.10 – Вибір спортсменів для порівняння

Після натискання на кнопку «Ok» модальне вікно зачиниться, і користувач повернеться до домашньої сторінки. Порівняльний графік буде виглядати наступним чином (рис. 3.11).

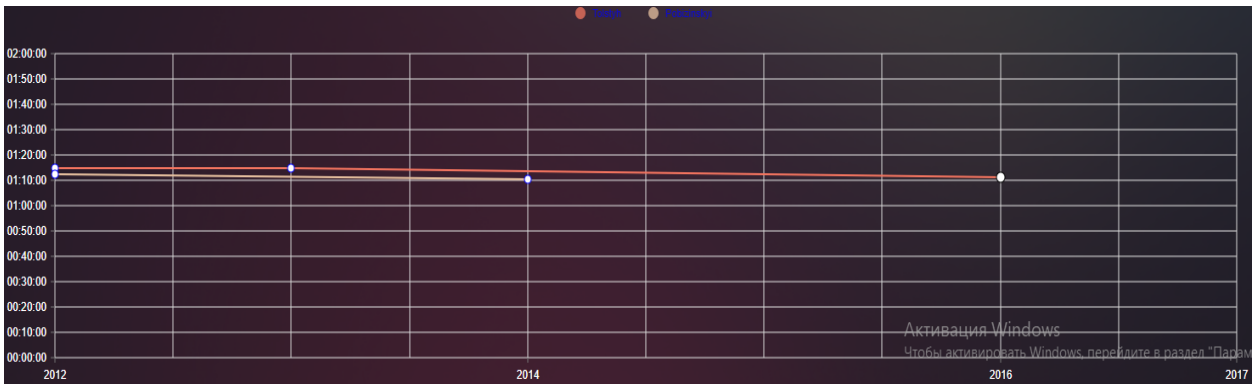


Рисунок 3.11 – Порівняльний графік

При наведені на ключові точки, користувач має змогу побачити свій найкращий результат зроблений у цьому році із конкретною категорією (рис. 3.12).

Рисунок 3.12 – Детальна інформація щодо виступу

Далі, потрібно звернути увагу на меню вибору, у якому користувач може обрати тип змагань, по яким буде виводитись список людей у нашому модальному віконці «Add data to compare». У випадаючому списку є основні категорії на вибір, а саме: SWIMMING_WITH_FINS, SCUBA_DIVING. Це два основних види змагань. У кожному із них є варіації у дистанції запливу, починаючи від 50 м і закінчуючи більш великими дистанціями (рис. 3.13).

Рисунок 3.13 – Випадаючий список для вибору типу і дистанції змагань

При натисканні на плашку із останніми змаганнями, спортсмен перейде до сторінки із більш розширеною інформацію про змагання (рис. 3.14).

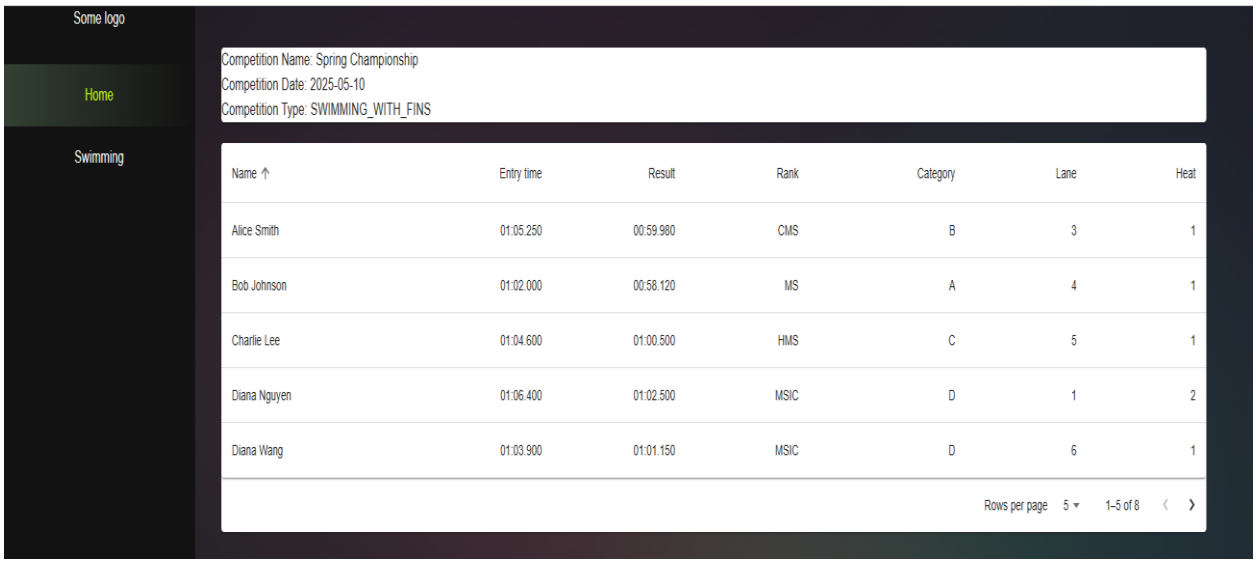


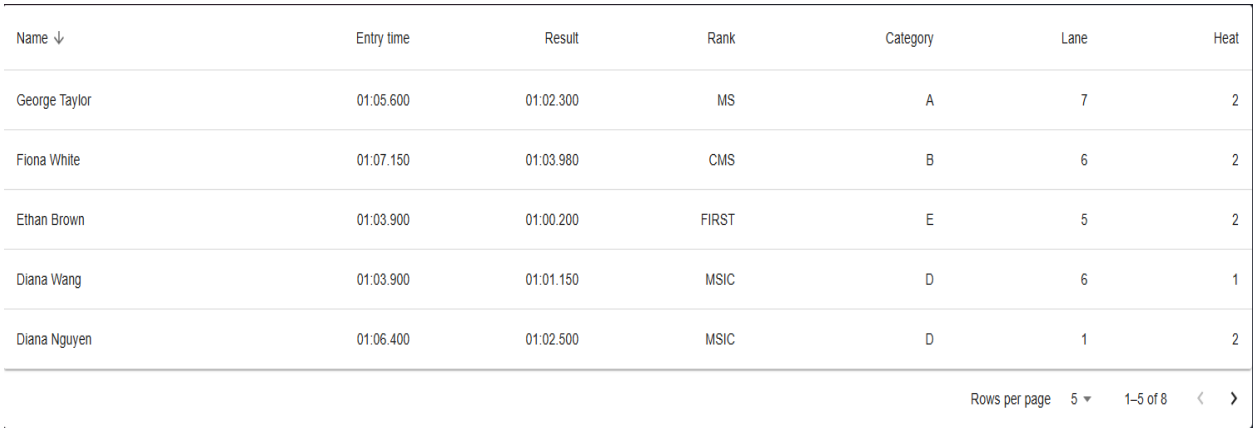
Рисунок 3.14 – Сторінка із інформацією про змагання

Трохи детальніше про цю сторінку:

– зверху відображається інформація про змагання, а саме: назва, дата проведення і тип;

– нижче приведена таблиця, у якій відображаються усі спортсмени, які приймали участь с заході;

– таблиця має пагінацію, можливість показувати потрібну кількість записів та вбудоване сортування по всім ознакам (рис. 3.15).

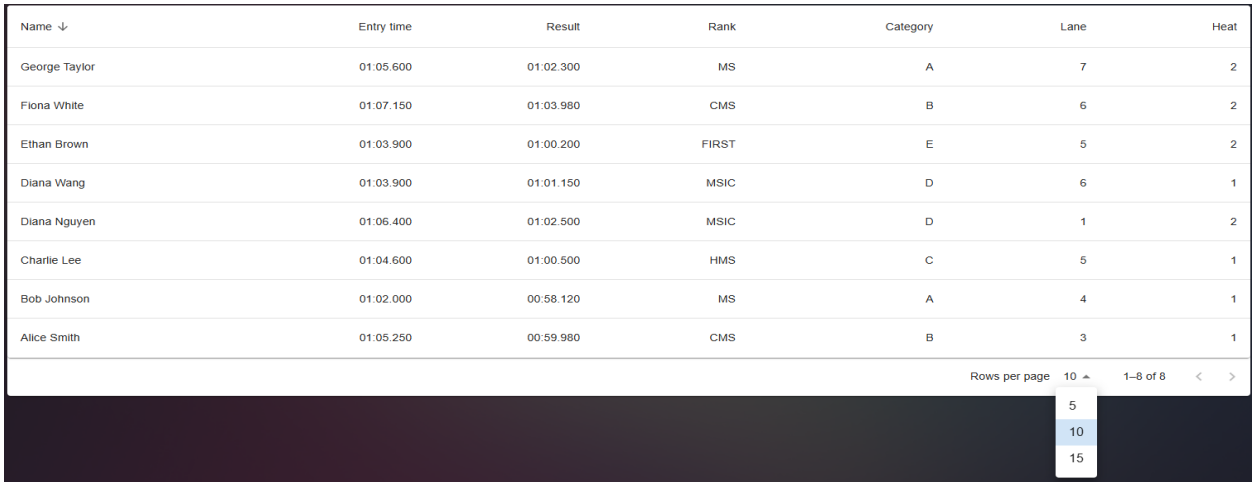


Name ↓	Entry time	Result	Rank	Category	Lane	Heat
George Taylor	01:05.600	01:02.300	MS	A	7	2
Fiona White	01:07.150	01:03.980	CMS	B	6	2
Ethan Brown	01:03.900	01:00.200	FIRST	E	5	2
Diana Wang	01:03.900	01:01.150	MSIC	D	6	1
Diana Nguyen	01:06.400	01:02.500	MSIC	D	1	2

Rows per page 5 1–5 of 8

Рисунок 3.15 – Приклад спадного сортування по імені та фамілії

І далі, якщо користувач захоче побачити набагато більше записів, на одній сторінці, тим самим спрощуючи собі життя, він може натиснути на випадаючий список, у котрому буде вибір кількості записів, які потрібно показувати на одній сторінці (рис. 3.16).



Name ↓	Entry time	Result	Rank	Category	Lane	Heat
George Taylor	01:05.600	01:02.300	MS	A	7	2
Fiona White	01:07.150	01:03.980	CMS	B	6	2
Ethan Brown	01:03.900	01:00.200	FIRST	E	5	2
Diana Wang	01:03.900	01:01.150	MSIC	D	6	1
Diana Nguyen	01:06.400	01:02.500	MSIC	D	1	2
Charlie Lee	01:04.600	01:00.500	HMS	C	5	1
Bob Johnson	01:02.000	00:58.120	MS	A	4	1
Alice Smith	01:05.250	00:59.980	CMS	B	3	1

Rows per page 10 1–8 of 8

5
10
15

Рисунок 3.16 – Вибір кількості відображення записів

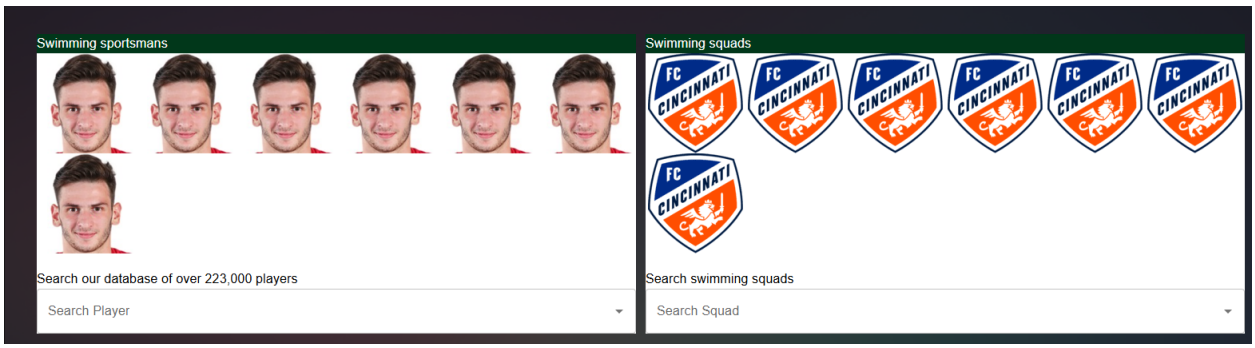


Рисунок 3.17 – Сторінка із інформацією про усіх існуючих спортсменів та команди

Розглянемо детальніше наступну сторінку під назвою «Swimming» (рис.3.17).

– «Swimming sportsmans» – таблиця усіх доступних спортсменів, які доступні у нашій базі даних. По кліку на спортсмена, або у випадаючиму меню нижче є поле вводу, у якому можно знайти і подивитись результати спортсмена, який тебе цікавить;

– «Swimming squads» – таблиця усіх існуючих команд у нашій бд. Аналогічно до попередньої таблиці, користувач може передевитися інформацію про будь-яку команду (рис. 3.18).

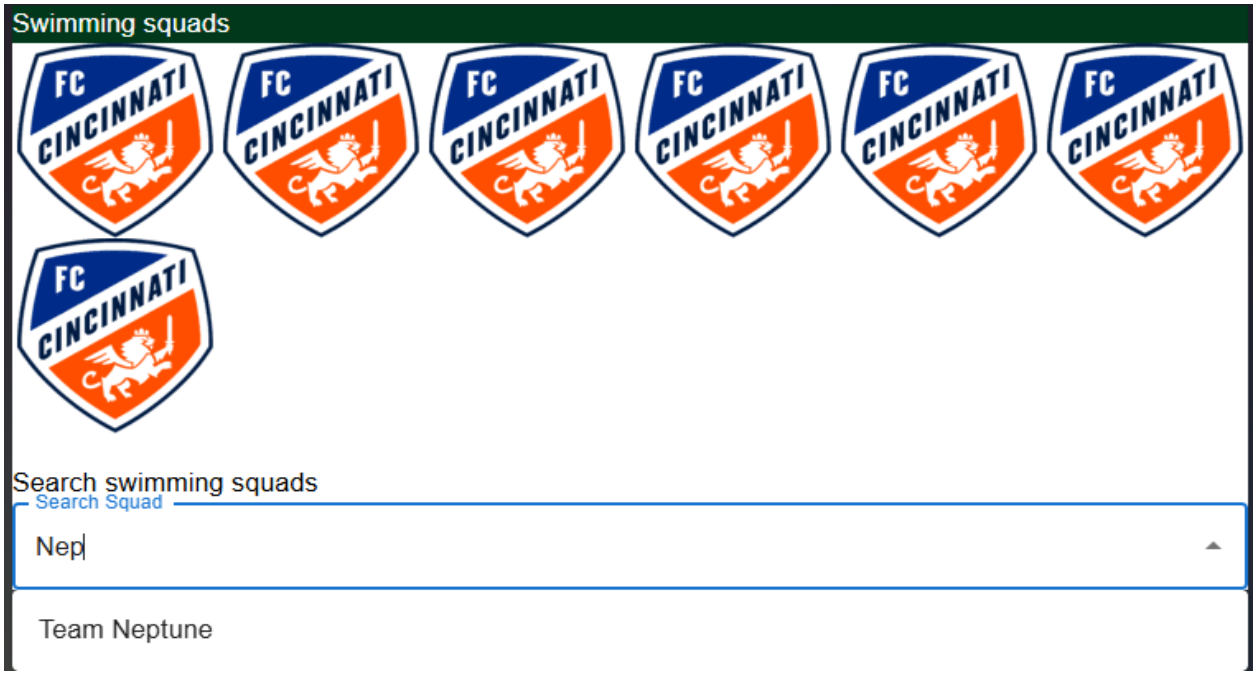


Рисунок 3.18 – Приклад пошуку команди

Варто зазначити, що пошук працює по імені спортсмена/команди. Так само, користувач може передевитись більш детальну інформацію про спортсмена, натиснувши на іконку із фото, або вибравши відповідну людину у випадаючому списку(рис. 3.19).

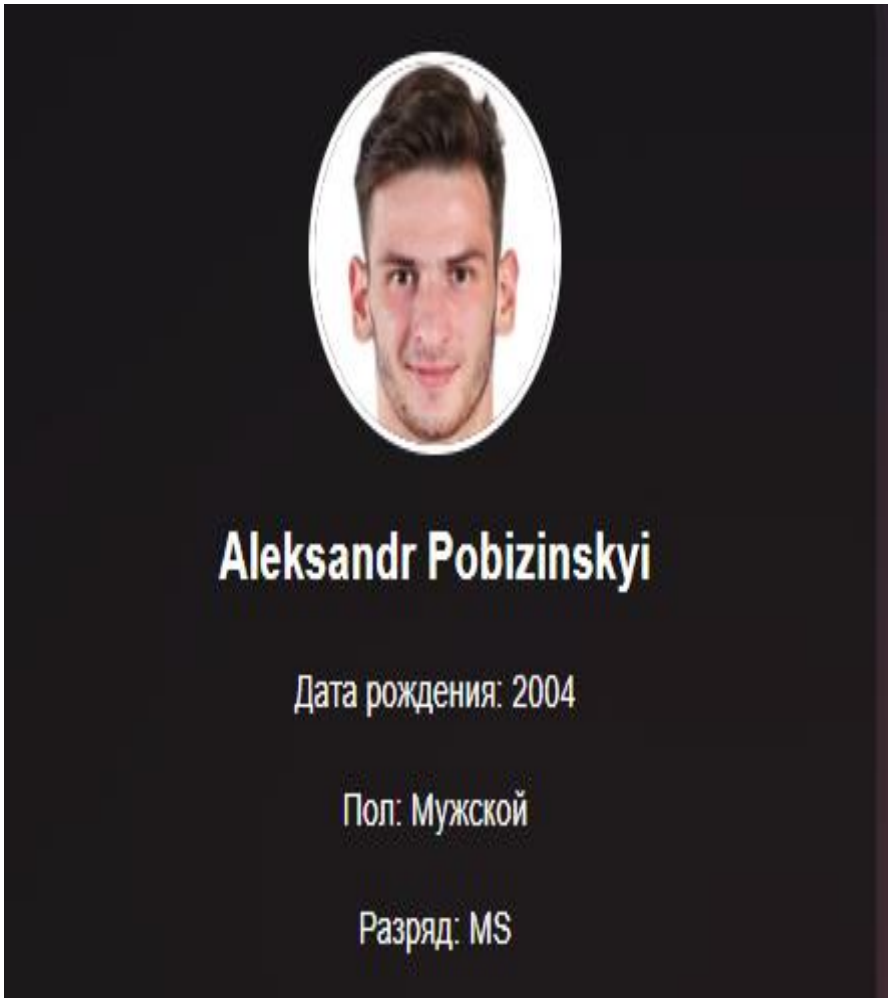


Рисунок 3.19 – Сторінка користувацького профілю

Аналогічними діями можна потрапити на сторінку з більш детальною інформацією про команду. Сама сторінка містить в собі багато корисних деталей пов’язаних із спортивним клубом. Завдяки гарному і дружелюбному інтерфейсу абсолютно кожен користувач без проблем може отримати доступ до бажаних даних. (рис. 3.20).

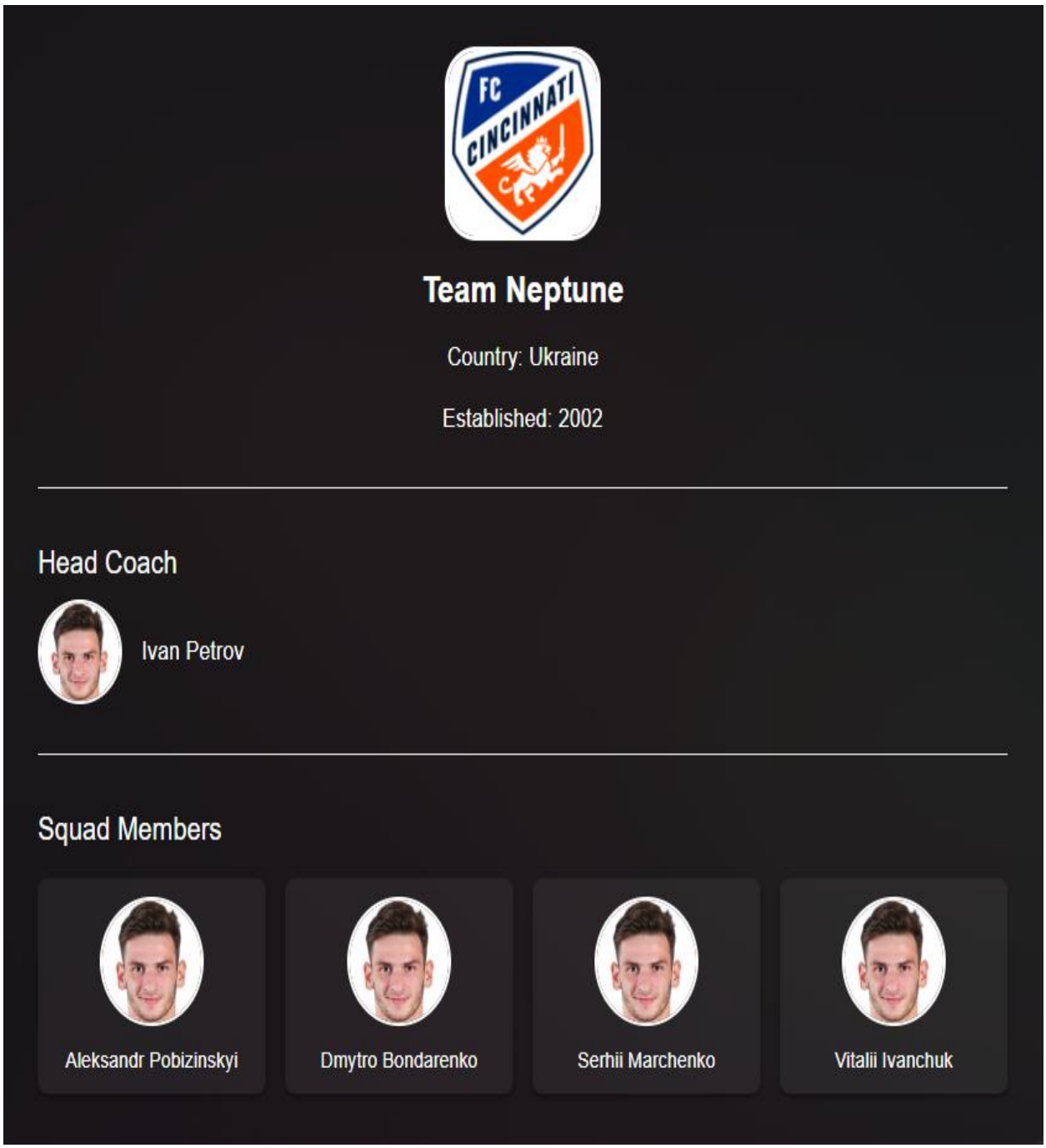


Рисунок 3.20 – Сторінка команди

Більш детальна інформація про сторінку команди:

– логотип та назва команди. Країна, та рік, коли команда була започаткована;

– тренер/тренера які тренують команду;

– спортсмени, які відносяться до команди.

3.1.2 Роль «Тренер»

Нагадаємо, що в застосунку присутні 2 головні ролі, а саме, тренер та спортсмен. Функціональні можливості тренера (рис. 3.21).

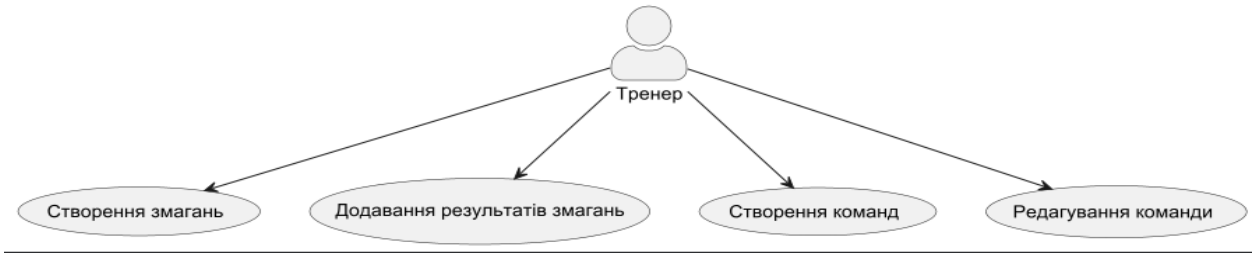


Рисунок 3.21 – Функціональні можливості тренера

Основні функції користувача з роллю «Тренер» включають в себе:

– створення змагань;

– додавання результатів змагань. Дуже зручний функціонал, який допомагає відслідковувати ріст спортсменів;

– створення команд;

– редагування існуючої команди. Додавання/виключення спортсменів;

Якщо на вебплатформу зайти як тренер, то можна одразу помітити візуальні відмінності сайд бару (рис. 3.22).

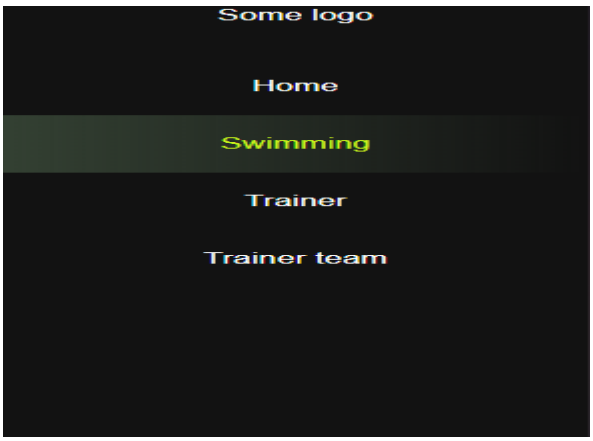


Рисунок 3.22 – Функціональні можливості тренера

Із малюнку вище, можна відокремити 2 нові сторінки. Trainer, та Trainer team. Почнемо із пункту Trainer (рис. 3.23).

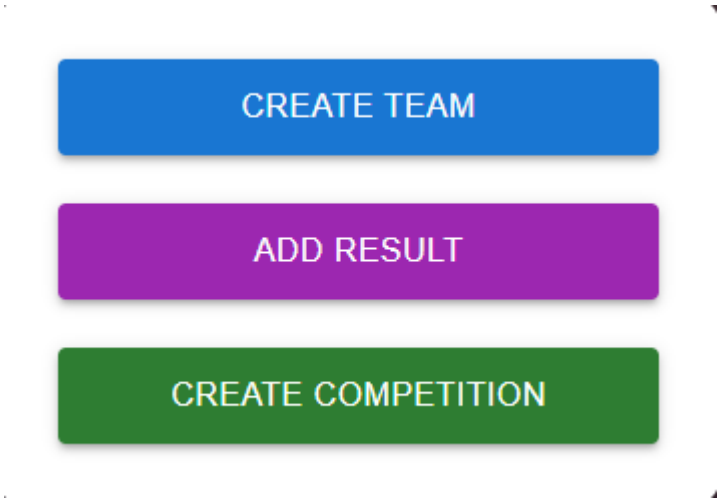


Рисунок 3.23 – Сторінка Trainer

Більш детальна інформація щодо функціональних можливостей:

– create team. Можливість додати нову команду, із певною кількістю спортсменів;

– add result. Додати результат зі змагань одному із обраних спортсменів;

– create competition. Створити змагання із певними параметрами.

Після натискання на кнопку create team відкриється модальне вікно наступного плану (рис. 3.24).

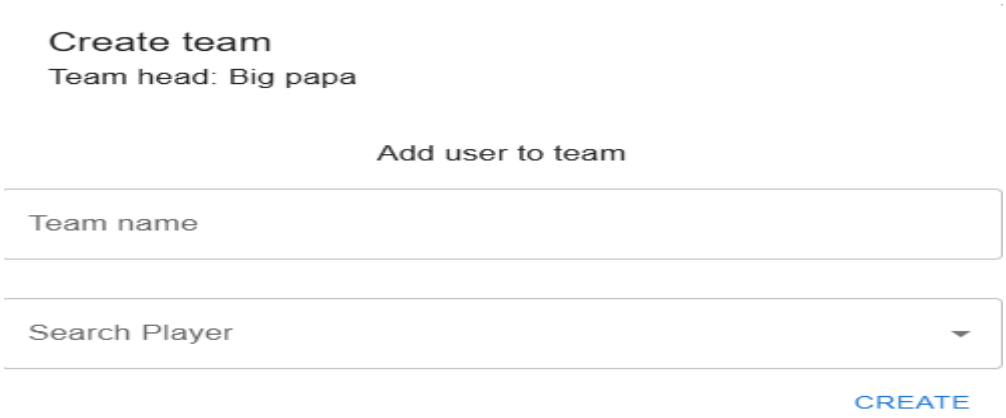


Рисунок 3.24 – Модальне вікно Create team

Із функціональних можливостей, тренер може додати назву команди та конкретних користувачів. Після додавання спортсмена, отримаємо наступні зміни у інтерфейсі (рис. 3.25).

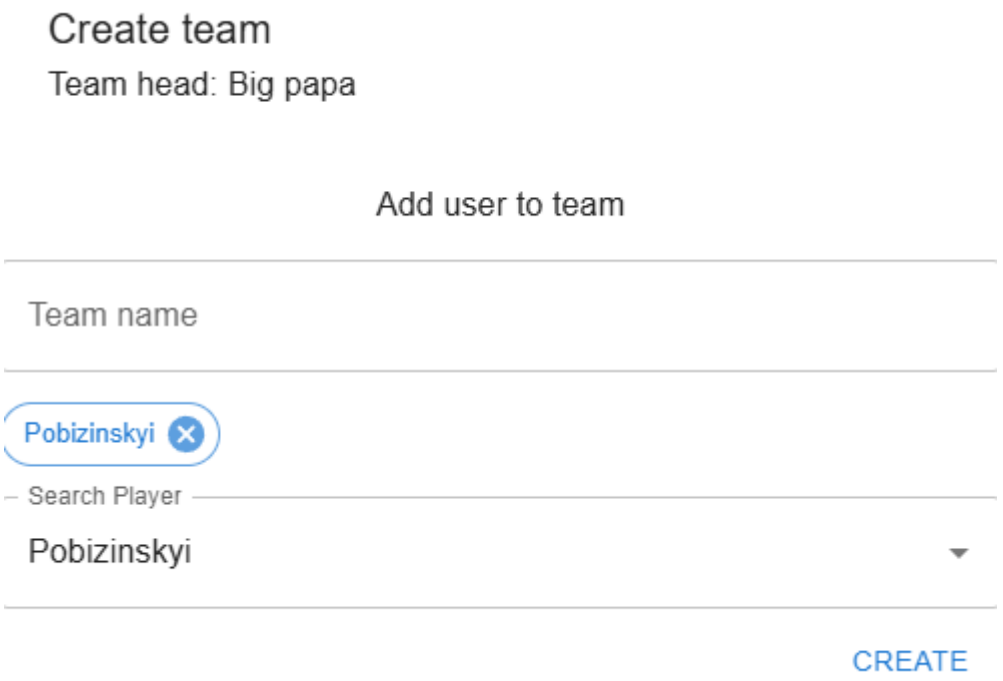


Рисунок 3.25 – Модальне вікно Create team після додавання спортсмену

Після вдалого створення команди, має з’явитися наступне модальне вікно, яке засвідчує, що команда була успішно створена (рис. 3.26).

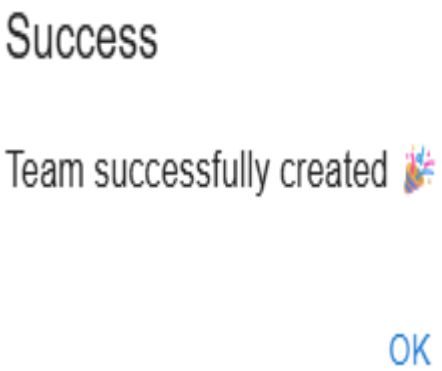


Рисунок 3.26 – Сповіщення про вдале створення команди

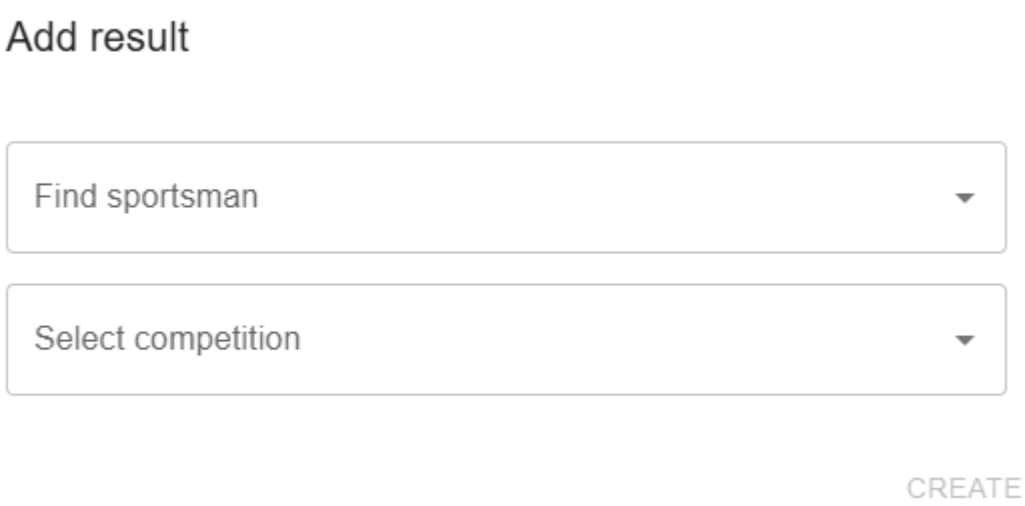


Рисунок 3.27 – Модальне вікно add result

Розглянемо більш детально наступне модальне вікно Add result (рис. 3.27). Функціональні можливості:

– знайти спортсмена, якому хочемо додати новий результат;
– вибір змагання, в якому виступав даний спортсмен.

Після вибору спортсмену, та введення назви змагань, інтерфейс зміниться наступним чином (рис. 3.28).

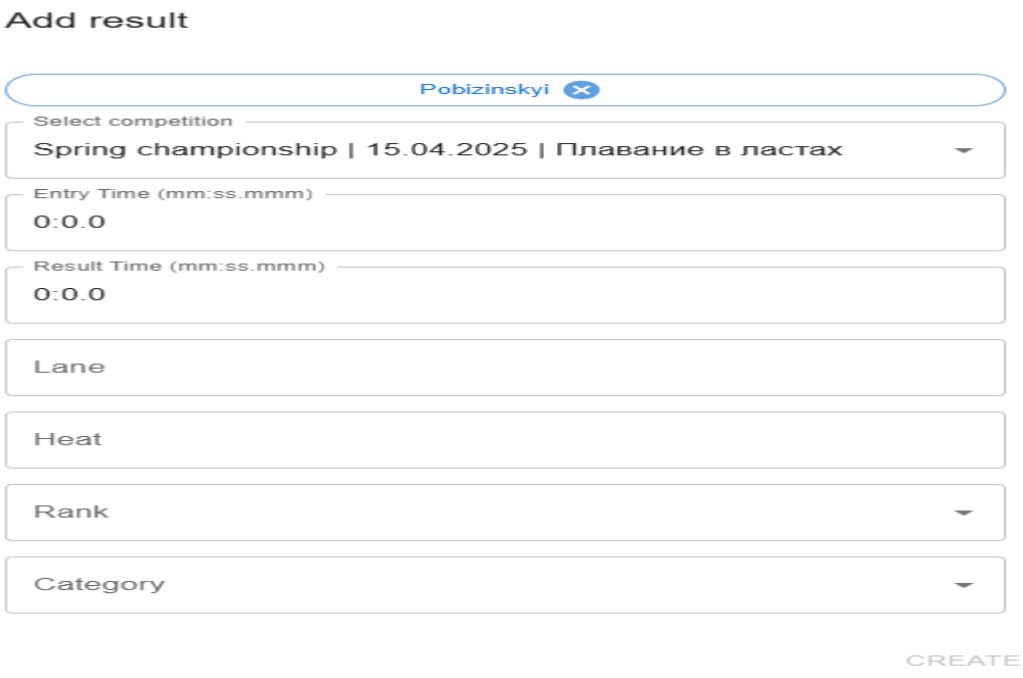


Рисунок 3.28 – Інтерфейс модального вікна add result після вибору змагань

Як ми можемо побачити, додались поля для заповнення спортивного результату:

– entry time. Відповідає за введений спортсменом час, за який він планує проплисти дистанцію;

– result time. Кінцевий результат, який продемонструвала людина яка змагалась;

– lane. Ряд, у якому пропливав спортсмен;

– heat. Змога, у яку був встановлений рекорд;

– rank. Спортивний розряд спортсмена(наприклад КМС, або МС);

– category. Категорія людини (A-E).

Після вдалого створення результат, отримаємо наступне модальне вікно (рис. 3.29).

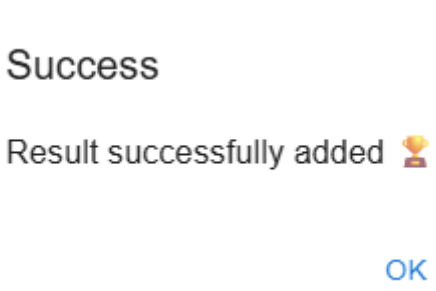


Рисунок 3.29 – Сповіщення про вдале додавання спортивного результату

Рисунок 3.30 – Модальне вікно create competition

Більш детальніше про форму створення змагань (рис. 3.30).

– назва змагань;
– дата змагань;
– тип змагань (підводне, або плавання в ластах).

Тренер повинен правильно заповнити усі дані. До тих пір кнопочка create залишатиметься неактивною. Так само після вдалого створення, у нас має з’явитися модальне вікно.

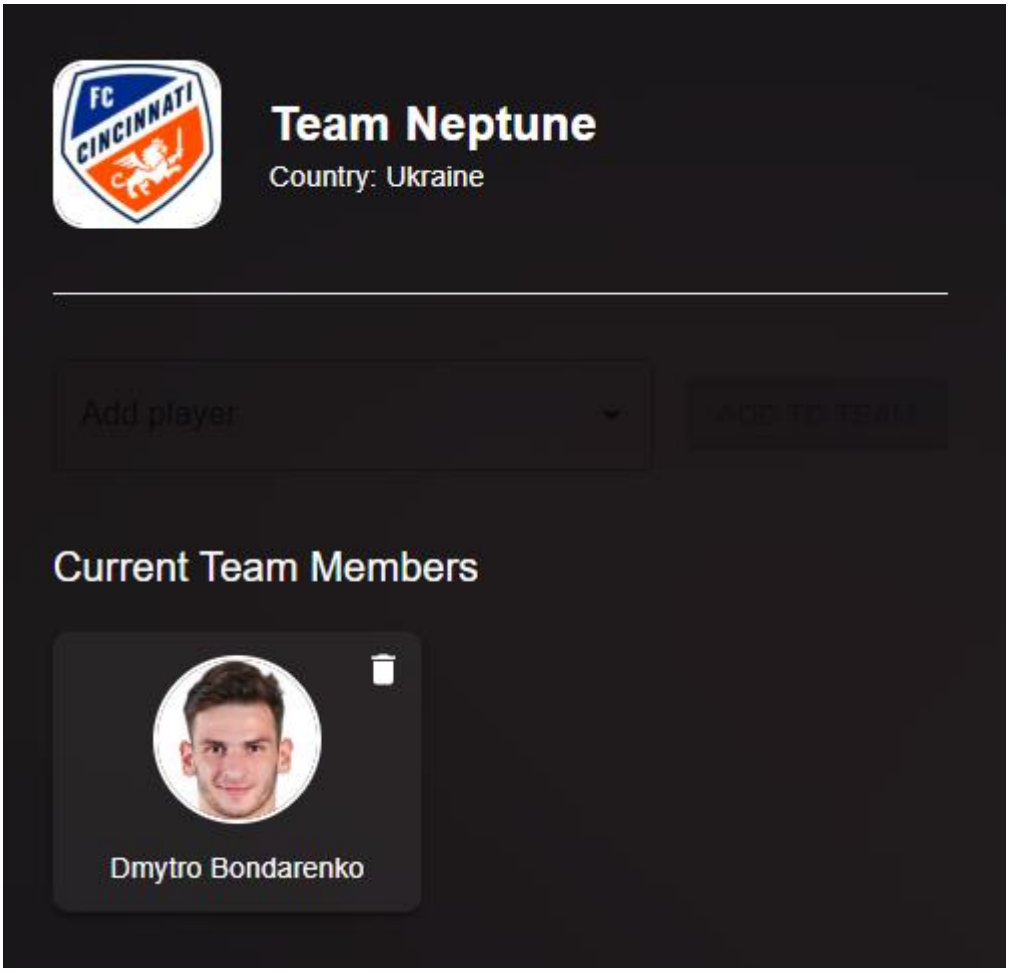


Рисунок 3.31 – Сторінка Trainer team

Детальніше про сторінку Trainer team (рис. 3.31).

– узагальнююча інформація про команду, назва та країна;
– можливість редагування людей, які існують у команді.

Ось приклад додавання користувача у команду (рис. 3.32).

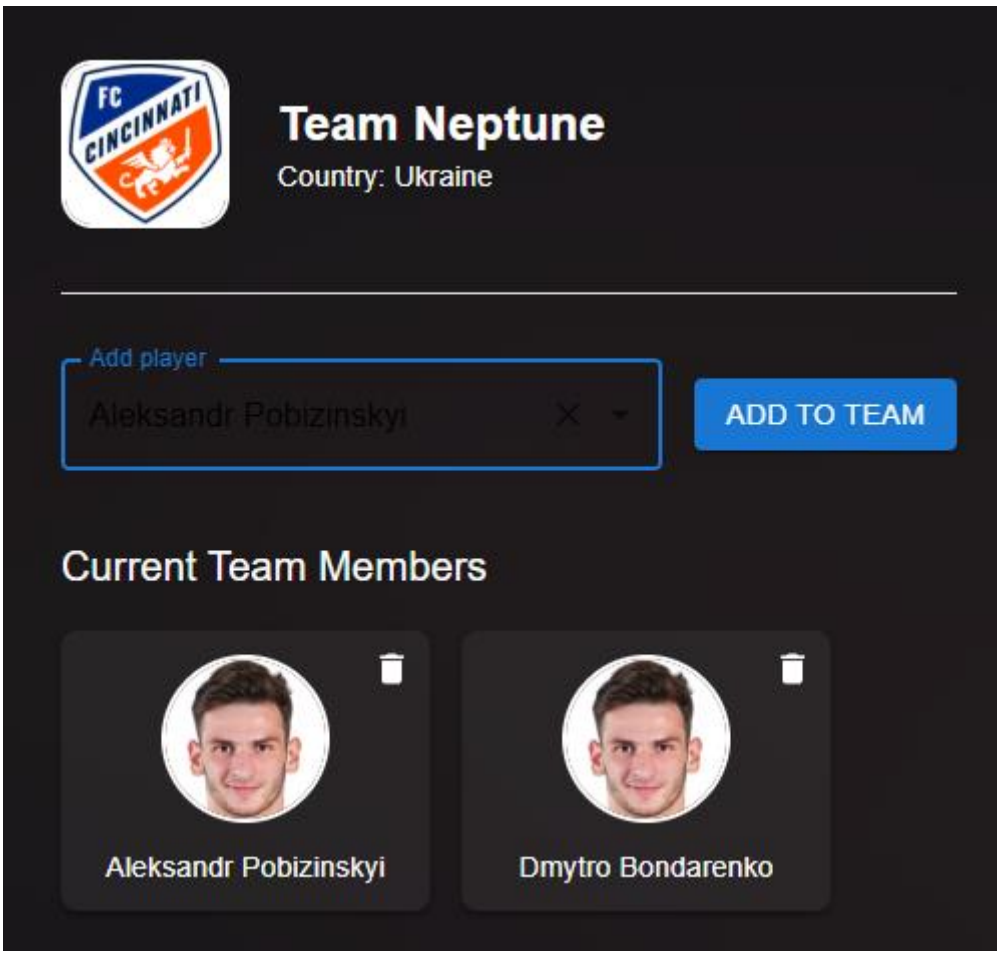


Рисунок 3.32 – Сторінка Trainer team. Вдале додавання спортсмена у команду

3.2 Серверна частина

Для зберігання даних використовується реляційна база даних Postgres, яка забезпечує гнучке та ефективне управління великими обсягами даних. Для створення серверної частини застосунку застосовуються Java та Spring Boot – це потужний фреймворк, який використовується для створення масштабованих серверних застосунків із використанням API. Для створення API використовується технологія REST API. Це бекенд, який є прошарком між вебзастосунком та базою даних. API надає стандартний інтерфейс для доступу до ресурсів. Відправляємо HTTP-запит з клієнта на сервер, у відповідь ми отримуємо не HTML-сторінку, а дані у форматі JSON [25].

Підключення до бази даних є критично важливим елементом більшості вебзастосунків, які використовують серверну логіку для зберігання та

обробки даних. Для підключення до бази даних використовується JPA(Java Persistence API), який під капотом використовує JDBC(Java Database Connectivity). Розглянемо програмний код підключення до бази даних, який працює під капотом програми. Його подано у лістингу 3.1 [26-28].

Лістинг 3.1 Реалізація підключення до бази даних Postgres за допомогою бібліотеки Jpa та Hibernate:

```
public static void main(String[] args) {
    String url = "jdbc:postgresql://localhost:9001/users ";
    String user = "admin";
    String password = "psw";
    try {
        Connection connection = DriverManager.getConnection(url, user, password);
        connection.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
```

У створеному застосунку підключення до бази даних відбувається наступним чином:

Крок 1. Для того, щоб підключитися до бази даних використовується DriverManager.getConnection(usr,user,password), де url– це шлях до нашої бази даних, і user та password для вдалої аутентифікації.

Крок 2. Після успішного підключення, серверний застосунок починає слухати вхідні HTTP запити на вказаному порті.

Крок 3. У випадку помилки підключення застосунок завершує свою роботу. Далі детально розглядається процес реалізації авторизації користувачів у системі вебзастосунку. Ця ключова функція є фундаментальною для забезпечення безпеки та індивідуалізації досвіду

користувача, дозволяючи користувачам управляти особистим обліковим записом та безпечно входити в систему для доступу до персоналізованих ресурсів і сервісів.

Спочатку розглядається програмний код авторизації користувача в системі. Його подано у лістингу 3.2 [29-30].

Лістинг 3.2 Реалізація функції реєстрації користувача в системі:

```
@Override
protected void doFilterInternal(HttpServletRequest request,
                                HttpServletResponse response,
                                FilterChain filterChain) throws ServletException,
IOException {
    String authHeader =request.getHeader(HttpHeaders.AUTHORIZATION);
    if (authHeader == null || !authHeader.startsWith("Bearer ")) {
       filterChain.doFilter(request, response);
       return;
    }
    String token = authHeader.substring(7);
    try {
       SecretKey key =
Keys.hmacShaKeyFor(jwtSecret.getBytes(StandardCharsets.UTF_8));
       Claims claims = Jwts.parserBuilder()
            .setSigningKey(key)
            .build()
            .parseClaimsJws(token)
            .getBody();

       String userId = claims.getSubject();
       User user = new User(userId, "", Collections.emptyList());
       UsernamePasswordAuthenticationToken authentication =
```

```
                    new UsernamePasswordAuthenticationToken(user, null,
user.getAuthorities());
                authentication.setDetails(new
WebAuthenticationDetailsSource().buildDetails(request));

                SecurityContextHolder.getContext().setAuthentication(authentication);
            } catch (Exception e) {
                response.sendError(HttpServletResponse.SC_UNAUTHORIZED,
"Invalid JWT Token");
                return;
            }
            filterChain.doFilter(request, response);
        }
```

Дана функція авторизації здійснюється за наступною схемою:

Крок 1. Виконується перевірка чи прислала клієнтська частина токен. Перевірка відбувається за допомогою перевірки Header із ключом Auth, який в свою чергу має починатись на слово Bearer.

Крок 2. Далі витягається авторизаційний токен, для подальших дій над ним.

Крок 3. Наступним кроком використовується бібліотека io.jsonwebtoken.security для безпечного хешування секретного ключа, використовуючи який можна дістати uid юзера.

Крок 4. Після цього Spring Security сетапить аутентифікацію у своє сховище, і завдяки цьому подальший користувач матиме доступ до ендпоінтів.

Крок 5. Якщо усі попередні кроки пройдені успішно, клієнту надсилається відповідь із статусом 201.

Після успішної реєстрації та входу в профіль, користувачу відкривається уся функціональність застосунку для введення персональної

статистики. Розглянемо найголовніші API-endpoint серверної частини вебзастосунку. Коротко про кожен з них наведено у таблиці 3.1.

Таблиця 3.1 – Опис API-endpoint

Назва API-endpoint	URL	Опис
Реєстрація	/api/users/register	Призначена для створення нових облікових записів користувачів у системі
Авторизація	/api/users/login	Призначена для авторизації користувачів у системі
Створення змагання	/api/competition/create	Створення нового змагання
Створення результату	/api/competition-result/create	Додавання спортивного результату у конкретному змаганні
Отримання персональних результатів	/api/user/personal-stats/{id}	Персональний результат у виді кількості перемог у змаганнях під конкретним званням
Створення команди	/api/team/create	Створення нової команди
Знаходження результатів із змагань	/api/competition-result/find-last	Отримання і результатів із останніх змагань, у яких спортсмен приймав участь

ВИСНОВКИ

У рамках кваліфікаційної роботи був розроблений застосунок для введення персональної статистики із змагань по підвоному синхронному плаванню.

Основною метою роботи було створення зручного та ефективного інструменту для користувачів, який дозволяє автоматизувати роботу тренерів, а саме процесу заповнення потрібних документів, порівняння спортивних статистик, тощо.

Під час роботи було використано надійні й ефективні технології та бібліотеки. Для реалізації клієнтської частини програмного забезпечення використовувалися TypeScript та React. Для розробки серверної частини використовувалися Java Spring Boot.

Базу даних системи було створено відповідно до всіх вимог, використовуючи Postgres.

Робочу версію застосунку було успішно розгорнуто і він працює відповідно до всіх вимог та є стійким до помилок у користуванні.

Результатом роботи є функціональний інструмент, який використовується для введення персональних статистик для спортсменів та команд.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Bodyanskiy Y., Vynokurova O., Kobylin, I., Kobylin O. (2016) Adaptive fuzzy clustering of short time series with unevenly distributed observations in Data Stream Mining tasks, Information Technology and Management Science. pp. 23-28.

2. Bodyanskiy, Y., Vynokurova, O., Szymański, Z., Kobylin, I., & Kobylin, O. (2016, August). Adaptive robust models for identification of nonstationary systems in data stream mining tasks. In 2016 IEEE First International Conference on Data Stream Mining & Processing (DSMP) (pp. 263-268). IEEE

3. Радіоелектроніка та молодь у XXI столітті: 28 міжнародний молодіжний форум. (Харків — 2025 р.) Харків 2025. С. Твоя с молодежки

4. Кобилін, І. О., & Ніколайчук, А. І. (2024). MONITORING AND DIAGNOSING FAULTS IN ONLINE MODE USING TIME SERIES DATA. Системи обробки інформації, (3 (178)), 27-32.

5. Кобилін, І. О., & Харченко, А. І. (2024). CLASSIFICATION TECHNIQUES FOR COMPUTER VISION. Системи обробки інформації, (3 (178)), 33-41.

6. Kharchenko, A. I., & Kobylin, I. O. Perfomance Analysis of Support Vector Machine for Vehicle Classification. Ministry of Education and Science of Ukraine vn Karazin Kharkiv National University, 101.

7. Бодянський, Є., Винокурова, О., Кобилін, І., & Мулеса, П. (2017). Адаптивна матрична нейро-фаззі самоорганізовна мережа для кластеризації багатовимірних потоків даних. Вісник Національного університету Львівська політехніка. Комп'ютерні науки та інформаційні технології, (864), 314-319.

8. Бодянський, Є. В., Винокурова, О. А., Ізонін, І. В., Кобилін, І. О., & Мулеса, П. П. (2017). Кластеризація багатовимірних часових рядів на основі адаптивної матричної нейро-фаззі самоорганізовної мережі. Intellectual Systems For Decision Making and Problems of Computational Intelligence, 247.

9. Rabotiahov, A., Kobylin, O., Dudar, Z., & Lyashenko, V. (2018, February). Bionic image segmentation of cytology samples method. In *2018 14th International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET)* (pp. 665-670). IEEE.

10. Kobylin, O., & Lyashenko, V. (2016). Contrast Modification as a Tool to Study the Structure of Blood Components.

11. Kinoshenko, D., Kobylin, O., Mashtalir, S., & Stolbovyi, M. (2019, March). Metric video retrieval speedup by irrelevant data elimination. In Eleventh International Conference on Machine Vision (ICMV 2018) (Vol. 11041, pp. 176-183). SPIE.

12. Gorokhovatskyi, V. A., Rusakova, N., & Tvoroshenko, I. S. (2020). The application of image analysis methods and predicate logic in applied problems of magnetic monitoring. Telecommunications and Radio Engineering, 79(20).

13. Daradkeh, Y. I., & Tvoroshenko, I. (2020). Technologies for making reliable decisions on a variety of effective factors using fuzzy logic. International Journal of Advanced Computer Science and Applications, 11(5).

14. Gorokhovatskyi, V. O., Tvoroshenko, I. S., & Vlasenko, N. V. (2020). Using fuzzy clustering in structural methods of image classification. Telecommunications and Radio Engineering, 79(9).

15. Yakovleva, O., & Nikolaieva, K. (2020). Research of descriptor based image normalization and comparative analysis of SURF, SIFT, BRISK, ORB, KAZE, AKAZE descriptors. Advanced Information Systems, 4(4), 89-101.

16. Daradkeh, Y. I., & Tvoroshenko, I. (2020). Technologies for making reliable decisions on a variety of effective factors using fuzzy logic. International Journal of Advanced Computer Science and Applications, 11(5).

17. Pomazan, V., Tvoroshenko, I., & Gorokhovatskyi, V. (2023). Development of an application for recognizing emotions using convolutional neural networks. *International Journal of Academic Information Systems Research*, 7(7), (pp. 25-36).

18. Lyashenko V., Kobylin O., Selevko O. (2020) Wavelet Analysis and Contrast Modification in the Study of Cell Structures Images. International Journal of Advanced Trends in Computer Science and Engineering. 9(4). – 4701-4706.

19. Oleg, K., Sergii, M., & Mykhailo, S. (2017, October). Video Clustering via Multidimensional Time-Series Analysis. In *Proceedings of the 9th International Conference on Information Management and Engineering* (pp. 60-63). ACM.

20. Кобилін, О. А., & Творошенко, І. С. (2021). Методи цифрової обробки зображень.

21. Yakovleva, O., Kovtunenko, A., Liubchenko, V., Honcharenko, V., & Kobylin, O. (2023). Face Detection for Video Surveillance-based Security System. In COLINS (3) (pp. 69-86).

22. Yakovleva, O., Slyusar, V., Kushnir, O., & Sabovchyk, A. (2021). New trends in scientific and technological revolution (STR) and transformation of science and education systems in the paradigm of sustainable development. In E3S Web of Conferences (Vol. 277, p. 06006). EDP Sciences.

23. Bodyanskiy, Y., Vynokurova, O., Kobylin, I., & Kobylin, O. (2016). Adaptive fuzzy clustering of short time series with unevenly distributed observations in Data Stream Mining tasks. *Information Technology and Management Science*, *19*(1), 23-28.

24. Gorokhovatskyi, V., Tvoroshenko, I., & Olena, Y. (2024). Transforming image descriptions as a set of descriptors to construct classification features. *Indonesian Journal of Electrical Engineering and Computer Scienc*, 33 (1), (pp. 113-125)

25. Kuzminska, O., Mazorchuk, M., Morze, N., & Kobylin, O. (2019, June). Digital learning environment of ukrainian universities: The main components to influence the competence of students and teachers. In International Conference on Information and Communication Technologies in Education, Research, and Industrial Applications (pp. 210-230). Springer, Cham.

26. Кобилін, О. А., & Путятіна, О. Є. (2024). Знешумлення зображень, зіпсованих дробовим шумом, у реальному часі. Системи обробки інформації, (1 (176), 46-51.

27. Gorokhovatskyi , V., Chmutov , Y., Tvoroshenko , I., & Kobylin , O. (2025). Reducing computational costs by compressing the structural description in image classification methods. Advanced Information Systems, 9(1), 5–12. https://doi.org/10.20998/2522-9052.2025.1.01

28. Кобилін, О., Вечірська, І., Афанасьєв, А. (2024). Аналіз існуючих моделей глибинного навчання в задачах обробки природної мови. Information Technology: Computer Science, Software Engineering and Cyber Security, 3, 63–76, https://doi.org/10.32782/IT/2024-3-7

29. Кобилін, О., Вечірська, І., Кравченко, О. (2024). Порівняння нейронних мереж типу RNN та LSTM. Information Technology: Computer Science, Software Engineering and Cyber Security, 3, 97–107, https://doi.org/10.32782/IT/2024-3-10

30. Vechirska, I., Kobylin, O., Prokopiev , S., Vechirska, A., & Kucherenko, M. (2022). Building a logical network for solving the problem of car rental by means algebra of finite predicates. Computer Systems and Information Technologies, (2), 78–87. https://doi.org/10.31891/csit-2022-2-9

31. Kobylin, O.; Putiatina, O. (2025). Some aspects of real-time image denoising influenced by shot noise and compound Poisson noise. CEUR Workshop Proceedings ,volume = 3943 , 109-117