Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет *Комп'ютерних наук*
(повна назва)

Кафедра *Системотехніки*
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Розробка системи аналізу футбольних матчів з побудовою метрик ефективності та рекомендаційною підтримкою тренерських рішень
(тема)

Виконав:
здобувач *2* року навчання,

групи *ІТПм-24-1*

Олексій Латишев
(власне ім'я, прізвище)

Спеціальність *122 Інформаційні технології проектування*
(код і повна назва спеціальності)

Тип програми *освітньо-професійна*
(освітньо-професійна або освітньо-наукова)

Освітня програма *Комп'ютерні науки*
(повна назва освітньої програми)

Керівник *проф. Світлана Гавриленко*
(посада, прізвище, ініціали)

Допускається до захисту
Завідувач кафедри _____ *Гребеннік І.В.*
(підпис) (прізвище, ініціали)

2025 р.

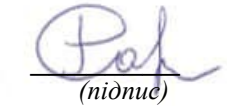
Я, як студент ХНУРЕ розумію і підтримую політику закладу із академічної доброчесності. Я не надавав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

(підпис) *«10» грудня 2025 р.* *(дата)* *Латишев О.О.* *(прізвище, ініціали)*

Кваліфікаційна робота не містить відомостей заборонених до відкритого опублікування.

Кваліфікаційна робота виконана у відповідності до стандартів, що діють в Україні.

Попередній захист проведено «10» грудня 2025 р.

Керівник кваліфікаційної роботи  *проф. Гавриленко С.Ю.*

(підпис) *(прізвище, ініціали)*

Харківський національний університет радіоелектроніки

Факультет *Комп'ютерних наук*
Кафедра *Системотехніки*
Рівень вищої освіти *другий (магістерський)*
Спеціальність *122 Інформаційні технології проектування*
(код і повна назва)
Тип програми *освітньо-професійна*
(освітньо-професійна або освітньо-наукова)
Освітня програма *Комп'ютерні науки*
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)
«____» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові *Латишеву Олексію Олександровичу*
(прізвище, ім'я, по батькові)

1. Тема роботи *Розробка системи аналізу футбольних матчів з побудовою метрик ефективності та рекомендаційною підтримкою тренерських рішень*

затверджена наказом університету від *24* *листопада* 2025 р. № *1058 Ст*

2. Термін подання студентом роботи до екзаменаційної комісії *12* *грудня* 2025 р.

3. Вихідні дані до роботи: *дослідити методи комп'ютерного зору для детекції та трекінгу об'єктів на відеопотоці, розробити систему для автоматизованого збору статистики матчу та розрахунку індивідуальних метрик гравців, реалізувати компонент рекомендаційної підтримки для тренерського штабу.*

4. Перелік питань, що потрібно опрацювати в роботі *4.1 Вступ. 4.2 Аналіз предметної області. 4.2.1 Опис сучасного стану розвитку інформаційних систем аналізу футбольних матчів. 4.2.2 Огляд існуючих інформаційних систем аналізу футбольних матчів. 4.3 Постановка завдання на дослідження та розробку. 4.3.1 Формування задачі на дослідження. 4.3.2 Формування задачі на розробку. 4.4 Дослідження існуючих методів та технологій аналізу футбольних матчів. 4.4.1 Огляд існуючих методів та стратегій аналізу ефективності гравця на полі. 4.4.2 Методи комп'ютерного зору для аналізу футбольних матчів. 4.4.3 Методи розрахунку аналітичних метрик на основі даних розпізнавання. 4.4.4 Використання моделей штучного інтелекту для формування рекомендацій. 4.5 Розробка інформаційної системи для вирішення задачі. 4.5.1 Обґрунтування вибору технологій та архітектурних рішень. 4.5.2 Проєктування структури баз даних. 4.5.3 Компонент обробки відеозапису матчу. 4.5.4 Компонент розрахунку метрик. 4.5.5 Компонент метаданих. 4.5.6 Компонент інтерфейсу взаємодії з системою.*

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) *5.1 Футбольне поле з ключовими точками у системі координат. 5.2 Штрафна та воротарська площі з розрахованими координатами для ключових точок. 5.3 Результат перетворення тестового кадру відеоряду за допомогою алгоритму Гомографії. 5.4 Архітектура інформаційної системи. 5.5 ERD-модель розробленої БД системи. 5.6 Діаграма послідовності дій компонента обробки відеозапису. 5.7 Результат навчання моделі YOLOv8 на датасеті з розміченими гравцями, м'ячом та суддями. 5.8 Результат виконання розпізнання гравців, суддів та м'яча. 5.9 Процес виділення кольору форми гравців. 5.10 Процес визначення кольору форми методом K-Means. 5.11 Процес глобальної кластеризації кольорів форм методом K-Means. 5.12 Результат розпізнання кольору форми гравців. 5.13 Кадр з відеоряду гри EA SPORTS FC™ 26*

на якому видно гравців і їх позиції. 5.14 Представлення результату роботи алгоритму Гомографії. 5.15 Діаграми послідовності дій компонента розрахунку метрик. 5.16 Візуалізація схеми GraphQL. 5.17 Головна сторінка веб-додатку. 5.18 Частина сторінки про матч з інформацією про команди. 5.19 Агрегована статистики матчу на базі метрик гравців за цей матч. 5.20 Статистика гравців за матч. 5.21 Відео плеєр з записом матча. 5.22 Заповнена сторінка створення матча. 5.23 Сторінка завантаження відео матча. 5.24 Sequence діаграма процесу генерації рекомендацій за допомогою LLM. 5.25 Контекст, що передається до LLM на сторінці матча. 5.26 Процес комунікації користувача з LLM за допомогою чата. 5.27 Рекомендації, що були надані користувачеві LLM після аналізу результатів матча.

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
	проф. Гавриленко С. Ю.		10.12.2025

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / термін виконання етапів роботи	Примітка
1	Отримання завдання кваліфікаційної роботи	24.11.2025	Виконано
2	Аналіз завдання та предметної області	10.10.2025 – 19.10.2025	Виконано
3	Постановка завдання на дослідження та розробку	20.10.2025 – 24.10.2025	Виконано
4	Дослідження існуючих методів та технологій аналізу футбольних матчів	25.10.2022 – 09.11.2025	Виконано
5	Розробка інформаційної системи для вирішення задачі	10.11.2025 – 28.11.2025	Виконано
6	Дослідження результатів роботи розробленої інформаційної системи	29.12.2025 – 02.12.2025	Виконано
7	Формування пояснювальної записки	03.12.2025 – 07.12.2025	Виконано
8	Представлення на рецензування	10.12.2025	Виконано

Дата видачі завдання *24* листопада 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ *проф. Гавриленко С. Ю.*
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи магістра: 86 с., 2 табл., 33 рис., 2 додатока, 22 джерел інформації.

ВЕЛИКІ МОВНІ МОДЕЛІ, КЛАСТЕРИЗАЦІЯ ОБ'ЄКТІВ, КОМП'ЮТЕРНИЙ ЗІР, МЕТРИКИ ЕФЕКТИВНОСТІ, ТРЕКІНГ ОБ'ЄКТІВ, ФУТБОЛЬНА АНАЛІТИКА, GRAPHQL, YOLO

Об'єктом дослідження є процес аналізу футбольних матчів з побудовою метрик ефективності та рекомендаційною підтримкою тренерських рішень на основі відеозаписів ігрових епізодів.

Предметом дослідження є методи комп'ютерного зору для розпізнавання об'єктів на футбольному полі, підходи до автоматичного отримання та формалізації метрик ефективності, а також технології інтеграції цих метрик з великими мовними моделями для генерації тактичних рекомендацій.

Мета дослідження – розробка системи аналізу футбольних матчів, що забезпечує автоматичне розпізнавання гравців, м'яча та суддів на відеозаписах, побудову метрик ефективності та формування рекомендацій для тренерського штабу на основі отриманих даних.

Методи дослідження передбачають системний підхід, аналіз існуючих систем, літератури, методів та алгоритмів, застосування комп'ютерного зору та обробку інформації за допомогою великих мовних моделей.

Сфера застосування – професійний і аматорський футбол для поглибленого аналізу матчів і тренувань, спортивна аналітика для підготовки експертних звітів, освітні програми зі спортивної тактики та підготовки тренерів.

ABSTRACT

Qualification work: 86 pages, 2 tables, 33 figures, 2 appendixes, 22 information sources.

LARGE LANGUAGE MODELS, OBJECT CLUSTERING, COMPUTER VISION, PERFORMANCE METRICS, OBJECT TRACKING, FOOTBALL ANALYTICS, GRAPHQL, YOLO

The object of the research is the process of analyzing football matches with the construction of performance metrics and providing recommendation support for coaching decisions based on video recordings of match episodes.

The subject of the research is the methods of computer vision for recognizing objects on the football field, approaches to the automatic extraction and formalization of performance metrics, as well as technologies for integrating these metrics with large language models to generate tactical recommendations.

The aim of the research is the development of a system for analyzing football matches that enables automatic recognition of players, the ball, and referees in video recordings, construction of performance metrics, and generation of recommendations for the coaching staff based on the obtained data.

The research methods include a systematic approach, analysis of existing systems, literature, methods and algorithms, application of computer vision techniques, and processing of information using large language models.

The field of application includes professional and amateur football for in-depth analysis of matches and training sessions, sports analytics for preparing expert reports, and educational programs in football tactics and coaching.

ЗМІСТ

ВСТУП

Футбол як найпопулярніший командний вид спорту у світі характеризується надзвичайно високим рівнем конкуренції, що стимулює постійний пошук інноваційних підходів до підвищення ефективності команд. За останні роки сфера спортивної аналітики зазнала значних трансформацій завдяки впровадженню технологій штучного інтелекту та комп'ютерного зору, які дозволяють отримувати об'єктивні кількісні дані про гру, недоступні при традиційному візуальному спостереженні.

Традиційно аналіз футбольних матчів здійснювався вручну – тренери та аналітики переглядали записи матчів, фіксували ключові моменти та формували висновки на основі власного досвіду та інтуїції. Однак такий підхід має суттєві обмеження: він є суб'єктивним, трудомістким, потребує високої кваліфікації та не дозволяє отримати точні кількісні показники фізичної активності гравців. Професійні команди використовують дороге обладнання – GPS-трекери, спеціалізовані камери високої роздільності, датчики руху – що робить об'єктивний аналіз недоступним для аматорських команд, шкільних та університетських спортивних секцій.

Водночас розвиток методів глибокого навчання, зокрема згорткових нейронних мереж для детекції об'єктів, алгоритмів трекінгу та методів комп'ютерного зору відкриває можливості для створення доступних систем автоматизованого аналізу на основі звичайних відеозаписів матчів. Це дозволяє демократизувати спортивну аналітику, зробивши її інструментом не лише для елітних клубів, але й для широкого кола спортивних колективів.

Тому задача кваліфікаційної – це розробити та реалізувати інформаційну систему автоматизованого аналізу футбольних матчів на основі відеозаписів, яка забезпечує детекцію та трекінг гравців, обчислення метрик фізичної та тактичної ефективності, візуалізацію просторових характеристик гри та генерацію персоналізованих рекомендацій для підтримки тренерських рішень.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис сучасного стану розвитку інформаційних систем аналізу футбольних матчів

Сучасний стан всіх сфер життя суттєво змінився після розвитку штучного інтелекту (ШІ) і комп’ютерного зору та продовжує стрімко змінюватись. Суттєвий вплив також відбувся і у сфері спорту, де однією з галузей застосування цих технологій став футбол – найпопулярніший вид спорту, де витрачаються мільйони на покращення результатів команд і невеличкі переваги у вивчені команди суперника може призвести до перемоги.

Наразі вже існують великі інформаційні системи, що виконують аналіз матчів і допомагають суддям виконувати їх роботу під час великих матчів, де змагаються найкращі команди [1]. Команди ж і клуби почали використовувати штучний інтелект для відбору кандидатів до команди, де штучний інтелект у парі з комп’ютерному зору оцінює ефективність гравця на полі та надає об’єктивне рішення щодо нього.

Інформаційні системи аналізу футбольних матчів зазвичай охоплюють кілька ключових компонентів, а саме:

– перший – це збір і обробка інформації, що відбувається на полі. Зазвичай це виконується за допомогою комп’ютерного зору, але іноді застосовують і додаткові пристрої, що дозволяють більш точно і об’єктивно отримувати інформацію такі як GPS-трекер;

– другий – це розрахунок метрик на базі зібраної і обробленої інформації такі як середня швидкість пересування, подолана дистанція, кількість результативних дій, наприклад, пасів або ударів, тощо;

– третій – це аналітична та рекомендаційна підтримка, де за допомогою спеціальних моделей які натреновані для аналізу футбольних подій і метриків виконуються пошук оптимального рішення на базі існуючих даних.

Такі інформаційні системи дозволяють отримати можливість аналізувати не тільки поточні проблеми, а й запобігти майбутнім або передбачити їх. Завдяки

цьому відбувається поступовий перехід від традиційного аналізу після матчу до інтерактивних інтелектуальних систем підтримки прийняття рішень, які працюють у реальному часі.

1.2 Огляд існуючих інформаційних систем аналізу футбольних матчів

На поточний стан існує кілька інформаційних систем, які виконують аналіз футбольних матчів. Серед них можна виділити найбільш популярні і закінчені системи якими користуються у високих лігах футболу.

1.2.1 Огляд існуючої інформаційної системи «Hudl»

Однією з найбільших та найпоширеніших інформаційних систем, що займаються аналізом спортивних матчів, є «Hudl», який орієнтований на відеоаналізі гри. Цей сервіс позиціонує себе не лише як платформа для аналізу футбольних матчів, а включає в себе ще кілька найпопулярніших видів спорту На рисунку 1.1 зображено головну сторінку системи.

Функціональність даної системи базується на інформації, що отримується з відеозапису, де в результаті виконується розмічення ігрових моментів, наприклад, виконання пасу, удару по воротам, відбір м’яча тощо. В результаті ми отримуємо інтерактивний звіт по матчу (див. рисунок 1.2), де виділені ключові моменти, зображено позиції гравців і місця де виконувались дії – це дозволяє аналітикам і тренерам виявляти проблеми або закономірності у грі суперника спираючись на об’єктивні метрики, але це потребує високої кваліфікації для правильної інтерпретації показників.

Основні бізнес-функції системи «Hudl»:

– відеоаналіз матчів — автоматичне розпізнавання та розмічення ігрових епізодів із подальшою побудовою інтерактивних звітів;

– збереження та організація відеоматеріалів — формування бібліотеки матчів і тренувань для подальшого аналізу;

– спільна робота команди — можливість для тренерів і гравців коментувати, позначати моменти та обмінюватися аналітичними висновками у межах команди;

– формування статистичних звітів — генерація метрик по кожному гравцю (кількість передач, ударів, відборів, точність дій тощо).

Недоліки системи «Hudl»:

– потребує високої кваліфікації тренера або аналітика для правильної інтерпретації отриманих даних;

– більшість процесів розмічення залишається напівручними, що обмежує швидкість аналізу;

– відсутність автоматизованого формування рекомендацій — система не пропонує тактичних висновків або порад для тренерського штабу;

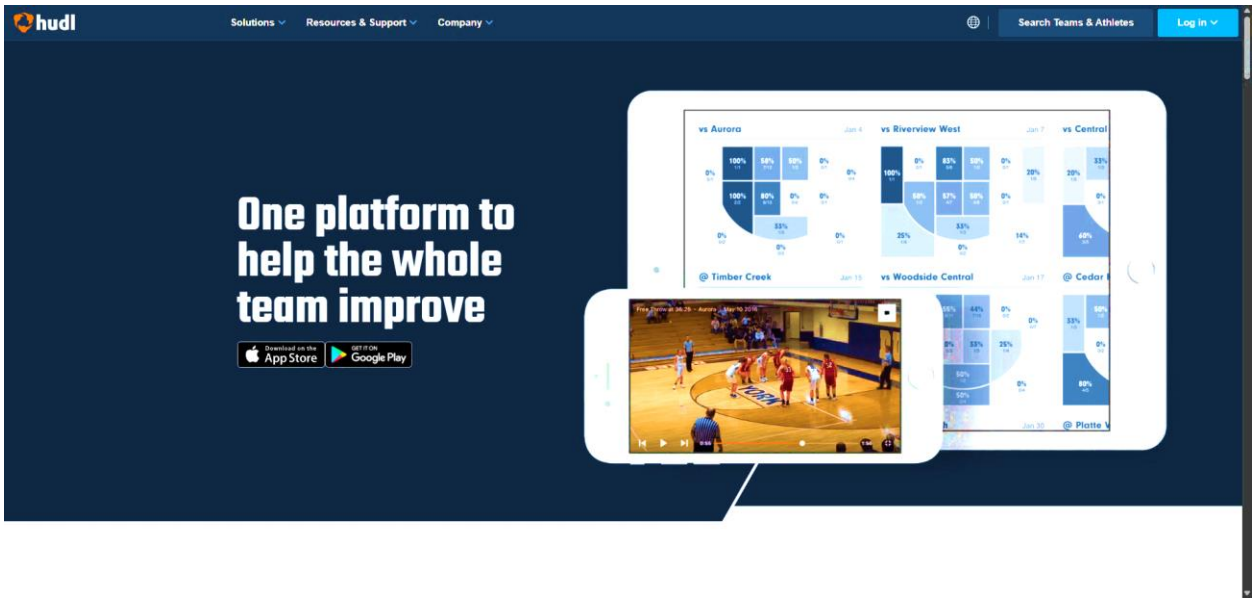


Рисунок 2.1 – Головна сторінка системи «Hudl»

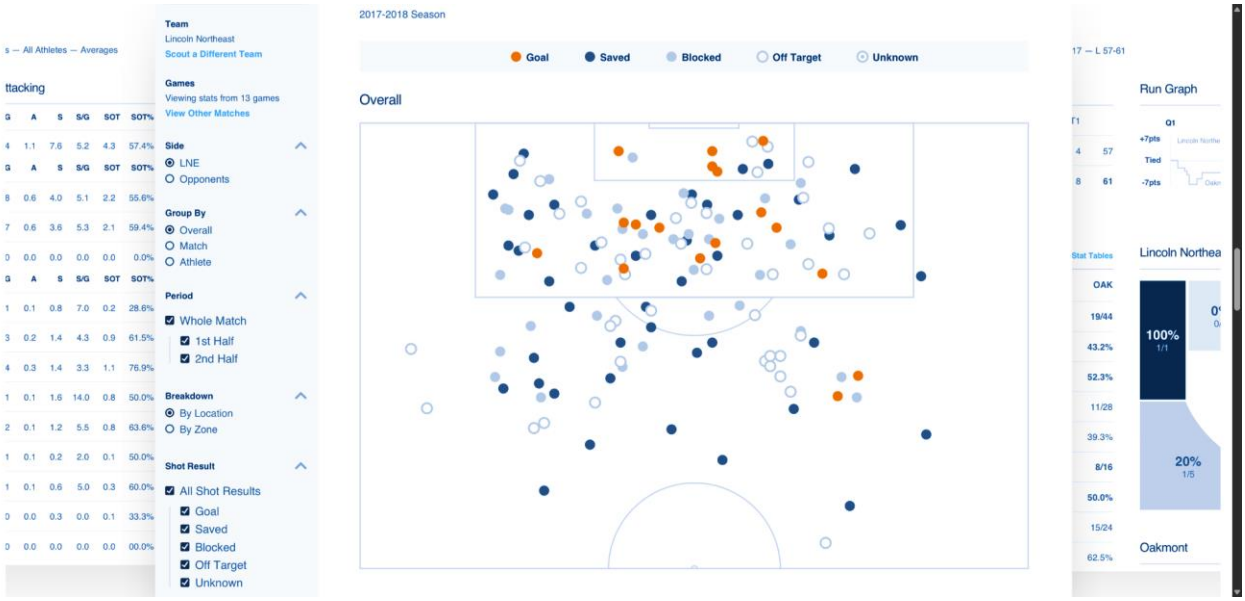


Рисунок 2.2 – Зображення однієї з функцій системи «Hudl»

1.2.2 Огляд існуючої інформаційної системи «Hudl Wyscout»

Інформаційна система «Hudl Wyscout» є однією з найбільш відомих і поширених платформ у сфері футбольної аналітики. Вона має велику базу даних і використовується більшістю провідних футбольних клубів, федерацій і аналітичних центрів. Ця система орієнтована насамперед на скаутів і тренерів команд, які шукають гравців на певні позиції шляхом надавання доступу до аналітичних інструментів. Головна сторінка цієї системи зображена на рисунку 2.3.

Функціональність системи базується на аналітичній обробці статистичних і відеоданих, що надходять із зовнішніх джерел. Система надає можливість тренерам і скаутам передивлятись матч і помічати моменти під час перегляду у ручному режимі які потім можна розібрати з командою. Також система дозволяє переглядати інформацію по певним гравцям, де можна створити таблиці з порівнянням їх на базі різних показників. Система містить розширену базу профілів футболістів, у яких зібрані ключові метрики — кількість пасів, відборів, ударів, перехоплень, швидкість пересування тощо.

Система дозволяє представили метрики у вигляді візуальних графіків, схем та 2D карт (див. рисунок 2.4).

Основні бізнес-функції системи «Hudl Wyscout»:

– аналітика гравців — надання метрик на базі інформації про гравців з детальними статистичними показниками;

– скаутингові інструменти — пошук і фільтрація гравців за позицією, віком, лігою, клубом або типом гри;

– відеоаналіз матчів — можливість перегляду відеозапису матчу з інструментами по розміченню моментів у ручному режимі;

– порівняльний аналіз — створення таблиць і графічних звітів для порівняння декількох гравців між собою;

Недоліки системи «Hudl Wyscout»:

– класифікації подій у відео матча виконується вручну;

– відсутність компонента на базі комп’ютерного зору для отримання власної інформації з відео матча, що створює залежність від сторонніх ресурсів;

– рекомендаційні механізми обмежуються порівнянням показників;

Рисунок 2.3 – Головна сторінка системи «Hudl Wyscout»

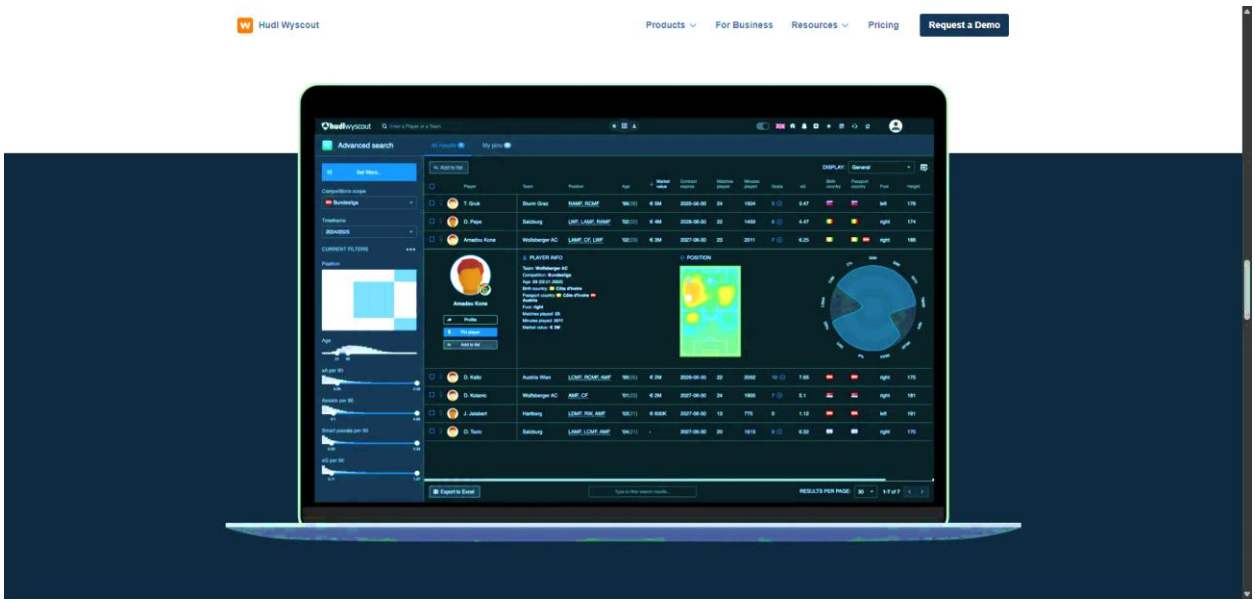


Рисунок 2.4 – Зображення однієї з функцій системи «Hudl Wyscout»

1.2.3 Огляд існуючої інформаційної системи «Metrica Sports»

Інформаційна система «Metrica Sports» має сильне поглиблення у інструменти для відеоаналізу футбольних матчів і є одним з найпопулярніших рішень у даній задачі. Основний напрям у цій системі лежить у автоматичному виділені подій на полі, визначення координат гравців і м’яча, що дозволяють створювати візуальні звіти для тренерів. Головна сторінка інформаційної системи зображено на рисунку 2.5.

Функціональність системи базується на комп’ютерному зорі який дозволяє автоматично визначати позиції гравців на полі у реальному часі з можливістю визначення майбутнього руху за допомогою машинного навчання. Також для детального аналізу матчів система дозволяє користувачеві у ручному розмічати ігрові моменти з заготовлених вже шаблонів або повністю створений з нуля елемент (див. рисунок 1.6).

Результатом інформаційної системи є аналіз матчу з розміченими ігровими моментами, де може бути наявні найбільш часті позиції гравців, що зображені у вигляді теплової картки гравців, аналіз побудови захисту або нападу команди з зонами які вони покривають. Цей результат дозволяє тренеру візуально

продемонструвати команді проблемні місця або моменти гри та виявити слабкі місця які необхідно покращувати.

Основні бізнес-функції системи «Metrica Sports»:

– автоматичне розпізнавання гравців і м’яча;

– тактичний аналіз який базується на отриманих метрик за допомогою комп’ютерного зору;

– ручне розмічення моментів — можливість створення тренером власних елементів на відео;

– візуалізація показників у вигляді побудови графіків та 2D карт;

Недоліки системи «Metrica Sports»:

– відсутній модуль рекомендаційної підтримки — система не надає порад по покращенню гри під час аналізу, а лише виділяє моменти;

– відсутність скаутінг модуля, що не тренеру використовуючи систему знайти заміну гравцю.

Рисунок 2.5 – Головна сторінка системи «Metrica Sports»

Рисунок 2.6 – Зображення однієї з функцій системи «Metrica Sports»

2 ПОСТАНОВКА ЗАВДАННЯ НА ДОСЛІДЖЕННЯ ТА РОЗРОБКУ

Метою кваліфікаційної роботи є створення інформаційної системи для автоматизованого аналізу футбольних матчів на основі технологій комп'ютерного зору та штучного інтелекту, що забезпечить об'єктивну оцінку ефективності команд і окремих гравців та сформує інтелектуальні рекомендації для підтримки тренерських рішень.

Для досягнення поставленої мети необхідно дослідити існуючи методи на технології, що можуть забезпечити автоматичну обробку відеозаписів матчів та розрахунок метрик ефективності і на базі виконаного дослідження необхідно розробити інформаційну систему, що забезпечуватиме аналіз футбольних матчів на базі комп’ютерного зору для виконання об’єктивної оцінки команди та окремих гравців і за допомогою штучного інтелекту отримувало на базі цих оцінок рекомендації, що спрямовані на підвищення ефективності роботи тренера.

2.1 Формування задачі на дослідження

Задачею на дослідження є аналіз існуючих підходів до автоматизованого збору та обробки даних з відеозаписів футбольних матчів для визначення ефективності гравців та команд.

Фундаментальною основою дослідження є аналіз підходів до визначення ефективності футболістів. Під ефективністю гравця розуміється оцінка його внеску у гру команди, що включає фізичні показники (подолана дистанція, швидкість), технічні дії (паси, перехоплення, удари, втрати м'яча), тактичні показники (ефективність позиційних дій, володіння м'ячем) та результативність (голи, гольові передачі). Необхідно визначити, які показники є ключовими для побудови індексу ефективності гравця (PPI) та як вони співвідносяться між собою. Особливу увагу слід приділити складу фізичних метрик, що можуть бути розраховані на основі відеоаналізу: загальна подолана дистанція та середня швидкість руху гравця. Дослідження має визначити математичні моделі розрахунку цих показників на основі траєкторій руху гравців, а також методи розрахунку командних метрик володіння м'ячем.

Для реалізації системи автоматизованого збору метрик необхідно проаналізувати сучасні технології комп'ютерного зору, методи визначення просторового положення об'єктів та алгоритми їх ідентифікації, враховуючи технічні обмеження вихідного відеоматеріалу та обчислювальних ресурсів.

Результатом дослідження має стати раціональний вибір технологій та методів для реалізації системи автоматизованого аналізу футбольних матчів.

2.1.1 Аналіз моделей комп'ютерного зору для обробки відео

Необхідно проаналізувати сучасні моделі глибокого навчання для детекції та трекінгу об'єктів на відеозаписах футбольних матчів. Аналіз має охопити архітектури згорткових нейронних мереж для детекції гравців, м'яча та суддів, такі як YOLO, EfficientDet, Detectron2, а також алгоритми трекінгу об'єктів між кадрами, зокрема ByteTrack. Дослідження має враховувати обмеження щодо частоти кадрів відео та вимог до обчислювальних ресурсів.

2.1.2 Аналіз методів визначення координат гравців на полі

Необхідно проаналізувати методи перетворення координат гравців з системи координат кадру відео у систему координат реального футбольного поля. Дослідження має охопити методи автоматичного визначення ключових точок поля (лінії розмітки, кути воріт, центральне коло, штрафні майданчики) за допомогою комп'ютерного зору та алгоритми побудови матриці гомографії для проєктивного перетворення координат.

2.1.3 Аналіз алгоритмів класифікації гравців за командною належністю

Необхідно проаналізувати алгоритми розпізнавання командної належності гравців на основі візуальних ознак їх спортивної форми. Класифікація здійснюється за кольором джерсі гравців з використанням методів аналізу кольору у різних колірних просторах (RGB, HSV, LAB), виділення домінантного кольору

через кластеризацію (K-means, Mean Shift) та методів машинного навчання для класифікації з урахуванням варіацій освітлення.

2.2 Формування задачі на розробку

Задача розробки є створення інформаційної системи, що буде обробляти на базі комп’ютерного зору відеозапис футбольного матчу і за допомогою штучного інтелекту аналізувати результат його для автоматизованого отримання та розрахунку метрик ефективності гравців і команд, а також формування інтелектуальних рекомендацій для підтримки тренерських рішень.

Вхідними даними до інформаційної системи повинно бути відеозапис матчу на якому достатньо видно гравців, їх джерсі (що включає колір та номер на спині), а також розмітку на полі для успішного використання алгоритму Гомографії для відображення позиції гравця у вигляді 2D-карти.

Вихідними даними обробки відеозапису є визначенні метрики такі як середня швидкість гравця, дистанцію яку він подолав, процент володінням м’ячом командою, кількість успішних дій у вигляді пасів або перехоплень, відеозапис з розміченими гравцями та м’ячом, згенеровані 2D карти з зображення позицій гравців та зони їх контролю. На базі отриманих метрик відбувається їх аналіз штучним інтелектом, а саме мовною моделлю (LLM), яка налаштована за допомогою prompt engineering на створення рекомендацій для тренера та аналітика. Відображення вихідних даних відбувається за допомогою веб-застосунку з яким взаємодії користувач.

2.2.1 Основні функціональні можливості системи

Система повинна мати такі основні функціональні можливості:

– можливість обробки відеозаписів футбольних матчів, що включає:

а) завантаження відеофайлів матчів у поширених форматах (MP4, AVI);

б) автоматичне розпізнавання гравців, м'яча та суддів на кожному кадрі;

в) трекінг об'єктів протягом усього матчу з присвоєнням унікальних ідентифікаторів;

г) класифікація гравців за командною належністю на основі кольору джерсі;

д) збереження результатів розпізнавання для можливості повторного використання без перерахунку.

– можливість розрахунку метрик на базі отриманих даних, що включає:

а) визначення координат гравців на реальному полі за допомогою гомографії;

б) розрахунок фізичних метрик (подолана дистанція, середня швидкість);

в) розрахунок командних метрик (володіння м'ячем у відсотках);

г) обчислення індексу ефективності гравця (PPI) на основі зважених показників.

– можливість отримання візуальних результатів обробки відеозапису матчу, що включає:

а) створення анотованого відеозапису з виділеними гравцями, м'ячем;

б) генерація 2D тактичних карт з відображенням позицій гравців.

– можливість формування рекомендацій для тренерського штабу на базі метрик, що включає:

а) інтеграція з API великої мовної моделі для аналізу метрик;

б) генерація текстових рекомендацій для тренера на основі результатів матчу.

2.2.2 Нефункціональні вимоги

Система повинна відповідати таким нефункціональним вимогам:

– система повинна виконувати розпізнання гравців на полі та м’яча з середньої точністю більше 80%;

– система повинна надавати рекомендації за наявними метриками які були зібрані після аналізу матча;

– система повинна витримувати навантаження не менше 500 одночасних користувачів;

– архітектура системи повинна бути масштабованою та включати щонайменше три мікросервіси, що відповідають за розпізнавання, обчислення метрик та надання обробленої інформації користувачам.

3 ДОСЛІДЖЕННЯ ІСНУЮЧИХ МЕТОДІВ ТА ТЕХНОЛОГІЙ АНАЛІЗУ ФУТБОЛЬНИХ МАТЧІВ

3.1 Огляд існуючих методів та стратегій аналізу ефективності гравця на полі

Сучасні дослідження пропонують поєднувати класичні подієві дані (пас, удар, відбір, перехоплення) із трекінговими параметрами (позиція, швидкість, дистанція, прискорення), що дозволяє отримати об’єктивну оцінку внеску кожного футболіста у гру команди.

3.1.1 Підходи до визначення ефективності гравців

Серед основних підходів, що зазначаються виділяють наступні три:

– подієвий аналіз – це аналіз при якому оцінюється кількість і якісь дій гравця, таких як удари, пас, втрата м’яча та інші;

– трекінговий аналіз – це аналіз при якому враховується місцеположення гравця на полі і його пересування. Наприклад отримані через системи розпізнання гравців на відео або за допомогою додаткових інструментів таких як GPS-трекери;

– контекстний аналіз – це аналіз при якому враховується події навколо гравця так само як і події самого гравця. Наприклад серед подій які враховуються до контексту можна виділити відкриті зони, потенційні паси або удари та розташування інших гравців на полі.

Комбінація цих підходів використовується для вираховування ефективності гравців у вигляді індексу ефективності гравця (Player Performance Index), що широко використовується у високих лігах. Згідно з McHale та Scarf [2], цей індекс є ваговою сумою показників, які характеризують як атакувальні, так і оборонні дії футболіста.

3.1.2 Метрики оцінки ефективності гравця

Згідно підходам Oliva-Lozano et al. [3] та Pappalardo et al. [4], що описані в роботах – типові показники, що використовуються для розрахунку PPI, поділяють на три групи:

– фізичні показники – середня швидкість v_{avg}, подолана дистанція d;

– технічні показники – кількість успішних передач p_{acc}, успішних відборів м'яча t_{succs}, кількість втрат м'яча l;

– тактичні показники – ефективність позиційних дій a_{eff}, володіння м'ячом pos.

У результаті ці показники об'єднуються у формулу, що зображена, яка сумує метрики з урахуванням їх важливості для певної ролі, яка дозволяє отримати узагальнену оцінку ефективності.

$$PPI = w_1 * \frac{v_{avg}}{v_{max}} + w_2 * \frac{d}{d_{max}} + w_3 * p_{acc} + w_4 * t_{succ} - w_5 * l + w_6 * a_{eff} + w_7 * pos$$

Де w_i – це ваговий коефіцієнт, що відображає важливість метрики в залежності від ролі гравця, наприклад, нападнику коефіцієнт середньої швидкості є більш важливим, аніж захиснику, а захиснику важливіші успішні відбори. Саме ж значення для цього вагового коефіцієнту лежить у діапазоні від 0 до 1 і отримується за допомогою аналізу великої кількість статистичних даних і за допомогою машинного навчання динамічна змінюється. Для визначення цих коефіцієнтів у разі відсутності великої кількості статистичних даних використовують евристичний аналіз, де у ручному режимі коригуються ці коефіцієнти.

На основі проаналізованих джерел [3][4] та з урахуванням функціональних обов'язків гравців різних позицій, у таблиці 3.1 наведено рекомендовані значення вагових коефіцієнтів для розрахунку індексу ефективності гравця.

Таблиця 3.1 – Рекомендовані значення вагових коефіцієнтів PPI в залежності від позиції гравця

Коефіцієнт	Метрика	Воротар	Захисник	Півзахисник	Нападник
w_1	v_{avg}	0,05	0,1	0,15	0,2
w_2	d	0,05	0,15	0,2	0,15
w_3	p_{acc}	0,15	0,2	0,25	0,15
w_4	t_{succs}	0,1	0,25	0,15	0,05
w_5	l	0,25	0,15	0,1	0,1
w_6	a_{eff}	0,3	0,1	0,1	0,15
w_7	pos	0,.1	0,05	0,05	0,2

Визначення коефіцієнти обґрунтовуються особливостями ігрових функцій кожної з позицій. Для позиції воротаря значення є ефективність позиційних дій (0,30), оскільки правильний вибір позиції є визначальним фактором успішної гри на останній лінії оборони, а також коефіцієнт штрафу за втрати м'яча (0,25), адже помилка воротаря у більшості випадків призводить до пропущеного голу. Захисник характеризується високим значенням коефіцієнта успішних відборів (0,25), що відображає його основну функцію – перешкоджання атаці суперника, та точності передач (0,20), що забезпечує надійний вихід команди з оборони. Для півзахисника пріоритетними є точність передач (0,25) та подолана дистанція (0,20), оскільки гравці на цій позиції виконують зв'язувальну функцію між лініями та покривають найбільшу площу поля. Нападник має найвищі коефіцієнти середньої швидкості (0,20) та володіння м’ячом (0,20), що відповідає його ролі у завершальній фазі атаки, де швидкість прийняття рішень та контроль м'яча є вирішальними.

Оскільки дані коефіцієнти створені на базі аналізу і не є еталонними значеннями, то відповідно вони можуть змінюватись в залежності від кількості наявної інформації яку можна проаналізувати для більш точної оцінки кожної з дій на позиції гравця.

3.1.3 Контекстні моделі

Окрім подієвого аналізу на поточний час активно розвиваються контекстні моделі, що враховують всю ситуацію (контекст) на полі при аналізу події. Такі моделі дозволяють передбачати ситуацію на полі і дозволяють отримувати рекомендації для гравців.

Згідно з дослідженнями Spearman [5], Decroos та інших [6] та Fernandez та інших [7], контекстні метрики оцінюють внесок кожної дії у зміну ймовірності забитого голу або загальної цінності володіння м’ячом.

Серед найпоширеніших моделей виділяють:

– xG (Очікуваний гол) – це ймовірність того, що атака гравця завершиться голом, де враховується позиції інших гравців на полі, швидкість поточного гравця і його характеристики та інші параметри, що можуть мати вплив на цю подію;

– xA (Очікувана допомога) – це ймовірність того, що передача може стати ключовою для створення гольового моменту на полі;

– EPV (Очікувана вартість володіння) – це оцінка володіння мячом гравцем після кожної дії або зміни його параметрів.

Ці моделі дозволяють будувати інтелектуальні системи, що будуть передбачувати ситуацію на полі. Вони широко використовуються у системах, що аналізують гри у вищих лігах футболу і потребують великої кількості даних і їх точність, оскільки це напряму впливає на правильність результатів.

3.2 Методи комп’ютерного зору для аналізу футбольних матчів

Аналіз футбольних матчів базується на інформації яку ми отримуємо, або з інструментів таких як GPS-трекерів, або опрацювання відеозапису матчу за допомогою комп’ютерного зору, який дозволяє автоматично розпізнати об’єкти на полі, визначити їх координати на полі і визначити їх приналежність до команди за допомогою визначення кольору джерсі.

3.2.1 Технології розпізнавання об’єктів на полі

Першочерговим етапом отримання інформації з відеозапису матчу є розпізнання об’єктів на полі. Для виконання цього етапу застосовуються глибокі згорткові нейронні мережі (CNN) які навчаються на заздалегідь підготовленому датасеті в якому необхідно в ручну розмітити об’єкти. Важливо зауважити, що датасет повинен бути якомога різноманітніший для підвищення точності нейронної мережі, що може включати зміну кута зйомки, освітлення та аспекти які впливають на картинку яка отримується при зйомці [8].

Серед найбільш поширених архітектур для задач детекції у спортивній аналітиці виділяють:

– YoLo – це архітектура використовує підхід при якому вона виконує детекцію об’єктів на кадрі за одну ітерацію, коли у класичній архітектури виконується це у кілька. Операція визначення об’єкта на кадрі відбувається шляхом накладання сітки, де для кожної з частини цієї сітки виконується прогнозування ймовірності наявності об’єкта та координат його рамки, що дозволяє отримати високу швидкість і високу точність архітектури;

– EfficientDet – це архітектура, що була розроблена командою Google Brain і використовує принцип компаундування. Ця архітектура є популярним рішення коли обчислювальні ресурси є обмеженими. Вона складається з трьох компонентів, а саме:

а) EfficientNet Backbone для виявлення ознак;

б) BiFPN для швидкого поєднання багатомасштабних ознак;

в) складове масштабування – це головна особливість цієї архітектури при якому глибина, ширина та роздільна здатність моделі масштабуються разом із використанням одного складового коефіцієнта.

– Detectron2 – це одна з найновіших архітектури, що була розроблена командою Meta AI і вона має модульну структуру, що дозволяє їй бути дуже гнучкою. Якщо порівнювати з YoLo яка пропонує виконувати розпізнання за одну ітерацію, то ця архітектура має багато ітераційний підхід в результаті якого ми отримуємо одну з найвищих точностей, але втрачаємо у швидкості. Detectron2 має два етапи розпізнання, а саме:

а) на першому він сканує зображення і виділяє зону, де найімовірніше знаходиться об’єкти для зменшення навантаження при наступному кроці;

б) на другому етапі він застосовує маску R-CNN на областях, що були поміченні на першому етапі, яка виконує класифікацію кожного регіону за певними категоріями об’єктів і створює маски, які окреслюють межі об’єктів, що забезпечує дуже чітке розмічення одного об’єкта від іншого.

В залежності від ресурсів і цілей аналітики обирають одну з архітектури, але найбільш універсальним і частим вибором є YoLo яка дозволяє оптимізувати процес для обробки відео матча у реальному часі.

3.2.2 Методи відслідковування об’єктів протягом матчу

Для відстежування одних і тих самих об’єктів у послідовних кадрах на сьогодні використовують алгоритм відстежування, що дозволяє присвоїти унікальний ідентифікатор до об’єкту.

Найбільш популярний на сьогодні алгоритм який використовується для вирішення цієї задачі є ByteTrack – цей алгоритм є одним з найновіших і використовує всі розпізнавання об’єкту включно з тими які мали низьку точність. Серед його ключових переваг також зазначається його стійкість до перекритті об’єктів і стабільність порівняно з іншими алгоритмами.

Алгоритм ByteTrack має наступні два етапи:

– на першому етапі алгоритм використовує розпізнання, що мали точність більше ніж 70% з існуючими треками;

– на другому етапі від використовує вже розпізнання, що мали точність від 30% до 70% для відновлення втрачених треків. Цей етап особливо ефективний коли два гравця знаходяться поруч.

3.2.3 Методи визначення координат гравців і м’яча на футбольному полі

Після визначення об’єктів на кадрі наступним необхідним етапом є визначення координат гравців і м’яча на футбольному полі. Ця задача включає кілька проблем які вирішуються комбінацією методів відслідковування і геометричних перетворень. Координати гравців які ми отримуємо під час розпізнання об’єктів на відео знаходяться у системі координат відеокадру (у вигляді трапеції через нахил камери) і для їх перетворення до системи координат реального футбольного поля використовують алгоритм Гомографії.

Алгоритм Гомографії – це геометричне перетворення, що дозволяє проектувати точки з площини камери на площину поля. Процес складається з наступних кроків:

– визначення ключових точок поля на відеокадрі (кути штрафних майданчиків, центральне коло, кутові прапорці);

– встановлення відповідності між точками на зображенні та їх реальними координатами на полі (наприклад, центр поля має координати (0, 0), а розміри стандартного поля 105×68 метрів);

– обчислення матриці гомографії H розміром 3×3, що описує перетворення між двома площинами за допомогою методу найменших квадратів;

– застосування матриці до координат гравців.

Для усунення проблеми при використання алгоритму у вигляді зміни положення камери, що також змінює положення ключових точок, використовують комп’ютерний зір для визначення ключових точок на кадрі відеоряду.

Для отримання точок поля які можна використати для гомографії можна представи поле у декартовій системі координат і оскільки футбольне поле має стандартизовані розміри згідно IFAB (International Football Association Board) [9], то використовуючи еталоні розміри – визначимо координати точок.

Еталонні розміри відповідно до офіційних правил IFAB:

– розміри поля: 12000 × 7000 см;
– штрафна площа: 2015 × 4100 см (глибина × ширина);
– воротарська площа: 550 × 1832 см;
– радіус центрального кола: 915 см;
– відстань до точки пенальті: 1100 см.

Для обчислення координат використовується декартова система координат, де початок (0, 0) розташований у лівому нижньому куті поля, вісь X спрямована вздовж довжини поля (0 – 12000 см), а вісь Y – вздовж ширини (0 – 7000 см). На рисунку 4.1 зображено футбольне поле з ключовими точками у системі координат з розмірами відповідно правилам IFAB.

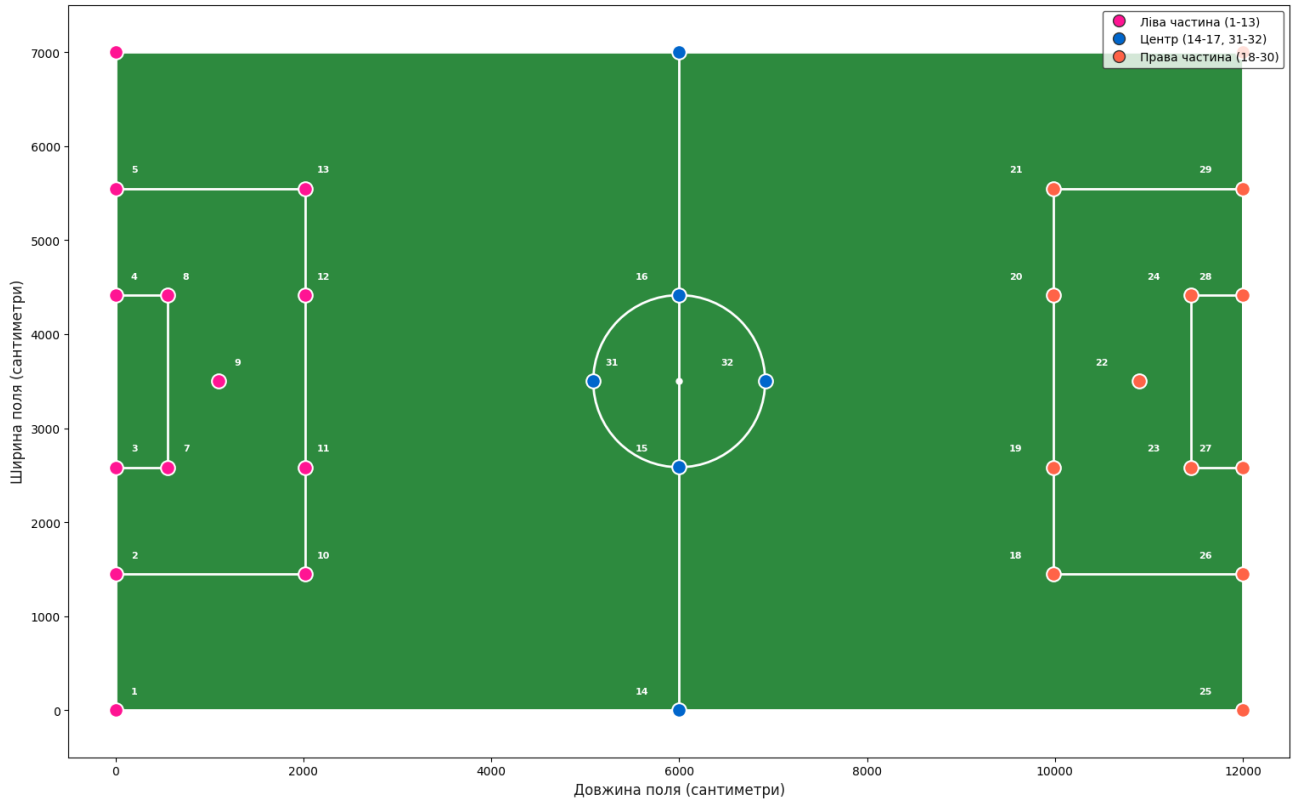


Рисунок 3.1 – Футбольне поле з ключовими точками у системі координат

Координати кутових точок поля обчислень не потребують – це крайні значення осей: (0, 0), (0, 7000), (12000, 0), (12000, 7000). Для внутрішніх елементів розмітки застосовується принцип центрування відносно воріт.

Розглянемо обчислення координат штрафної площі. Оскільки штрафна має бути симетрично розташована відносно центру воріт, необхідно визначити вертикальний відступ від краю поля (від точки 1 до точки 2). Цей відступ обчислюється за формулою:

$$Y_{\text{відступ}} = \frac{W_{\text{поле}} - W_{\text{штрафна}}}{2} = \frac{7000 - 4100}{2} = 1450 \text{ см}$$

Де:

– $W_{\text{поле}}$ – ширина поля, що згідно з правилами дорівнює 7000 см;

– $W_{\text{штрафна}}$ – ширина штрафної зони, що згідно з правилами дорівнює 4100 см.

Таким чином, нижній край штрафної знаходиться на висоті $Y = 1450$ см, а верхній – на $Y = 7000 - 1450 = 5550$ см. Координата X для дальнього краю штрафної дорівнює глибині штрафної площі: $X = 2015$ см.

Аналогічно обчислюються координати воротарської площі за формулою:

$$Y_{\text{ворт\_відступ}} = \frac{W_{\text{поле}} - W_{\text{в\_поле}}}{2} = \frac{7000 - 1832}{2} = 2584 \text{ см}$$

Точка пенальті розташована на відстані 1100 см від лінії воріт точно по центру, тобто її координати: (1100, 3500).

Для центрального кола використовується центр поля (6000, 3500) та радіус 915 см. Бокові точки кола мають координати: ліва – (6000 − 915, 3500) = (5085, 3500), права — (6000 + 915, 3500) = (6915, 3500). Відповідно до лівої половини, а саме дзеркально, вираховуються координати для точок, що знаходяться у правій частині поля.

На рисунку 4.2 детально зображено штрафну та воротарську площі на лівій частині поля з розрахованими координатами ключових точок.

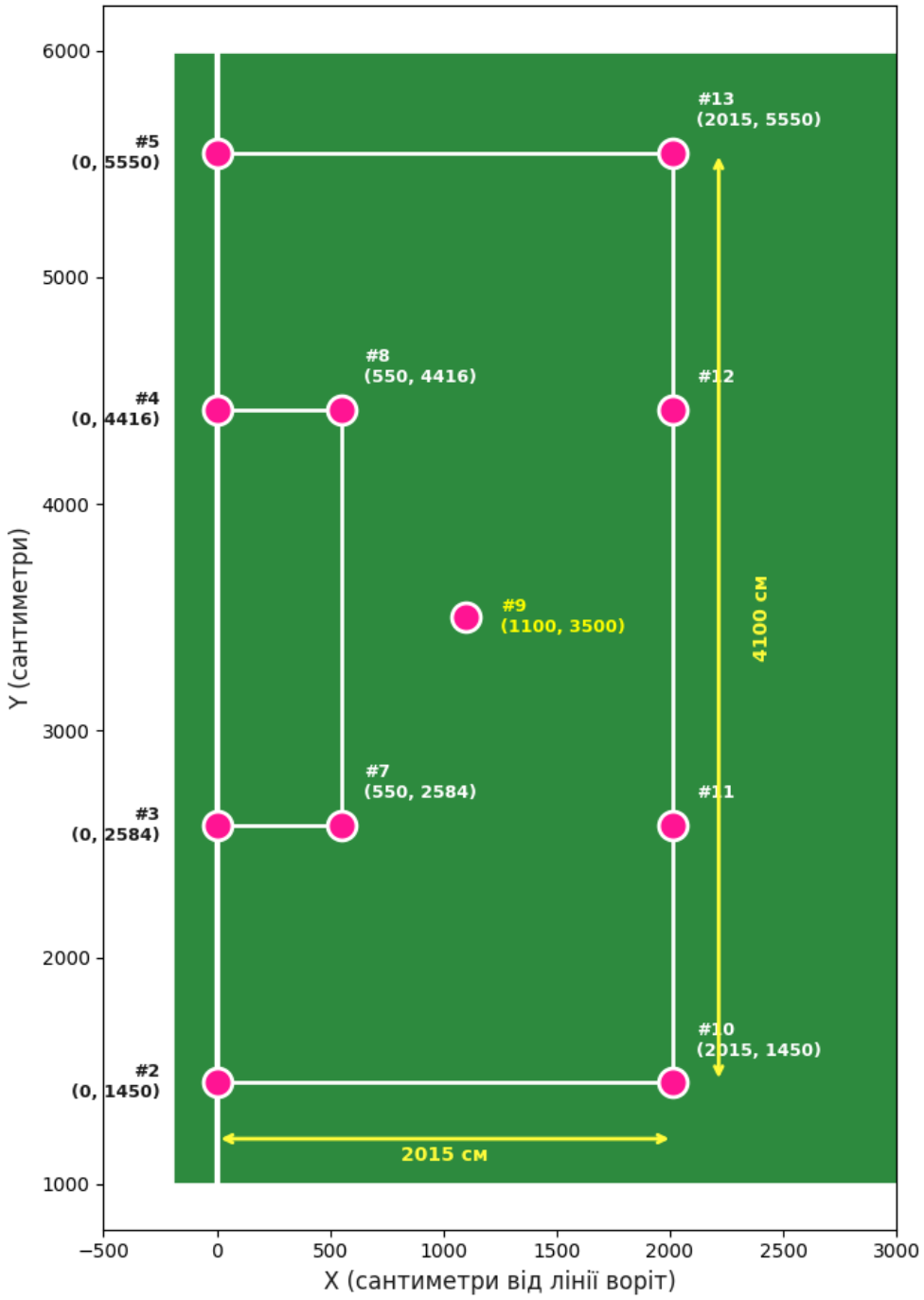


Рисунок 3.2 – Штрафна та воротарська площі з розрахованими координатами для ключових точок

В результаті обчислення координат ключових точок на футбольному полі і визначенню ключових на відеокадрі можна їх застосувати у алгоритмі Гомографії для перетворення перспективи і відображення позицій гравців і мяча відповідно реального поля з кадру відеоряду. Результат перетворення позиції гравця

відеокадру на позиції гравця відповідно реального поля зображено на рисунку 3.3, де зліва зображено приклад кадру з відеоряду, а справа після перетворення.

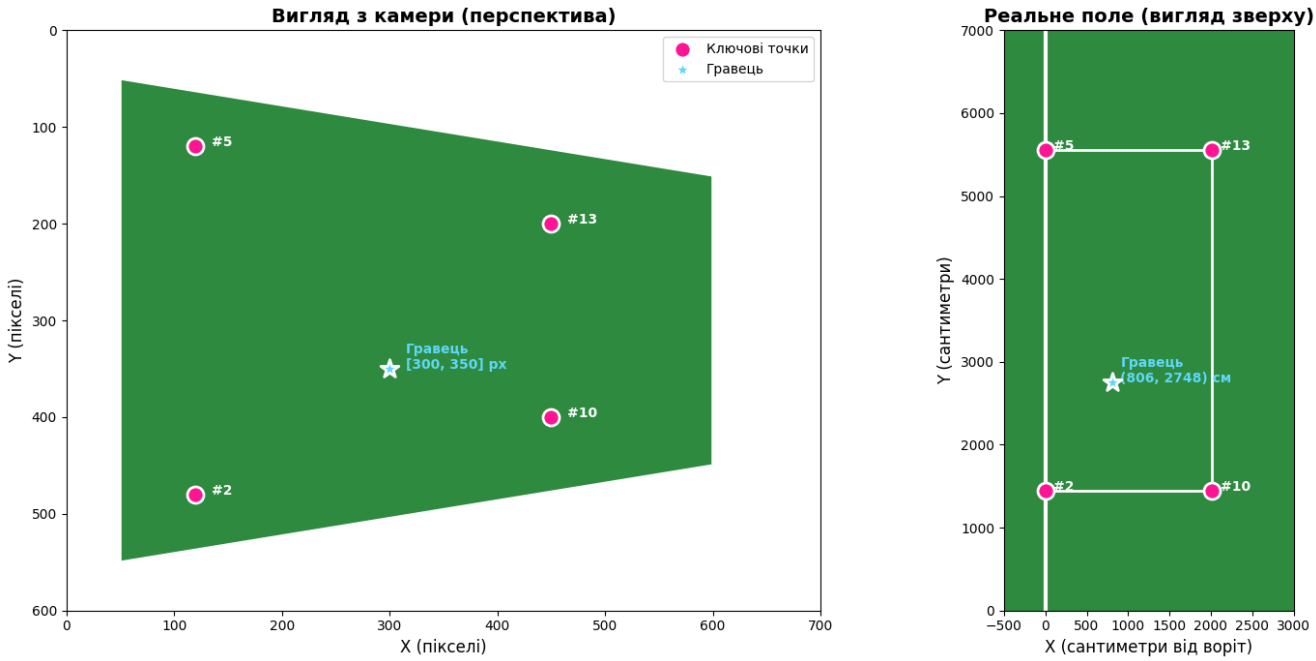


Рисунок 3.3 – Результат перетворення тестового кадру відеоряду за допомогою алгоритму Гомографії

3.2.4 Алгоритми класифікації гравців за кольором форми

Після розпізнання гравців у відеоряді наступним важливим етапом є визначення належність гравця до певної команди для подальшого правильного аналізу. Для розділення на команди застосовується виділення кольору джерсі гравців, оскільки по правилами футболу гравці мають ярко виражену різницю між кольорами форми.

Згідно з дослідженнями Maglo, A. та інших [10] та Ciopra, A. та інших [11], існує декілька основних підходів до класифікації гравців:

– метод на основі колірних гістограм – це метод при якому для кожного з гравця ми будуємо колірну гістограму у певному кольоровому просторі (RGB, HSV та інші) і порівнюємо виділену гістограму гравця з еталонною гістограмою

команди. Проблема цього методу в залежності від освітлення, оскільки наявність або відсутність його сильно впливає на гістограму;

– метод на основі домінантних кольорів – це метод при якому виділяють домінантний колір на джерсі. Це метод має більш велику точність і проблема з освітлення вже менше виражена порівняно з методом на основі колірних гістограм, але все ще має проблеми з тінями та відблисками на джерсі;

– метод машинного навчання з класифікатором – це найбільш точний і стійкий до будь якого шуму метод, але потребує перенавчання на нові кольори джерсі

Серед цього списку особливо поширеним є метод на основі домінантних кольорів через його швидкість і точність роботи.

Як результат виділення кольорів – наступним кроком є розділення гравців на групи. Для цього можна застосувати метод кластеризації K-means, який, згідно з дослідженнями Azzami S. Y. A., та інших [12] та Falaleev, N. S. та інших [13], дозволяє автоматично розділити гравців на групи за подібністю кольорів форми.

Принцип роботи цього метода поділяється на наступні етапи:

– визначення репрезентативної області – цей етап передбачає виділення центральної області гравця, а саме його джерсі;

– застосування підходу визначення домінантного кольору;

– кластеризація, де усі домінантні кольори будуть передані до алгоритму K-means з параметром $K = 2$ (дві команди). Після роботи алгоритму ми отримаємо мітку кластера (0, 1, 2) які будуть ідентифікуватись як команди.

3.3 Методи розрахунку аналітичних метрик на основі даних розпізнавання

Наступним етапом після успішного розпізнання, класифікації гравців та застосування алгоритму Гомографії для проектування точки з площини камери на площину поля відкривається можливість на розрахунок аналітичних метрик як персональних для кожного з гравців, так і агрегованих для розрахунку командної ефективності. Під час розрахунку метрик буде використовуватись координати

гравців на полі отриманих під час розпізнання, що дозволяє об’єктивно оцінити фізичні, технічні та тактичні аспекти гри.

3.3.1 Фізичні метрики активності гравців

Фізичні метрики активності гравця є однієї з базових і головних метрик, що відображають фізичну підготовку гравця і його активність протягом матчу.

Для розрахунку подоланої дистанції гравцем буде застосована формула Евкліда в якій буде використана відстань між двома точками. Тобто розрахунок передбачає сумування відстаней між послідовними позиціями гравця. Формула Евкліда яка буде використана для розрахунку:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Де:

- x_1, y_1 – координати гравця на полі на попередньому кадрі
- x_2, y_2 – координати гравця на полі на поточному кадрі
- d – відстань між позиціями

Загальна дистанція буде отримано за допомогою суми у метрах:

$$D = d_1 + d_2 + \cdots + d_i$$

Для розрахунку середньої швидкості гравця буде застосована формула:

$$V = \frac{d}{t}$$

Де:

- d – відстань між позиціями у метрах
- t – час між кадрами у секундах, який вираховується в залежності від кількості кадрів у відео;
- V – швидкість у метрах за секунду

3.3.2 Метрики володіння м’ячом командою

Наступною по важливістю метрикою після фізичних є метрика володінням м’ячом і при розрахунку цієї метрики буде досліджуватись відсоток контроля м’ячом командами. Умовою для зарахуванням, що гравець команди володіє м’ячом є відстань гравця до м’яча не більше 2 метра.

Для розрахунку відстані від м’яча до гравця буде застосовано формулу Евкліда:

$$d = \sqrt{((x\_\text{гравця} - x\_\text{м}'\text{яча})^2 + (y\_\text{гравця} - y\_\text{м}'\text{яча})^2)}$$

Для розрахунку володіння командою м’ячом у відсотках буде застосовано наступну формулу:

$$Possession = (\text{кадри\_з\_контролем} / \text{загальна\_кількість\_кадрів}) \times 100\%$$

3.4 Використання моделей штучного інтелекту для формування рекомендацій

На сьогодні все більше додаються штучні інтелект для формування рекомендацій користувачам системи і системи для аналізу футбольних матчів не є винятком. Якщо говорити про системи для аналізу футбольних матчів, то такі системи поєднують класичну статистичну аналітику з методами штучного інтелекту. У останніх дослідженнях Dang X та інших [14] зазначається, що застосування інтелектуальних моделей під час аналізу футбольних матчів дозволяє отримати не лише об’єктивну оцінку ефективності гравців на полі, а й може рекомендувати тактики гри на базі останніх результатів і закономірностей які він може виявити.

У межах поточної системи буде використано велику мовну модель (LLM) яка буде інтегровано до системи через GitHub Models API і за допомогою інженерно сформованих інструкцій (prompt engineering) буде налаштовано її роль у вигляді віртуального аналітика та тренера який буде формувати текстові рекомендації щодо тактики команди, зміни розташування гравців чи покращення індивідуальних

показників. Така система дозволяє створити інтелектуальну підсистему підтримки тренерських рішень, що поєднує аналітику з адаптивною інтерпретацією даних, і робить аналіз результатів матчу не лише об’єктивним, але й контекстно зрозумілим для користувача.

3.4.1 Вибір раціональної LLM для вирішення задачі

Для виявлення моделі LLM, що зможе забезпечити раціонально вирішувати поставлену задачу було проведено аналіз доступних з метою вибору оптимального рішення для поставленої задачі. Основними критеріями вибору були:

– ліміти токенів на запит – можливість передачі достатнього обсягу статистичних даних;
– швидкість генерації – забезпечення інтерактивного досвіду користувача;
– якість генерації тексту українською мовою;
– вартість використання та доступність API;
– здатність слідувати складним інструкціям (prompt following).

Порівняння ключових параметрів між популярними моделями наведено у таблиці 3.2.

Таблиця 3.2 – Порівняння популярних моделей

Характеристика	GPT-4o	GPT-4o-mini	GPT-3.5-turbo	Claude 3 Haiku
Контекстне вікно	128K токенів	128K токенів	16K токенів	200K токенів
Швидкість відповіді	Середня	Висока	Висока	Висока
Якість аналізу	Відмінна	Дуже добра	Добра	Дуже добра
Доступність через GitHub Models	Так	Так	Так	Ні

За результатами аналізу було обрано модель GPT-4o-mini з наступних причин:

– достатнє контекстне вікно, що дозволяє передати моделі всю необхідну статистику з додатковою можливістю передачі історії попередніх запитів користувача;

– висока швидкість генерації – ця модель значно швидше за повну версію GPT-4o, забезпечуючи час відповіді 1-3 секунди замість 5-10 секунд;

– доступність через GitHub Models API, що дозволяє використовувати її безкоштовно для розробників через GitHub Models, що спрощує інтеграцію та тестування.

Контекстне вікно у 128K токенів є особливо важливим для даної системи, оскільки дозволяє передавати до LLM:

– повну статистику матчу (команди, гравці, метрики) – ~2-3K токенів;
– системний промпт з інструкціями – ~500 токенів;
– історію діалогу до 10-15 повідомлень – ~2-5K токенів;
– запас для складних запитів користувача.

3.4.2 Prompt Engineering для створення інструкції

Prompt engineering – це методологія формування інструкцій для LLM, що визначають поведінку моделі, формат відповідей та обмеження. Правильно сформований промпт дозволяє досягти отримання якісних та очікуваних відповідей від моделі.

При розробці промпту для системи аналізу футбольних матчів необхідно застосовати наступні принципи prompt engineering:

– визначення ролі – необхідно вказати у інструкції, що модель повинна себе позиціонувати як професійний футбольний тактичний аналітик для тренерів та технічного персоналу. Це налаштовує модель на використання відповідної термінології та рівня деталізації;

– чіткі обмеження – необхідно встановити чіткі обмеження на використання лише наданих даних. Це критично важливо для запобігання

галюцинацій які спричиняють використання недостовірної інформації, що є типовою проблеми LLM, коли модель вигадує неіснуючі факти;

– локалізація відповідей – вказано обов'язкове використання української мови для того, щоб модель надавала відповідь згідно мові групи користувачів на яке націлена дана система;

– визначення аудиторії – необхідно вказати, що відповіді призначені для професійних тренерів, що впливає на рівень деталізації та використання тактичної термінології;

– формат виводу – це дозволяє накласти обмеження на формат виводу моделі для того, щоб відповіді мали необхідну структуру з акцентом на практичні рекомендації, а не загальні футбольні знання;

– негативні інструкції – цей принцип передбачає вказання чого модель НЕ повинна робити, тобто необхідно вказати в інструкції, щоб модель не вигадувала статистику, події поза контекстом і не надавала загальні поради без прив'язки до конкретних даних;

– інструкції для аналізу – цей принцип передбачає вказання конкретних вказівок щодо виявлення тактичних паттернів, статистичних аномалій та формування рекомендацій з посиланням на конкретні дані гравців.

Застосування цих принципів дозволяє мінімізувати ризик отримання відповіді яка буде базуватись на вигаданих даних, забезпечуючи при цьому практичну цінність рекомендацій для тренерського штабу.

4 РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ВИРІШЕННЯ ЗАДАЧІ

На базі виконаних досліджень методів і технологій (розділ 2) та постановки задачі (розділ 3) було сформовано вимоги та розроблено архітектуру інформаційної системи аналізу футбольних матчів. В цьому розділі описуються технічні рішення які були прийнятті під час розробки – це включає структуру базу даних та ключові моменти розробки.

4.1 Обґрунтування вибору технологій та архітектурних рішень

Під час проєктування інформаційної системи було прийнято ряд рішення з урахуванням специфіки аналізу футбольних матчів та обмежень технологій, що передбачаються у використанні для досягнення максимальної продуктивності і відмовостійкісті, масштабованості системи.

4.1.1 Вибір мікросервісної архітектури

Для реалізації інформаційної системи було прийнято рішення про використання мікросервісної арехітектури замість монолітної, що передбачає поділення системи на кілька мікросервісів між якими будуть розподіленні задачі. Згідно з дослідженнями Meijer, W. та інших [15] та Richardson C. [16], мікросервісна архітектура забезпечує кращу масштабованість та стійкість до відмов порівняно з монолітними системами, що є критичним для систем обробки відео з високими вимогами до обчислювальних ресурсів.

Серед причин які є ключовими у цьому виборі можна виділити:

– розділення відповідальності між компонентами – кожен компонент має свою відповідальність у системі, що дозволяє спростити розробку за допомогою використання інструментів незалежно від іншої частини системи і спростить

підтримку коду через його логічне розділення. Система передбачає наступні компоненти:

а) обробка відео (комп'ютерний зір, машинне навчання);

б) розрахунок метрик (математичні обчислення, агрегація даних);

в) управління метаданими (бізнес-логіка, інтеграція з LLM);

г) клієнтський інтерфейс (візуалізація, взаємодія з користувачем).

– незалежне використання ресурсів – кожен з компонентів різні вимоги до ресурсів, тому мікросервісна архітектура дозволяє їх розділити, що є економічно вигідним рішення. Опис використання ресурсів компонентами:

а) компонент по обробці відео потребує значних обчислювальних ресурсів (CPU/GPU) для роботи YOLOv8;

б) компонент по розрахунку метрик потребує швидкого доступу до пам'яті для обчислень;

в) компонент по управлінню метаданими має помірне навантаження без значного якогось використання;

г) компонент для клієнтського інтерфейсу масштабується горизонтально залежно від кількості користувачів і не потребує значного якогось використання через можливість кешування.

– гнучкість виборку інструментів – за допомогою розділення на мікросервіси відкривається можливість обирати стек технологій для кожного з них відповідно задачі, що дозволяє оптимізувати розробку, а саме:

а) Python для комп'ютерного зору (багата екосистема CV бібліотек);

б) TypeScript + Bun для високопродуктивних обчислень;

в) TypeScript + Node.js для бізнес-логіки з великою екосистемою, де стабільність є найбільш важливою.

– можливість ізоляції відмов – коли один з мікросервісів виходить з ладу, то всі інші продовжують виконувати свої задачі через їх атомарність в системі. Тому при виході з ладу, наприклад, мікросервісу по обробці відео користувачі зможуть продовжувати використовувати систему у вигляді отриманні вже оброблених матчів і даних які надає мікросервіс по управлінню метаданими системи;

– незалежне розгортання – це надає нам можливість оновлювати один з компонентів системи не вимикаючи інші, що дозволяє мінімізувати downtime системи.

4.1.2 Вибір мови програмування та фреймворків

Для реалізації компонента по обробці відео було прийнято рішення використання мови програмування Python через кількість готових рішень [17], що підходять для вирішення задач, що ставляться перед цим компонентом. Ключові готові рішення, що було задіяно у цьому компоненті:

- ultralytics – офіційний SDK для YOLOv8 з простим API;
- OpenCV – найпопулярніша бібліотека для обробки зображень;
- NumPy – високопродуктивні операції з масивами (координати треків);
- supervision – інструменти для аналізу спортивних відео.

За допомогою використання Python у зв'язку з PyTorch з'являється можливість використання GPU для використання під час розпізнавань за допомогою YoLo, що у свою чергу прискорило цей процес у 3 рази порівняно з використанням CPU. У свою чергу як веб-фреймворк для обробки запитів до цього компоненту було обрано FastAPI, що є популярним рішенням і надає можливість використовувати асинхронну архітектуру для обробки довгих запитів і дозволяє автоматично згенерувати OpenAPI документацію для майбутньої підтримки.

Для реалізації компоненту для обчислення метриків було обрано мову програмування TypeScript з середою виконання Bun. Мова програмування TypeScript дозволяє забезпечити типізацію під час обчислення, що дозволяє уникнути помилок під час runtime виконання цих операцій. У свою чергу як веб-фреймворк було обрано Elysia.js, що забезпечує ефективні інструменти для використання середовища Bun для отримання максимальної швидкості обробки запитів до цього компоненту. Якщо порівнювати цей компонент з використанням Python, то Python не надає типізацію, що може призвести до помилок у runtime та має гіршу продуктивність у обробці запитів порівняно з Bun.

Для реалізації компоненту по управлінню метаданими системи було обрано мову програмування Typescript та середу виконання Node.js. Оскільки це основний компонент з яким користувач буде взаємодіяти через клієнтський компонент, то необхідно забезпечити максимальну стабільність цього компоненту, тому було обрано найпопулярніші інструменти для цього, що надає веб-фреймворк Nest.js, а саме:

- Архітектурна зрілість: NestJS надає структуру для великих додатків:
  а) Dependency Injection для управління залежностями;
  б) Модульна архітектура (modules, providers, controllers);

в) Декоратори для чистого коду.

– Вбудована підтримка GraphQL: Інтеграція з Apollo Server "з коробки":
  а) Code-first підхід (генерація схеми з TypeScript класів);
  б) Автоматична валідація через class-validator;
  в) Middleware для авторизації.
– Інтеграція з Prisma ORM: Prisma 5.0+ забезпечує:
  а) Type-safe доступ до PostgreSQL;
  б) Автоматичну генерацію TypeScript типів з бази;
  в) Міграції через prisma migrate;
  г) Ефективні запити з eager/lazy loading.

Також у компоненті для управління метаданими прийнято рішення використовувати Graphql, оскільки згідно з Buna S. [18], GraphQL зменшує over-fetching даних на 30-50% порівняно з REST API, що критично важливо для мобільних клієнтів з обмеженим трафіком.

4.1.3 Вибір систем управління базами даних

У архітектурі інформаційної системи передбачається використання двох баз даних з різним призначенням, що обумовлено мікросервісною архітектурою та обмеженнями, а саме Postgresql повільніший для великих JSON документів, а у свою чергу MongoDB погано справляється з операціями зв’язування сутностей згідно з дослідженнями [19], тому розділення на дві бази даних дозволяє забезпечити усунення цих проблем.

Для зберігання виконаних розпізнань було обрано MongoDB. Серед причин по яким було обрано NoSQL базу даних можна виділити:

– гнучкість даних – оскільки з часом можуть додаватись нові метрики, то їх додавання не потребує зміну схеми, а також не на всіх кадрах може розпізнатись один з об’єктів;

– можливість зберігання великих вкладених документів – це дозволяє нам зберегти результат розпізнання як один документ з масивом треків. У специфікації MongoDB вказано, що він ефективно обробляє великі JSON документи до 16 МБ;

– швидкий запис – під час обробки відео виконується запис результатів послідовно і MongoDB має автоматичну оптимізацію вставку документів;

– відсутність транзакцій – треки одного матчу зберігаються атомарно (один документ), транзакції між документами не потрібні.

Для зберігання метрик та метаданих було обрано Postgresql. Оскільки всі ці данні мають фіксовану схему з чіткими типами, то застосування реляційної бази даних є можливими. Також реляційна база даних дозволяє описати зв’язки між сутностями, наприклад, відношення сутності гравця до команди яке є один до багатьох. Також Postgresql забезпечує ACID гарантії для забезпечення консистентності метриками за допомогою:

– при видаленні матчу автоматично видаляються всі метрики (CASCADE);
– при розрахунку метрик для команди транзакція гарантує атомарність.

Серед інструментів, що використовуються у компоненті по управлінню метаданими є Prisma ORM, що має найбільш оптимізований і функціональний драйвер для взаємодії з Postgresql.

4.1.4 Вибір технологій клієнтської частини

Для реалізації клієнтської частини було обрано мову програмування Typescript з використанням фреймворка Next.js, що базується на бібліотеці React. Ця комбінація дозволяти отримати SEO оптимізовані сторінки цього компоненту і забезпечить можливість створення статичних сторінок, які можна віддавати багатьому користувачам зменшуючи навантаження на серверну частину системи. Як інструмент комунікації з компонентом для управління метаданими використовується Graphql з допомогою спеціального інструмента Apollo Client, що дозволяє згенерувати типізацію відповідно схемам Graphql, що були створені у компоненті для управління метаданими з використанням Apollo Server. Для створення візуальної частини веб-додатку використовується бібліотека Tailwindcss для стилізації та бібліотеку компонентів Shadcn/ui, що надає базові компоненти, наприклад, кнопки та поля вводу для швидкого створення UI інтерфейсу.

4.1.5 Загальна архітектура системи

На основі вибраних технологій було спроєктовано мікросервісну архітектуру системи, яка складається з чотирьох основних компонентів та зовнішніх сервісів. Архітектуру зображено на рисунку 4.1.

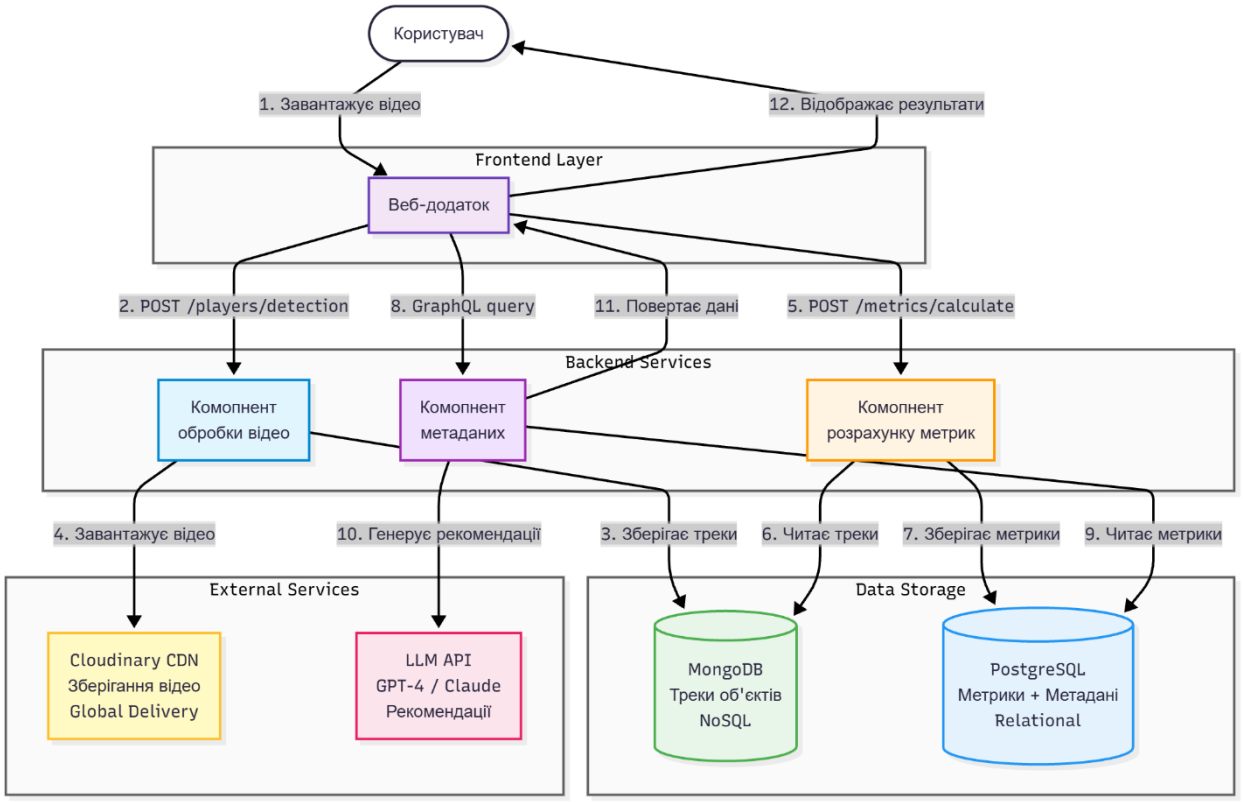


Рисунок 4.1 – Архітектура інформаційної системи

4.2 Проєктування структури баз даних

Система використовує дві бази даних з різним призначенням: MongoDB для зберігання нестандартизованих результатів розпізнавання та PostgreSQL для структурованих метаданих і метрик.

4.2.1 Створення бази даних для структурованих метаданих і метрик

PostgreSQL зберігає всі бізнес-сутності системи: команди, гравці, тренери, матчі, турніри та статистика. Для створення бази даних було описано схему у інструменті Prisma ORM, що дозволяє отримати синхронізувати типізацію для TypeScript та базою даних. Код створення бази даних наведено у додатку А.

Аналізуючи код схеми, що наведено у додатку А можна виділити наступні сутності і їх атрибути:

– сутність Team (Команда) – центральна сутність системи, яка представляє футбольну команду. Атрибути:

а) id – унікальний ідентифікатор команди, VARCHAR довжиною до 191 символів, первинний ключ;

б) name – назва команди, VARCHAR довжиною до 191 символів, унікальне поле;

в) description – опис команди, TEXT, необов'язкове поле;

г) logoUrl – URL логотипу команди, VARCHAR довжиною до 191 символів, необов'язкове поле;

д) foundedYear – рік заснування команди, INTEGER, необов'язкове поле;

е) city – місто базування команди, VARCHAR довжиною до 191 символів, необов'язкове поле;

ж) country – країна команди, VARCHAR довжиною до 191 символів, необов'язкове поле;

з) createdAt – дата створення запису в системі, TIMESTAMP;

и) updatedAt – дата останнього оновлення запису, TIMESTAMP.

– сутність Player (Гравець) – представляє футболіста з повною персональною інформацією. Атрибути:

а) id – унікальний ідентифікатор гравця, VARCHAR довжиною до 191 символів, первинний ключ;

б) name – повне ім'я гравця, VARCHAR довжиною до 191 символів;

в) position – позиція гравця на полі, ENUM (GOALKEEPER, DEFENDER, MIDFIELDER, FORWARD);

г) jerseyNumber – номер на футболці, INTEGER (1-99), необов'язкове поле;

д) imageUrl – URL фотографії гравця, VARCHAR довжиною до 191 символів, необов'язкове поле;

е) email – електронна пошта гравця, VARCHAR довжиною до 191 символів, унікальне поле;

ж) birthdate – дата народження гравця, DATE, необов'язкове поле;

з) height – зріст гравця у сантиметрах, FLOAT, необов'язкове поле;

и) weight – вага гравця у кілограмах, FLOAT, необов'язкове поле;

к) nationality – національність гравця, VARCHAR довжиною до 191 символів, необов'язкове поле;

л) preferredFoot – робоча нога гравця (Left, Right, Both), VARCHAR довжиною до 191 символів, необов'язкове поле;

м) teamId – ідентифікатор команди, VARCHAR довжиною до 191 символів, зовнішній ключ до таблиці Team;

н) createdAt – дата створення запису в системі, TIMESTAMP;

о) updatedAt – дата останнього оновлення запису, TIMESTAMP.

– сутність Match (Матч) – представляє футбольний матч між двома командами. Атрибути:

а) id – унікальний ідентифікатор матчу, VARCHAR довжиною до 191 символів, первинний ключ;

б) tournamentId – ідентифікатор турніру, VARCHAR довжиною до 191 символів, зовнішній ключ до таблиці Tournament, необов'язкове поле (для товариських матчів);

в) homeTeamId – ідентифікатор домашньої команди, VARCHAR довжиною до 191 символів, зовнішній ключ до таблиці Team;

г) awayTeamId – ідентифікатор гостьової команди, VARCHAR довжиною до 191 символів, зовнішній ключ до таблиці Team;

д) scheduledAt – запланована дата та час початку матчу, TIMESTAMP;

е) status – статус матчу, ENUM (SCHEDULED, IN_PROGRESS, COMPLETED, CANCELLED, POSTPONED);

ж) homeScore – рахунок домашньої команди, INTEGER, необов'язкове поле;

з) awayScore – рахунок гостьової команди, INTEGER, необов'язкове поле;

и) venue – місце проведення матчу (назва стадіону), VARCHAR довжиною до 191 символів, необов'язкове поле;

к) videoUrl – URL відеозапису матчу з Cloudinary, VARCHAR довжиною до 191 символів, необов'язкове поле;

л) detectionId – ідентифікатор результату розпізнавання з сервісу object-detection (логічний зв'язок з MongoDB), VARCHAR довжиною до 191 символів, необов'язкове поле;

м) createdAt – дата створення запису в системі, TIMESTAMP;

н) updatedAt – дата останнього оновлення запису, TIMESTAMP.

– сутність PlayerStatistics (Статистика гравця) – зберігає індивідуальні метрики гравця за конкретний матч, отримані компонентом по розрахунку метрик. Атрибути:

а) id – унікальний ідентифікатор запису статистики, VARCHAR довжиною до 191 символів, первинний ключ;

б) matchId – ідентифікатор матчу, VARCHAR довжиною до 191 символів, зовнішній ключ до таблиці Match;

в) playerId – ідентифікатор гравця, VARCHAR довжиною до 191 символів, зовнішній ключ до таблиці Player;

г) minutesPlayed – зіграно хвилин гравцем, INTEGER (0-90+);

д) goals – кількість забитих голів, INTEGER, значення за замовчуванням 0;

е) assists – кількість асистів, INTEGER, значення за замовчуванням 0;

ж) shots – кількість ударів по воротах, INTEGER, значення за замовчуванням 0;

з) shotsOnTarget – кількість ударів у створ воріт, INTEGER, значення за замовчуванням 0;

и) passes – кількість пасів, INTEGER, значення за замовчуванням 0;

к) passesCompleted – кількість точних пасів, INTEGER, значення за замовчуванням 0;

л) tackles – кількість відборів м'яча, INTEGER, значення за замовчуванням 0;

м) interceptions – кількість перехоплень, INTEGER, значення за замовчуванням 0;

н) fouls – кількість фолів, INTEGER, значення за замовчуванням 0;

о) yellowCards – кількість жовтих карток, INTEGER, значення за замовчуванням 0;

п) redCards – кількість червоних карток, INTEGER, значення за замовчуванням 0;

р) distanceCovered – подолана дистанція в кілометрах, FLOAT;

с) averageSpeed – середня швидкість гравця в км/год, FLOAT;

т) maxSpeed – максимальна швидкість гравця в км/год, FLOAT;

у) heatmapData – дані для побудови теплової карти позицій гравця, JSON, необов'язкове поле;

ф) createdAt – дата створення запису в системі, TIMESTAMP;

х) updatedAt – дата останнього оновлення запису, TIMESTAMP.

– сутність TeamStatistics (Статистика команди) – агрегована статистика команди за матч. Атрибути:

а) id – унікальний ідентифікатор запису статистики, VARCHAR довжиною до 191 символів, первинний ключ;

б) matchId – ідентифікатор матчу, VARCHAR довжиною до 191 символів, зовнішній ключ до таблиці Match;

в) teamId – ідентифікатор команди, VARCHAR довжиною до 191 символів, зовнішній ключ до таблиці Team;

г) possession – володіння м'ячем у відсотках (0-100), FLOAT;

д) corners – кількість кутових ударів, INTEGER;

е) offsides – кількість положень поза грою, INTEGER;

ж) createdAt – дата створення запису в системі, TIMESTAMP;

з) updatedAt – дата останнього оновлення запису, TIMESTAMP.

На базі розробленої схеми даних за допомогою інструменту Prisma ORM ми отримали базу даних, що зображено на рисунку 4.2 у вигляді ERD моделі.

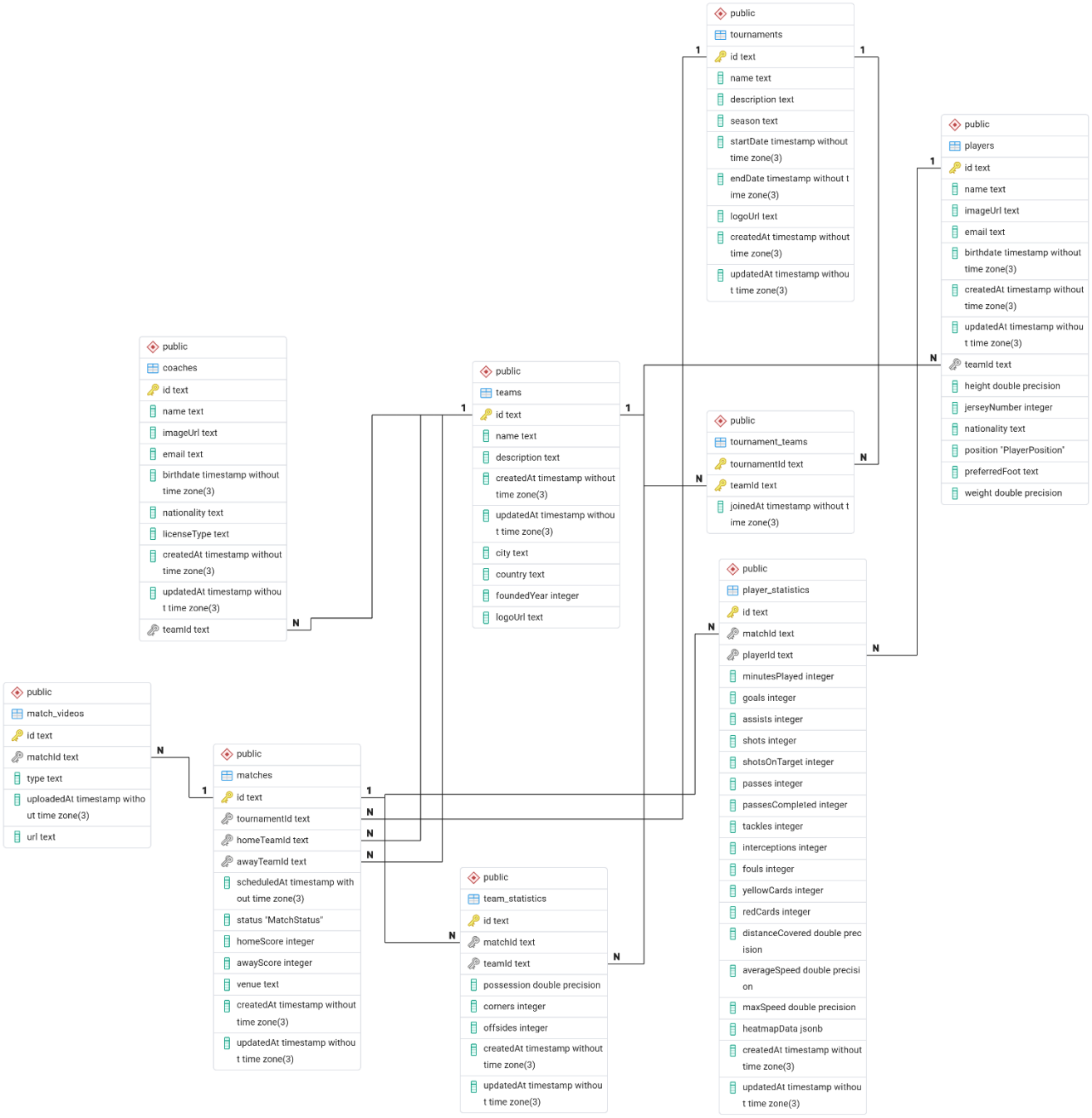


Рисунок 4.2 – ERD-модель розробленої БД системи

4.2.2 Створення бази даних для неструктурованих даних у вигляді результатів розпізнавання

MongoDB зберігає результати розпізнавання з компоненту по обробці відео у вигляді документів з гнучкою структурою. На відміну від PostgreSQL, тут немає фіксованої схеми, що дозволяє зберігати треки з різною кількістю об'єктів, але

треки зберігаються повноцінні на базі яких можна виконати розмітку відео у разі втрати. Також зберігання треків у MongoDB дозволяє не виконувати додаткового розпізнання відео у разі додавання нової метрики, оскільки ми зможемо використати інформацію з БД.

База даних зберігає такі колекції:

– колекція match_detections (Результати розпізнавання матчу) – ця колекція зберігає сирі треки, що ми отримали від компоненту по оброці відео на яких розмічена треки гравців, м’яча, тренера та ключових точок на полі. Приклад документу, що зберігається у цій колекції наведено у додатку А.

Аналізуючи документ, що зберігається у колекції match_detections можна виділити наступні атрибути:

– match_id – ідентифікатор матчу з PostgreSQL, тип String, створює логічний зв'язок з таблицею Match через поле detectionId;

– detection_id – унікальний ідентифікатор результату детекції, тип String, генерується сервісом object-detection;

– video_url – URL оригінального відео з Cloudinary для обробки, тип String;

– processed_video_url – URL анотованого відео після обробки, завантаженого до Cloudinary, тип String;

– fps – кількість кадрів за секунду у відео, тип Integer (наприклад, 30 або 24);

– total_frames – загальна кількість оброблених кадрів у відео, тип Integer;

– duration_seconds – тривалість відео в секундах, тип Integer, розраховується як total_frames / fps;

– created_at – дата та час створення запису, тип ISODate, генерується автоматично при збереженні;

– вкладений об'єкт stats – попередньо розраховані базові статистики розпізнавання:

– team_1_count – кількість гравців першої команди, тип Integer, визначається через підрахунок унікальних ID з team=1;

– team_2_count – кількість гравців другої команди, тип Integer, визначається через підрахунок унікальних ID з team=2;

– unique_players – загальна кількість унікальних гравців у матчі, тип Integer, сума team_1_count + team_2_count;

– ball_detections – кількість кадрів, у яких виявлено м'яч, тип Integer, використовується для оцінки якості детекції;

– вкладений об'єкт tracks – основні дані треків з координатами об'єктів на кожному кадрі:

– масив players – треки гравців, де кожен елемент є об'єктом з номером кадру як ключем, тип Array of Objects. Для кожного кадру зберігається об'єкт з наступними полями:

а) ключ (ID гравця, присвоєний алгоритмом ByteTrack, тип String);

б) bbox (координати bounding box у форматі [x1, y1, x2, y2] у пікселях, тип Array of Numbers);

в) team (номер команди – 1 або 2, визначений через алгоритм K-means, тип Integer);

г) team_color (домінантний колір форми команди у форматі [R, G, B], тип Array of Integers).

– масив referees – треки суддів, тип Array of Objects, опціональне поле. Структура аналогічна players, але без полів team та team_color, оскільки судді не належать до команд;

– масив ball – треки м'яча, тип Array of Objects. Кожен елемент містить номер кадру як ключ та поле bbox з координатами [x1, y1, x2, y2].

4.3 Компонент обробки відеозапису матчу

Компонент обробки відеозапису матчу є головним джерелом інформації про матч, який застосовує комп’ютерне бачення для розпізнання гравців, м’яча та поля у відеозаписі матчу. Перед цим компонентом стоїть задача перетворити сирий відеофайл у структуровані дані про рух об'єктів (треки), які надалі використовуються для розрахунку метрик.

4.3.1 Розробка діаграми послідовності дій компонента обробки відеозапису матча

Процес обробки відео включає детекцію об'єктів, трекінг, класифікацію за командами та збереження результатів. Для цього компонента створена діаграма послідовності дій яка зображена на рисунку 4.3.

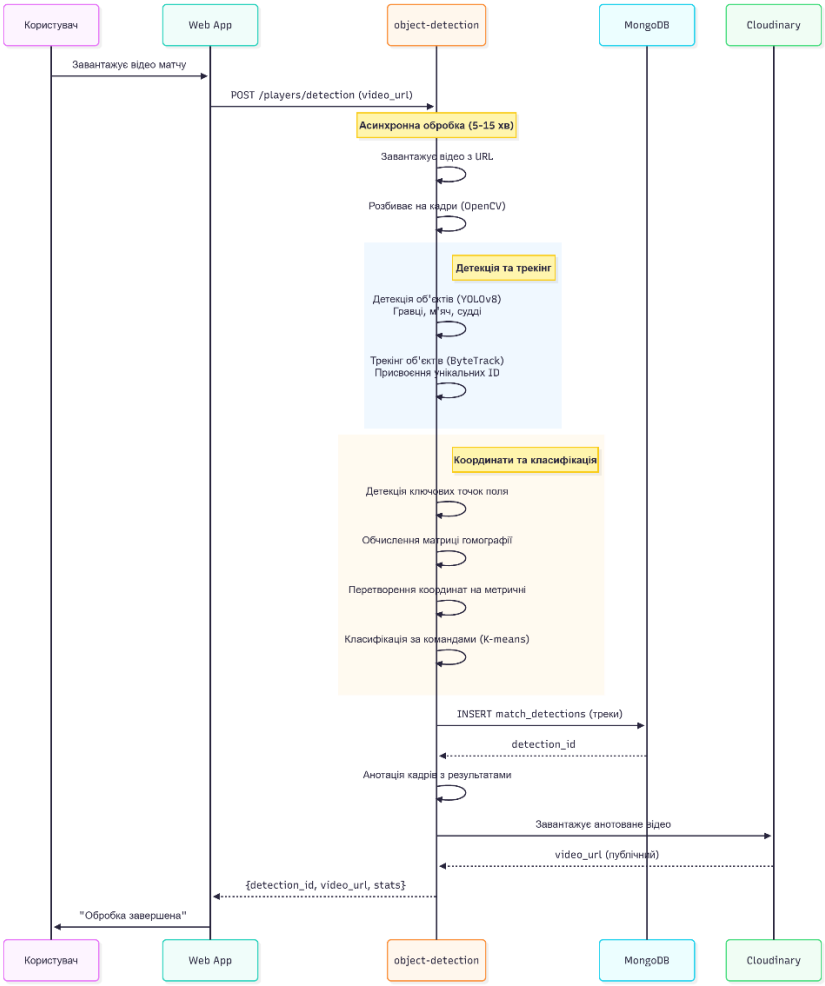
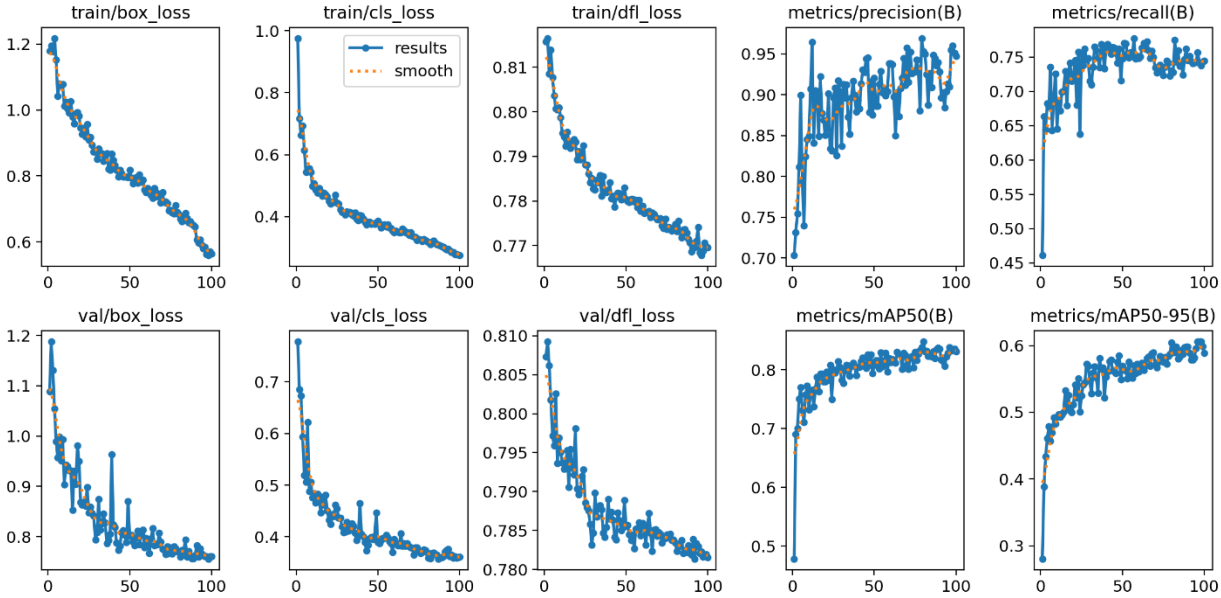


Рисунок 4.3 – Діаграма послідовності дій компонента обробки відеозапису

4.3.2 Реалізація детекції та трекінгу

На базі наведеної діаграми послідовності дій компонента обробки відеозапису матча реалізовано цей компонент з використанням бібліотеки Ultralytics для роботи з YOLOv8 та власної імплементації ByteTrack.

Для налаштування YOLOv8 було використано найбільшу модель для максимальної точності з використанням GPU як пристрій на якому буде працювати модель [20]. Саму ж модель було навчено на базі відкритого датасета [21] в якому знаходилось більше 800 фотографій з розміченими гравцями, м’ячом та суддями. Результат навчання моделі на датасеті наведено на рисунку 4.4.



Риуснок 4.4 – Результат навчання моделі YOLOv8 на датасеті з розміченими гравцями, м’ячом та суддями

Аналізуючи отримані графіки, можна сказати, що:

– mAP50 (Mean Average Precision at IoU=0.5) – досягнуто значення 0.831 (83.1%), що свідчить про високу точність детекції об'єктів. Цей показник означає, що модель правильно розпізнає гравців, м'яч та суддів у 83% випадків при порозі IoU 50%;

– mAP50-95 (Mean Average Precision at IoU=0.5:0.95) – фінальне значення 0.589 (58.9%) демонструє стабільну роботу моделі навіть при більш суворих порогах перетину bounding boxes;

– Precision (точність) – 0.947 (94.7%) на останній епосі, що означає мінімальну кількість хибнопозитивних детекцій;

– Recall (повнота) – 0.744 (74.4%), що вказує на здатність моделі виявити більшість об'єктів на кадрі.

– Функція втрат (loss) стабільно знижувалася протягом усіх 100 епох: train/box_loss знизилася з 1.18 до 0.56, train/cls_loss – з 0.98 до 0.28, що свідчить про успішне навчання без ознак перенавчання (overfitting);

– Валідаційні метрики (val/box_loss, val/cls_loss) демонстрували аналогічну тенденцію зниження, підтверджуючи узагальнюючу здатність моделі на нових даних;

Отримані результати підтверджують, що навчена модель YOLOv8x здатна з високою точністю розпізнавати гравців, м'яч та суддів у реальних умовах футбольних матчів, що є критично важливим для подальшого трекінгу та аналізу.

Наступним етапом є налаштування ByteTrack для забезпечення трекінгу об’єктів між кадрами за допомогою присвоєнню їм їх унікальних ідентифікатора. Параметри, що були застосовані під час конфігурації:

– track_thresh – цей параметр дозволяє фільтрувати випадкові детекції, зберігає лише впевнені об'єкти;

– track_buffer – цей параметр дозволяє відновити трек якщо гравець перекритий іншим гравцем до 1 секунди;

– match_thresh – цей параметр дозволяє встановити високий поріг для зв'язування детекцій для запобігання зміні ID при перетинах гравців.

Конфігурація ByteTrack наведено на лістингу 4.1.

Лістинг 4.1 – Код конфігурації ByteTrack

```
tracker = ByteTrack(
    track_thresh=0.6,       # Висока впевненість для початку треку
    track_buffer=30,        # Зберігати трек 30 кадрів після зникнення
    match_thresh=0.8,       # IoU поріг для зв'язування детекцій
    frame_rate=30           # Частота кадрів відео
```

Надалі реалізовано клас трекер, що застосовує на кадрах відео YOLO і за допомогою Bytetrack надає об’єктам унікальний ідентифікатор. Результат роботи цього класу є треки відео до яких додано до кожного з кадрів результати розпізнання. Код цього класу наведено на лістингу 4.2.

Лістинг 4.2 – Код класу трекера

```
class Tracker:
    def __init__(self, model_path: str, device: str = 'cpu'):
        self.model = YOLO(model_path)
        self.tracker = sv.ByteTrack()
        self.device = device
    def detect_frames(self, frames: list) -> list:
        return detect_frames_in_batches(
            model=self.model,
            frames=frames,
            batch_size=20,
            conf=0.1,
            device=self.device
        )
    def get_object_tracks(self, frames: list) -> dict:
        detections = self.detect_frames(frames)
        tracks = initialize_tracks_structure()
        for frame_num, detection in enumerate(detections):
            cls_names = detection.names
            cls_names_inv = get_inverted_class_names(cls_names)
            sv_detections = convert_to_supervision_detections(detection)
            sv_detections    =    normalize_goalkeeper_class(sv_detections,
cls_names_inv)
            detection_with_tracks                                          =
self.tracker.update_with_detections(sv_detections)
            add_frame_to_tracks(tracks)
            self._process_tracked_objects(
                detection_with_tracks=detection_with_tracks,
                cls_names_inv=cls_names_inv,
                tracks=tracks,
                frame_num=frame_num
            )
            self._process_ball_detections(
                sv_detections=sv_detections,
                cls_names_inv=cls_names_inv,
                tracks=tracks,
                frame_num=frame_num
            )
        return tracks
    def _process_tracked_objects(
        self,
        detection_with_tracks,
        cls_names_inv: dict,
        tracks: dict,
        frame_num: int
    ):
        for frame_detection in detection_with_tracks:
```

```
            bbox = frame_detection[0].tolist()
            class_id = frame_detection[3]
            track_id = int(frame_detection[4])
            if class_id == cls_names_inv['player']:
                add_player_track(tracks, frame_num, track_id, bbox)
            elif class_id == cls_names_inv['referee']:
                add_referee_track(tracks, frame_num, track_id, bbox)
    def _process_ball_detections(
        self,
        sv_detections,
        cls_names_inv: dict,
        tracks: dict,
        frame_num: int
    ):
        for frame_detection in sv_detections:
            bbox = frame_detection[0].tolist()
            class_id = frame_detection[3]
            if class_id == cls_names_inv['ball']:
                add_ball_track(tracks, frame_num, bbox)
```

Результат класу трекера, що наведено на лістингу 4.1 зображено на рисунку 4.5 після анотації. Розпізнання показало дуже хороший результат, де кожного з гравців було розпізнано, суддя був успішно виділений у окремий клас та добре розпізнається м’яч.

Рисунок 4.5 – Результат виконання розпізнання гравців, суддів та м’яча

4.3.3 Реалізація класифікації за командами

На базі отриманих результатів за допомогою розпізнання гравців на полі (див. рисунок 4.5) можна класифікувати гравців за командами. Для класифікації буде використовуватись колір джерсі граців, оскільки згідно з правилами футболу – їх колір має сильно відрізнятись, тому можна застосувати алгоритм домінантного кольору з використанням K-means кластеризацією.

Для отримання джерсі гравця ми беремо кожен з кадрів, отримуємо границі гравців і обробляємо кожного окремо шляхом обрізання нижньої частини кадру для уникнення впливу кольору газону на виділення кольору джерсі (див. рисунок 4.6).

Рисунок 4.6 – Процес виділення кольору форми гравців

Наступним кроком це є виділення колір джерсі кожного гравця окремо. Це виконується за допомогою застосування K-Means кластеризації з двома кластерами, що розділяє пікселі на два кластери: фон (поле, трибуни) та форма гравця (див. рисунок 4.7). Лістинг функції, що виконує кластеризацію наведено у лістингу 4.3.

Лістинг 4.3 – Код функції для отримання кольору джерсі гравця на кадрі

```
def get_clustering_model(self, image):
    # Перетворюємо зображення у двовимірний масив пікселів
    pixels = image.reshape(-1, 3)

    # K-Means з k=2 (форма + фон)
    kmeans = KMeans(n_clusters=2, init="k-means++", n_init=1)
    kmeans.fit(pixels)

    return kmeans
```

Для визначення, який з двох кластерів належить формі гравця, аналізуються кутові пікселі зображення. Оскільки кути зазвичай містять фон, кластер, що найчастіше зустрічається в кутах, вважається фоном. У лістингу 4.4 наведено код функції, що застосовується для визначення кластеру форми гравця. Результат наведено на рисунку 4.7

Лістинг 4.4 – Код функції для визначення кластеру форми гравця

```
def get_clustering_model(self, image):
labels = kmeans.labels_
clustered_image = labels.reshape(top_half_image.shape[0],
top_half_image.shape[1])
corner_clusters = [
    clustered_image[0, 0],       # Лівий верхній кут
    clustered_image[0, -1],      # Правий верхній кут
    clustered_image[-1, 0],      # Лівий нижній кут
    clustered_image[-1, -1],     # Правий нижній кут
]
non_player_cluster = max(set(corner_clusters), key=corner_clusters.count)
player_cluster = 1 - non_player_cluster
player_color = kmeans.cluster_centers_[player_cluster]
```

Рисунок 4.7 – Процес визначення кольору форми методом K-Means

Після обробки кожного гравця формується масив кольорів форм оскільки колір форми залежить від світла і багато інших факторів, то спочатку необхідно отримати середній колір між ними. Як останній етап виконується глобальна кластеризація масиву у RGB-просторі на дві групи, що відповідають двом командам (див. рисунок 4.8). Процес глобальної кластеризації наведено на лістингу 4.5.

Лістинг 4.5 – Код функції для глобальної кластеризації у RGB-просторі на дві групи

```
def assign_team_color(self, frame, player_detections):
    ...
    # Глобальна кластеризація кольорів на 2 команди
    kmeans = KMeans(n_clusters=2, init="k-means++", n_init=10)
    kmeans.fit(player_colors)

    self.kmeans = kmeans

    # Зберігаємо кольори команд (центроїди кластерів)
    self.team_colors[1] = kmeans.cluster_centers_[0]
    self.team_colors[2] = kmeans.cluster_centers_[1]
```

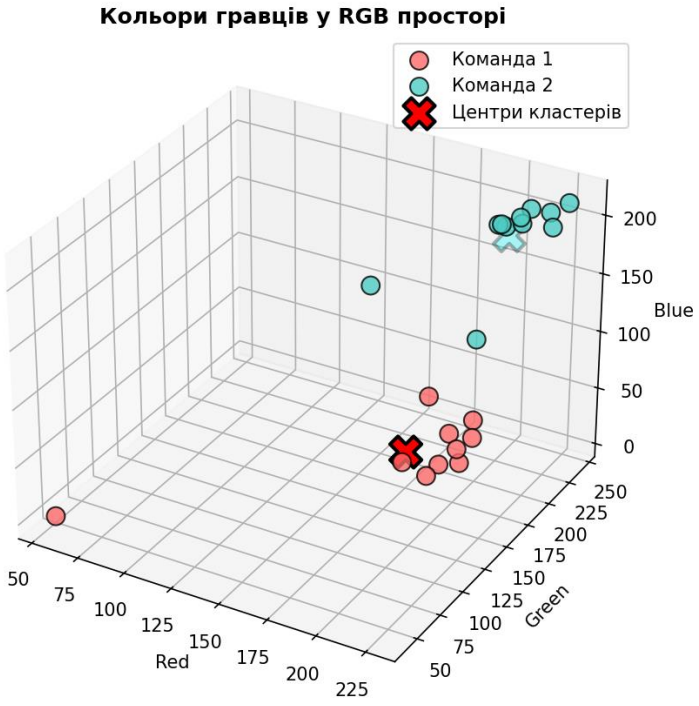


Рисунок 4.8 – Процес глобальної кластеризації кольорів форм методом K-Means

Для визначення команди конкретного гравця його колір форми порівнюється з центроїдами команд за допомогою методу predict який надається K-means. Процес присвоєння кольору до кожного гравця зображено на лістингу 4.6 і результат на рисунку 4.9.

Лістинг 4.6 – Код функції для присвоєння гравцю кольору його команди

```
def get_player_team(self, frame, player_bbox, player_id):
    # Кешування для оптимізації (не переобчислювати для того ж гравця)
    if player_id in self.player_team_dict:
        return self.player_team_dict[player_id]

    # Визначаємо колір форми гравця
    player_color = self.get_player_color(frame, player_bbox)

    # Класифікуємо за найближчим центроїдом команди
    team_id = self.kmeans.predict(player_color.reshape(1, -1))[0]
    team_id += 1  # Команди нумеруються з 1

    # Зберігаємо результат у кеш
    self.player_team_dict[player_id] = team_id

    return team_id
```

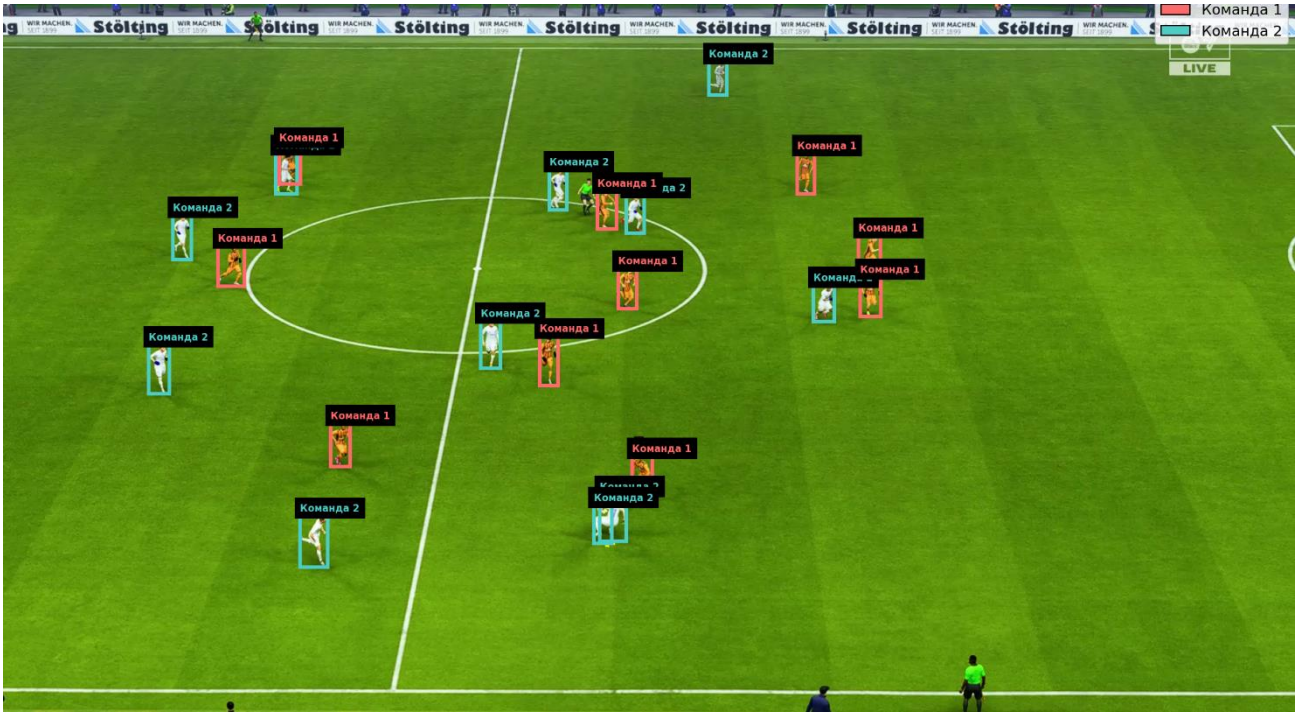


Рисунок 4.9 – Результат розпізнання кольору форми гравців

4.3.4 Реалізація детекції ключових точок поля

Для побудови тактичних діаграм у вигляді 2D карт та перетворення координат з пікселів у метри необхідно визначити геометрію футбольного поля. Згідно з проведеним дослідженням було обрано метод детекції з використанням YOLOv8 для розпізнавання ключових точок на основі спеціального відкритого датасету [22], в якому розмічені ключові точки футбольного поля. Використання YOLOv8 дозволяє автоматизувати процес знаходження розмітки на полі без ручного втручання. Координати ключових точок на реальному футбольному полі було розраховано за допомогою скрипту, що використовує обчислення, що наведені у дослідженні для розрахунку координат.

Кадри відеоматчу передаються до функції detect_frames, яка запускає розпізнавання на кожному з кадрів за допомогою навченої моделі YOLOv8. Усього на полі визначається 32 ключові точки (кути поля, лінії штрафних зон, центральне коло тощо). Функція виконує первинну детекцію з низьким порогом впевненості (0.3) та додатково фільтрує ключові точки з точністю менше 50% для підвищення надійності подальших обчислень гомографії. Функція наведено у лістингу 4.7.

Лістинг 4.7 – Код функції для розпізнавання ключових точок на полі

```
    def detect_frames(self, frames, confidence=0.3):
        detection_results = []
        for frame in frames:
            result = self.model(frame, conf=confidence,
verbose=False, device=self.device)[0]
            keypoints = sv.KeyPoints.from_ultralytics(result)
            if len(keypoints.xy) > 0 and len(keypoints.confidence) >
0:
                filter_mask = keypoints.confidence[0] > 0.5
                filtered_keypoints_xy = keypoints.xy[0][filter_mask]
                filtered_keypoints =
sv.KeyPoints(xy=filtered_keypoints_xy[np.newaxis, ...])
            else:
                filter_mask = np.array([])
                filtered_keypoints = sv.KeyPoints(xy=np.empty((1, 0,
2)))
            detection_results.append({
                'keypoints': keypoints,
                'filtered_keypoints': filtered_keypoints,
                'filtered_indices': filter_mask
            })
        return detection_results
```

Наступним кроком є реалізація класу ViewTransformer для обчислення гомографії – математичного перетворення, яке дозволяє перетворити координати гравців з відеокадру (2D проекція з перспективою) у реальні метричні координати на полі (вид зверху).

Для обчислення гомографії використовуються дві системи координат:

– target (ціль) – розпізнанні ключові точки на кадрі у пікселях – отримані від YOLOv8;

– source (джерело) – координати ключових точок у сантиметрах, що відповідають реальному полю.

Ці координати ключових точок, що відповідають реальному полю є незмінними та використовуються як еталонна система для всіх відеоматчів незалежно від ракурсу камери. Клас містить 32 ключові точки поля у форматі (x, y), де координати вказані у сантиметрах для точності обчислень.

Для обчислення матриці гомографії використовується бібліотека cv2, а саме її функція findHomography, що знаходить оптимальну матрицю методом найменших квадратів на основі відповідності точок поля (source) та їх пікселів на кадрі (target). Основними причинами застосування гомографії є:

– преспективні спотворення які відбуваються через те, що камера знаходиться під котом і в результаті елементи поля можуть здаватись більшими

– точні метричні розрахунки, де нам необхідно знати реальну швидкість гравця у км/год та відстань у метрах яку від пробіг

– отримання тактичних діаграм, які зображуються у вигляді 2D поля з видом зверху

Реалізація класу, який інкапсулює обчислення та застосування матриці гомографії зображено на лістингу 4.8.

Лістинг 4.8 – Код класу для обчислення гомографії

```
class ViewTransformer:
    def __init__(
            self,
            source: npt.NDArray[np.float32],
            target: npt.NDArray[np.float32]
    ) -> None:
        source = source.astype(np.float32)
```

```
        target = target.astype(np.float32)
        self.m, _ = cv2.findHomography(source, target)
        if self.m is None:
            raise ValueError(
                "Homography matrix could not be calculated. "
                "Possible causes: insufficient points (need at least
4), or points are collinear."
            )
    def transform_points(
            self,
            points: npt.NDArray[np.float32]
    ) -> npt.NDArray[np.float32]:
        reshaped_points = points.reshape(-1, 1, 2).astype(np.float32)
        transformed_points                                          =
cv2.perspectiveTransform(reshaped_points, self.m)
        return transformed_points.reshape(-1, 2).astype(np.float32)
```

Результатом виконаних перетворень і обчислень ми можемо створити 2D представлення футбольного поля, де зобразимо позиції гравців. На рисунку 4.10 зображено кадр з відеоряду гри EA SPORTS FC™ 26 на якому видно гравців і їх позиції, а на рисунку 4.11 зображено представлення результатів роботи алгоритму Гоморграфії у вигляді 2D представлення футбольного поля з відображення позицій гравців і їх командної належності.

Рисунок 4.10 – Кадр з відеоряду гри EA SPORTS FC™ 26 на якому видно гравців і їх позиції

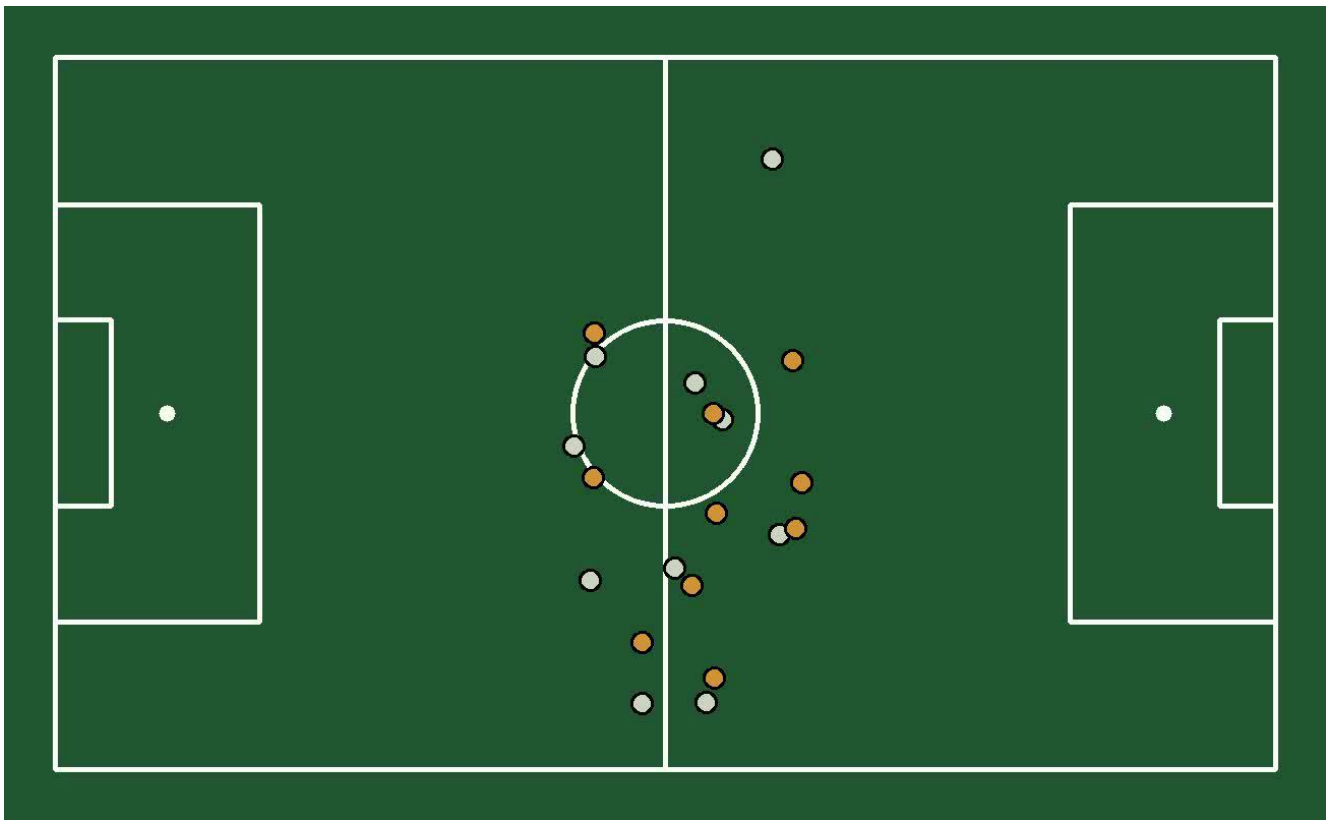


Рисунок 4.11 – Представлення результату роботи алгоритму Гомографії

Як видно на рисунку 4.11 позиції гравців можуть мати похибку яка зменшується при наявності якомога більше ключових точок на відеокадрі, але наявна точність є прийнятною для аналізу. Найменша кількість точок яка потрібна для цього процесу – чотири точки.

4.4 Компонент розрахунку метрик

Компонент розрахунку метрик відповідає за перетворення сирих треків з MongoDB у структуровані метрики продуктивності гравців та команд, які зберігаються у PostgreSQL. В цьому компонентів розраховуються всі метрики, окрім індексу ефективності гравця (PPI), оскільки цей індекс має в собі динамічні коефіцієнти, тому обрахунок виконується на етапі коли користувач запитує дані де він використовується.

4.4.1 Розробка діаграми послідовності дій компонента розрахунку метрик

Процес розрахунку метрик отримує сирі треки з MongoDB, виконує математичні обчислення та зберігає структуровані результати в PostgreSQL. Для цього компонента створена діаграма послідовності дій яка зображена на рисунку 4.12.

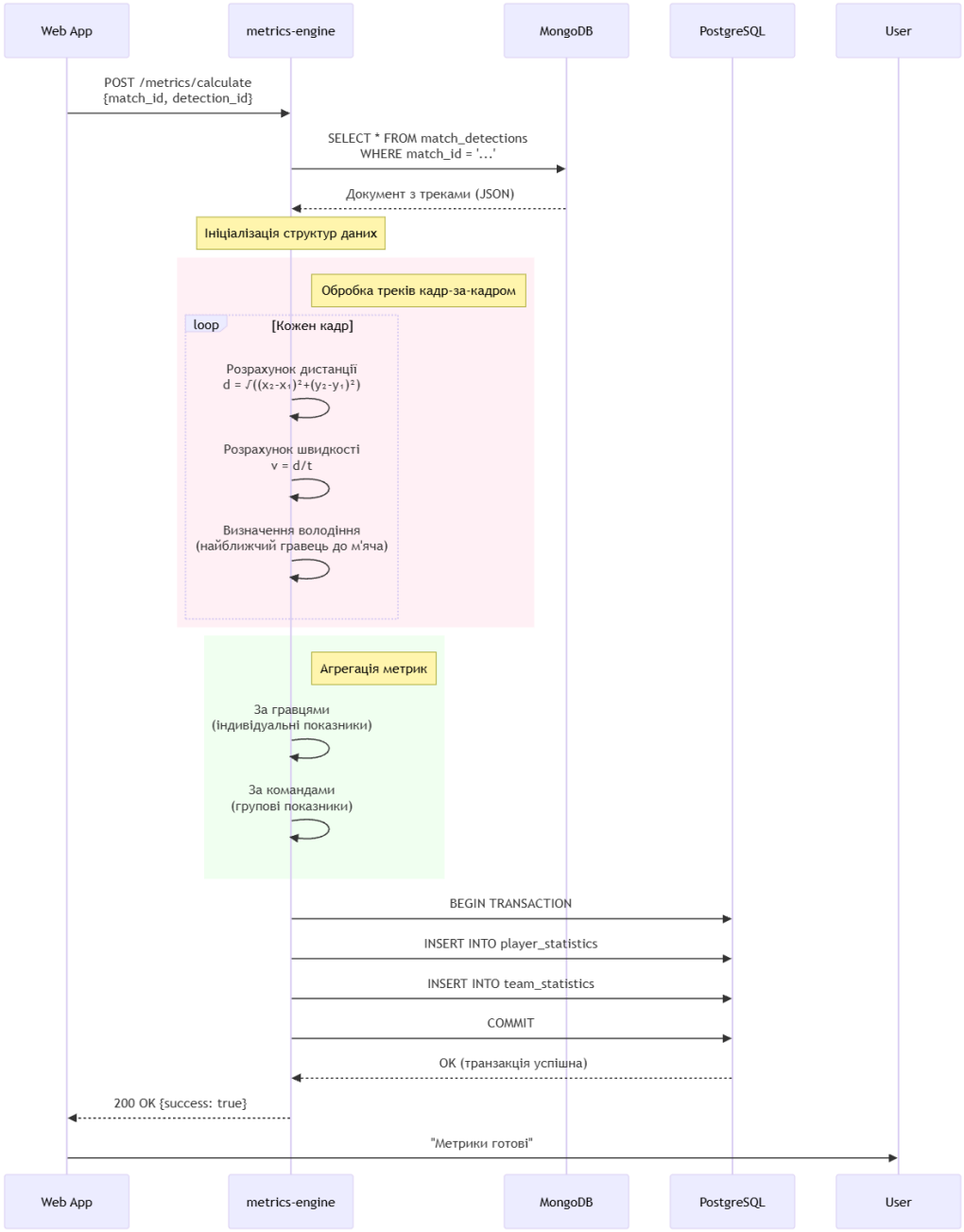


Рисунок 4.12 – Діаграми послідовності дій компонента розрахунку метрик

4.4.2 Реалізація розрахунку метрик швидкості і дистанції гравця

Для розрахунку дистанції та швидкості гравця необхідно використовувати реальну відстану, що пробіг гравець яку було отримано під час перетворення координат через гомографію. Формули для розрахунку метрик будуть застосовані згідно з дослідженнями, а саме:

– розрахунок Евклідової відстані – між двома послідовними кадрами обчислюється відстань за формулою $d = \sqrt{((x_2 - x_1)^2 + (y_2 - y_1)^2)}$;

– обчислення швидкості – на основі відстані та часового інтервалу між кадрами: $v = d/t$, конвертується з м/с у км/год.

Розрахунку метрик швидкості і дистанції гравця зображено у лістингу 4.9.

Лістинг 4.9 – Код функції для метрик швидкості і дистанції гравця

```
function calculatePlayerMetrics(
  tracks: PlayerTrack[],
  fps: number,
  homography: number[][]
): Map<string, PlayerMetrics> {
  const playerMetrics = new Map<string, PlayerMetrics>();

  // Покадрова обробка (від кадру 1, оскільки потрібен попередній кадр)
  for (let i = 1; i < tracks.length; i++) {
    const prevFrame = tracks[i - 1];
    const currFrame = tracks[i];

    // Обробка кожного гравця у поточному кадрі
    for (const playerId in currFrame) {
      // Пропустити, якщо гравця не було у попередньому кадрі (щойно
з'явився)
      if (!prevFrame[playerId]) continue;

      // 1. Отримати центри bounding box у пікселях
      const prevCenter = getBboxCenter(prevFrame[playerId].bbox);
      const currCenter = getBboxCenter(currFrame[playerId].bbox);

      // 2. Перетворити координати у метри через гомографію
      const prevMetric = transformPoint(prevCenter, homography);
      const currMetric = transformPoint(currCenter, homography);

      // 3. Обчислити Евклідову відстань у метрах
      const distance = Math.sqrt(
        Math.pow(currMetric[0] - prevMetric[0], 2) +
        Math.pow(currMetric[1] - prevMetric[1], 2)
      );
```

```
      // 4. Розрахувати швидкість: v = d/t, конвертувати у км/год
      const timeInterval = 1 / fps;  // Час між кадрами у секундах (30fps
→ 0.033c)
      const speed = (distance / timeInterval) * 3.6;   // м/с → км/год
(множимо на 3.6)

      // 5. Оновити або ініціалізувати метрики гравця
      if (!playerMetrics.has(playerId)) {
        playerMetrics.set(playerId, {
          distanceCovered: 0,
          speeds: [],
          positions: []
        });
      }

      const metrics = playerMetrics.get(playerId)!;
      metrics.distanceCovered += distance;               // Акумулювати
загальну дистанцію
      metrics.speeds.push(filterAnomalousSpeed(speed));    //    Зберегти
швидкість з фільтрацією
      metrics.positions.push(currMetric);            // Зберегти позицію
для heat map
    }
  }

  return playerMetrics;
}
```

Після обробки всіх кадрів обчислюються підсумкові показники для кожного гравців. Код функції для агрегації метрик наведено на лістингу 4.10.

Лістинг 4.10 – Код функції для метрик швидкості і дистанції гравця

```
function aggregatePlayerStatistics(metrics: PlayerMetrics) {
  const validSpeeds = metrics.speeds.filter(s => s > 0);

  return {
    totalDistance: Math.round(metrics.distanceCovered),              //
Загальна дистанція (м)
    averageSpeed: calculateAverage(validSpeeds),                     //
Середня швидкість (км/год)
    maxSpeed: Math.max(...validSpeeds),                              //
Максимальна швидкість (км/год)
    sprintCount: validSpeeds.filter(s => s > 20).length,             //
Кількість спринтів (>20 км/год)
    heatmapData: metrics.positions                                 // Дані
для теплової карти
```

4.4.3 Реалізація розрахунку метрик володіння м’ячом

Володіння м'ячом визначається як відсоток часу, коли команда контролює м'яч. Реалізовано алгоритм на основі пошуку найближчого гравця до м'яча у кожному кадрі. Алгоритм розрахунку:

– для кожного кадру визначається позиція м'яча у метричних координатах;

– виконується пошук найближчого гравця за допомогою обчислення Евклідової відстані від кожного гравця до м'яча;

– перевірка порогу контролю, де ми перевіряємо якщо найближчий гравець знаходиться ближче 2 метрів – м'яч під контролем його команди;

– підрахунок кадрів, де ми сумуємо кількість кадрів володіння для кожної команди;

– конвертація у відсотки, де виконується обчислення відсотку від загальної кількості кадрів з м’ячом.

Код функції для розрахунку метрик володіння м’ячом наведено на лістингу 4.11.

Лістинг 4.11 – Код функції для розрахунку метрик володіння м’ячом

```
function calculatePossession(
  playerTracks: PlayerTrack[],
  ballTracks: any[],
  homography: number[][]
): PossessionResult {
  const possession = { team1: 0, team2: 0, neutral: 0 };

  // Обробка кожного кадру
  for (let i = 0; i < ballTracks.length; i++) {
    const ballFrame = ballTracks[i];
    const playerFrame = playerTracks[i];

    // Пропустити кадри без м'яча (м'яч не детектовано або вилетів за
поле)
    if (!ballFrame || Object.keys(ballFrame).length === 0) {
      possession.neutral++;
      continue;
```

```
    }

    // Отримати позицію м'яча у метрах
    const ballBbox = Object.values(ballFrame)[0].bbox;
    const ballCenter = getBboxCenter(ballBbox);
    const ballPos = transformPoint(ballCenter, homography);

    // Пошук найближчого гравця серед усіх гравців на полі
    let minDistance = Infinity;
    let closestTeam = 0;

    for (const playerId in playerFrame) {
      const playerBbox = playerFrame[playerId].bbox;
      const playerCenter = getBboxCenter(playerBbox);
      const playerPos = transformPoint(playerCenter, homography);

      // Обчислити відстань між гравцем та м'ячем у метрах
      const distance = Math.sqrt(
        Math.pow(playerPos[0] - ballPos[0], 2) +
        Math.pow(playerPos[1] - ballPos[1], 2)
      );

      // Оновити найближчого гравця
      if (distance < minDistance) {
        minDistance = distance;
        closestTeam = playerFrame[playerId].team;
      }
    }

    // Поріг контролю: гравець повинен бути ближче 2 метрів до м'яча
    // (середня довжина ноги + крок футболіста)
    const CONTROL_THRESHOLD = 2.0;  // метрів

    if (minDistance < CONTROL_THRESHOLD) {
      if (closestTeam === 1) {
        possession.team1++;
      } else if (closestTeam === 2) {
        possession.team2++;
      }
    } else {
      // М'яч у повітрі або ніхто не контролює
      possession.neutral++;
    }
  }

  // Конвертувати у відсотки від загальної кількості кадрів
  const totalFrames = ballTracks.length;

  return {
    team1: Math.round((possession.team1 / totalFrames) * 100 * 10) /
10,  // 1 знак після коми
```

```
    team2: Math.round((possession.team2 / totalFrames) * 100 * 10) /
10,
    neutralFrames: possession.neutral
  };
}
```

4.5 Компонент метаданих

Компонент метаданих відповідає за основну комунікацію з користувачем, а саме цей компонент надає інформацію, що зберігається у базі даних Postgresql про гравців, команди, матчів і їх статистики. Також цей компонент відповідає за розрахунок PPI згідно з проведеним дослідженням. Обрахунок виконується кожного разу коли виконується запит за статистикою гравця, оскільки цей розрахунок базується на динамічних коефіцієнтів, то результат обрахунку не зберігається у базі даних.

Комунікація між веб-додатком яким користується користувач і цим компонентом виконується за допомогою технології Graphql, що дозволяє оптимізувати запит шляхом забезпечення у відповіді лише тих даних, що наразі потрібні. Основні сутності на які створено GraphQL схеми:

- MatchEntity – сутність матчу, що пов'язує команди, відео та статистику гравців;
- TeamEntity – сутність команди з гравцями та тренерами;
- PlayerEntity – сутність гравця з персональною статистикою;
- PlayerStatisticEntity – статистика гравця за конкретний матч;
- TeamStatisticEntity – статистика гравця за конкретний матч.

Взаємодія з цими сутностями відбувається шляхом створення резолвера з описанням методів взаємодії з ними їх відношень між собою. На рисунку 4.13 зображено схему, що представляє зав’язки між сутностями GraphQL. Код резолвера, що відповідає за обробку запитів за командами зображено у лістингу 4.11.

Лістинг 4.11 – Код резолвера, що відповідає за обробку запитів за командами

```
@Resolver(() => MatchEntity)
export class MatchResolver {
  constructor(
    private readonly matchService: MatchService,
    private readonly teamService: TeamService,
    private readonly matchVideoService: MatchVideoService,
    private readonly playerStatisticService: PlayerStatisticService
  ) {}

  @Query(() => [MatchEntity], { name: 'matches' })
  findAll(): Promise<MatchEntity[]> {
    return this.matchService.findAll()
  }

  @Query(() => MatchEntity, { name: 'match', nullable: true })
  findOne(@Args('id', { type: () => ID }) id: string): Promise<MatchEntity
| null> {
    return this.matchService.findOne(id)
  }

  @ResolveField(() => [TeamEntity], { description: 'Команди матчу' })
  teams(@Parent() match: MatchEntity): Promise<TeamEntity[]> {
    return this.teamService.findAll({
      where: {
        OR: [{ id: match.homeTeamId }, { id: match.awayTeamId }],
      },
    })
  }

  @ResolveField(() => [MatchVideoEntity], { description: 'Відео матчу' })
  videos(
    @Parent() match: MatchEntity,
    @Args('types', { type: () => [MatchVideoType], nullable: true })
    types?: MatchVideoType[]
  ): Promise<MatchVideoEntity[]> {
    return this.matchVideoService.findByMatchId(match.id, types)
  }
}
```

Ключовою особливістю є використання декоратора ResolveField, що дозволяє визначити логіку завантаження пов'язаних даних. Це забезпечує ліниве завантаження (lazy loading) – пов'язані дані завантажуються лише коли клієнт явно їх запитує.

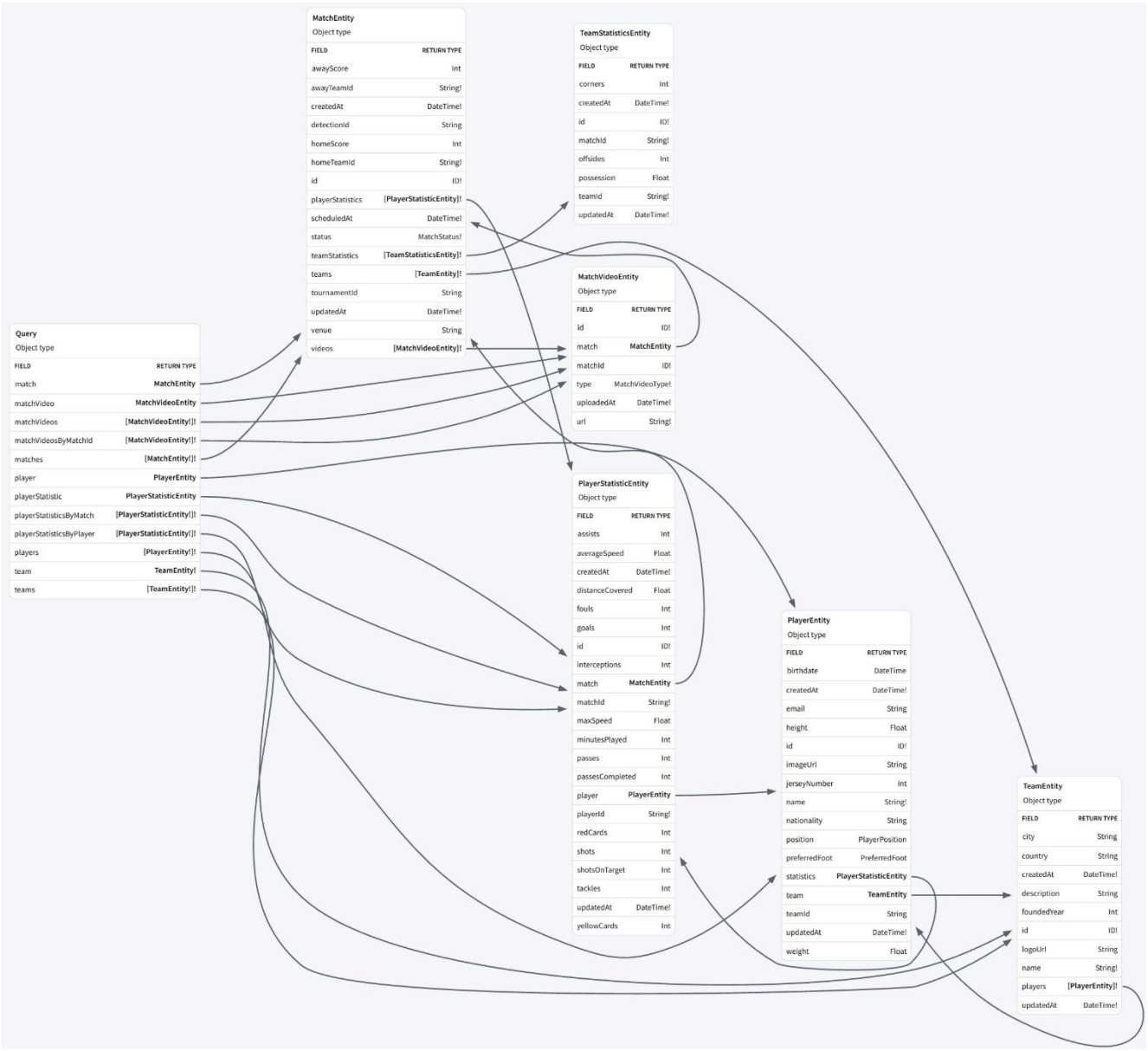


Рисунок 4.13 – Візуалізація схеми GraphQL

4.6 Компонент інтерфейсу взаємодії з системою

Інтерфейс взаємодії з системою представляє з себе веб-додаток за допомогою якого користувач може переглядати, редагувати наявну інформацію та додавати нову. Для забезпечення отримання рекомендацій до веб-додатку було інтегровано за допомогою серверних технологій комунікацію з LLM через її публічне API. Цей веб-додаток реалізовано на базі фреймворку Next.js, що дозволяє генерувати статичні сторінки з інформацією які з часов автоматично перестворюються, що дозволяє зменшити навантаження на компонент метаданих і досягти максимальної швидкості відповіді користувачу на запит за сторінкою. Для пришвидшення створення візуальної частини веб-додатку було використано технологію Tailwindcss для написання стилів для компонентів та бібліотеку компонентів Shadcn в якій надається список вже стилізованих за допомогою технології Tailwindcss компонентів. Для комунікації з компонентом метаданих використовується технологія Apollo Client (Graphql), що дозволяє досягти отримання схем даних і типізацію відповіді на запит за даними.

На головній сторінці веб-додаток представлено всі матчі, що створені у системі та топ найкращих гравців у яких значення PPI є найбільшим. Головна сторінка наведена на рисунку 4.14.

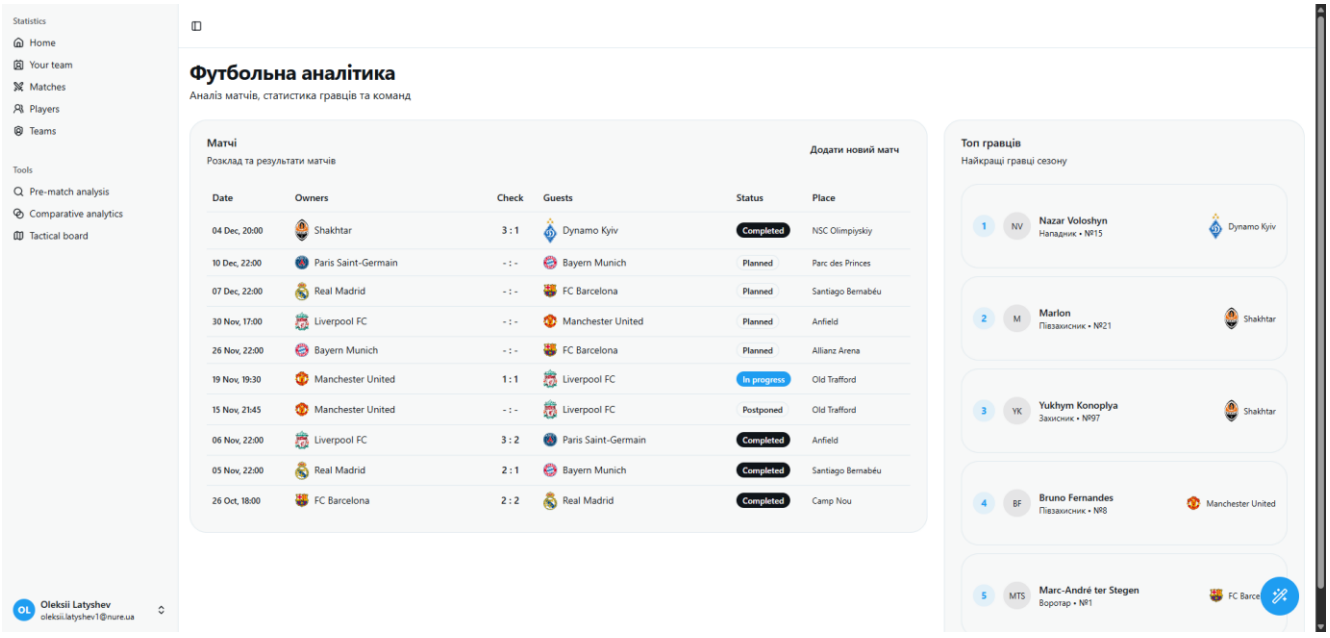


Рисунок 4.14 – Головна сторінка веб-додатку

4.6.1 Взаємодія користувача з матчами

Реалізовано сторінку матчу, де користувач може переглянути інформацію про матч, де якщо матч вже було завершено – можна також побачити відеозапис матчу, відеозапис після розпізнання гравців та 2D представлення поля матчу. На цій сторінці також представлено велику кількість метрик та графіків повязаних з цим матчів, а саме користувач може переглянути статистику команди, де відображаються метрики повязані лише з командою і графік де порівнюються ключові дії між цими двома командами (див. рисунок 4.15).

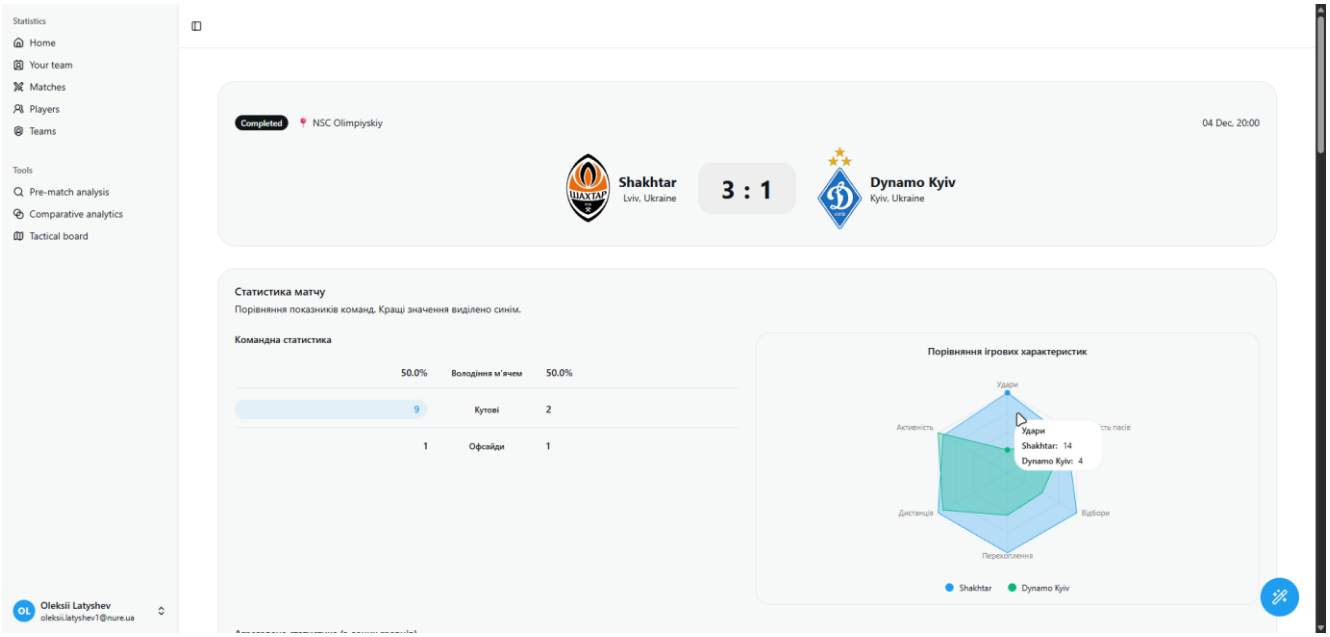


Рисунок 4.15 – Частина сторінки про матч з інформацією про команди

Нижче на сторінці з інформацією про матч користувач може переглянути агреговану статистику яка отримується на базі статистики гравців у цьому матчі. Зображення цієї статистики відбувається у вигляді графіку, де порівнюється атакуючі та оборонні дії команд та у вигляді списку з середніми значеннями метриків команд, де синім кольором виділена метрика, що є кращою ніж у опонента. Ця статистика дозволяє побачити наскільки фізичні можливості гравців мали перевагу над опонентом чи причина поразки було у тактиці (див. рисунок 4.16).

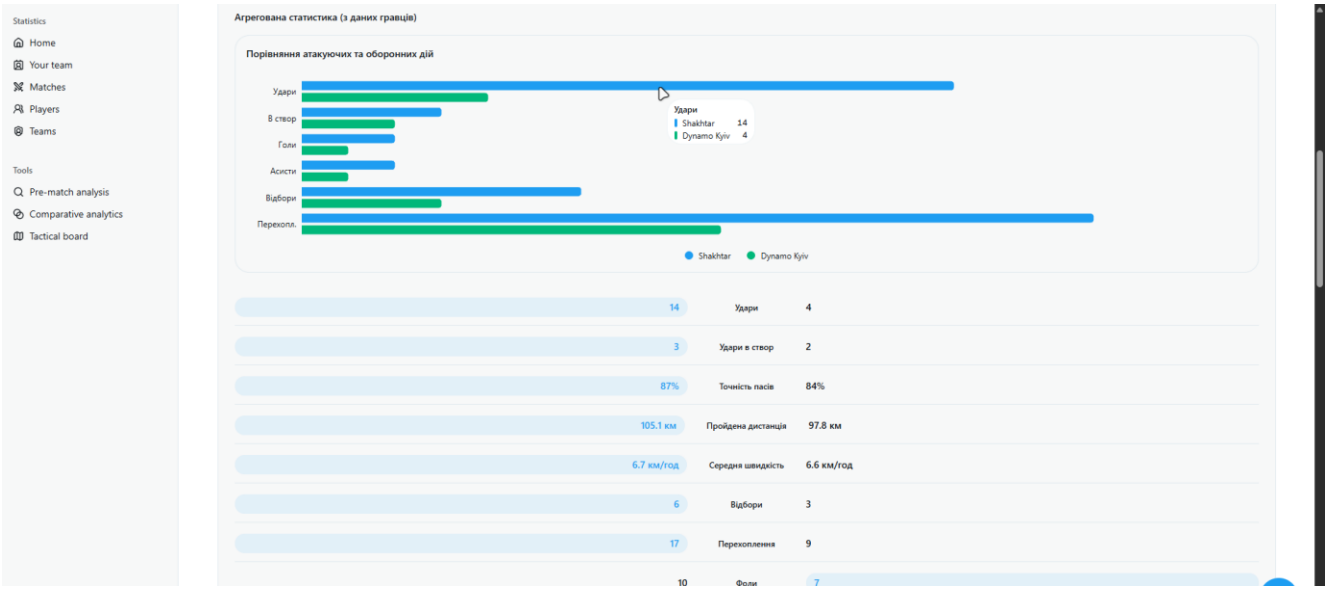


Рисунок 4.16 – Агрегована статистики матчу на базі метрик гравців за цей матч

Нижче наводиться статистика відносно самих гравців у цьому матчі, де за допомогою графіку відображаються топ гравців за певним показником який можна обрати справа у таблиці. Знизу ж зображена статистика гравців у вигляді таблиці, де можна обрати за допомогою вкладок гравців якої команди переглядати. У таблиці наведені основні метрики які необхідні для первинного аналізу та обрахований коефіцієнт ефективності гравця (PPI) згідно з виконаним дослідженням. На рисунку 4.17 зображено статистику гравців, де як топ гравців за показником було обрано показник “паси”.

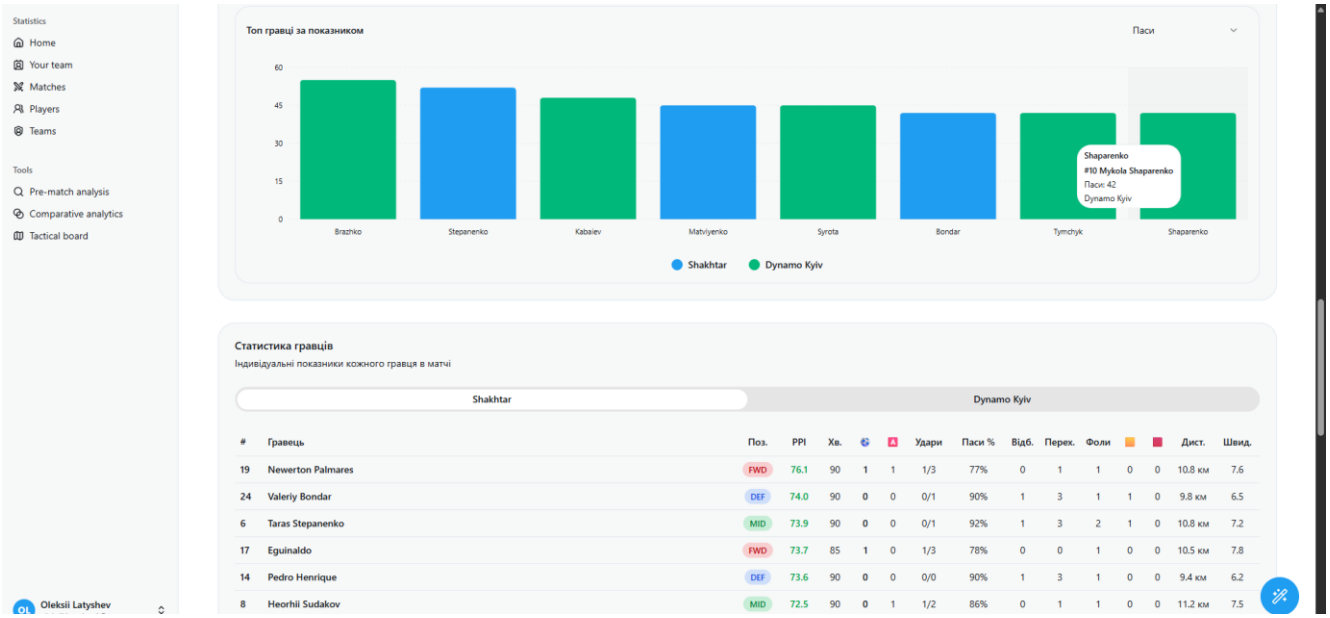


Рисунок 4.17 – Статистика гравців за матч

У самому низу сторінки про матч користувач може побачити відеозапис матчу та його оброблені версії між якими він може переключатись за допомогою меню справа зверху. На рисунку 4.18 наведено нижня частина сторінки з відео плеєром.

Рисунок 4.18 – Відео плеєр з записом матча

4.6.2 Процес додавання нового матча

Додавання нового матча відбувається за допомогою спеціальної кнопки “Додати новий матч”, що знаходиться зверху справа таблиці на головній сторінці (див. рисунок 4.14). Після натискання на цю кнопку користувача буде перенаправлено на сторінку з формою, яка складається з кількох кроків. На першому кроці користувачеві буде запропоновано ввести базову інформацію про цей матч, де потрібно вибрати команду, що грає на своєму стадіоні та команду, що приїхала грати, дату, стадіон та статус матча. Заповнена сторінка створення матча зображена на рисунку 4.19.

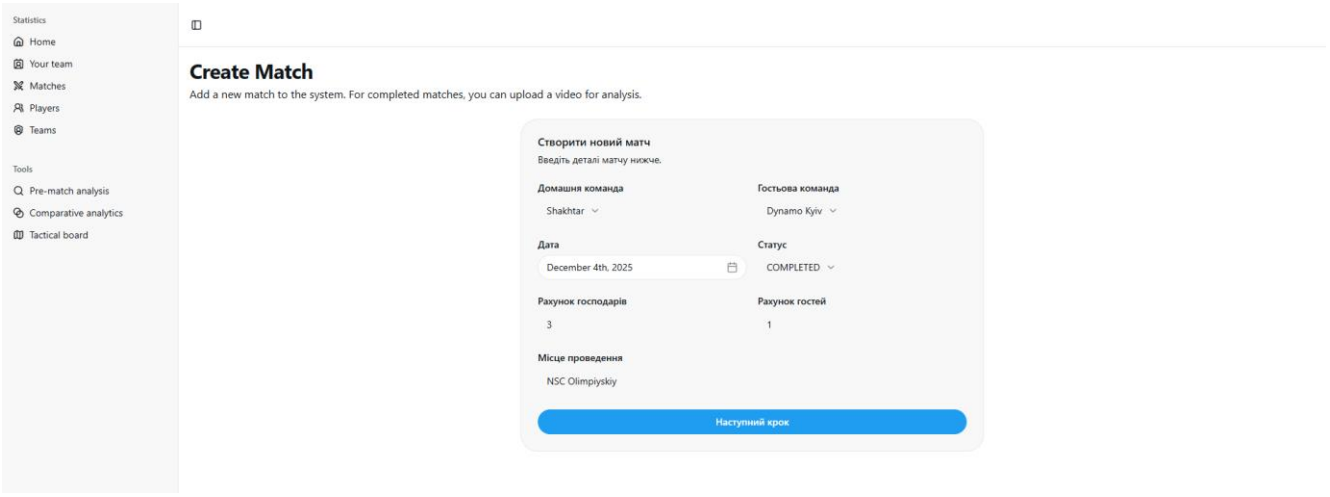


Рисунок 4.19 – Заповнена сторінка створення матча

У випадку, якщо статус матча обрано як “Завершено”, то користувачеві після натискання кнопки “Наступний крок” відкриється наступна форма в якій буде запропоновано завантажити відео результату матча яке система зможе обробити. Після завантаження відео буде у форму буде розпочато процес завантаження відео на сервіс Cloudinary на якому зберігаються всі медіа файли повязанні з системою – результатом же буде посилання на це відео яке наступним кроком буде відправлено до компонента по обробці відео. Сторінку з формою завантаження відео зображено на рисунку 4.20.

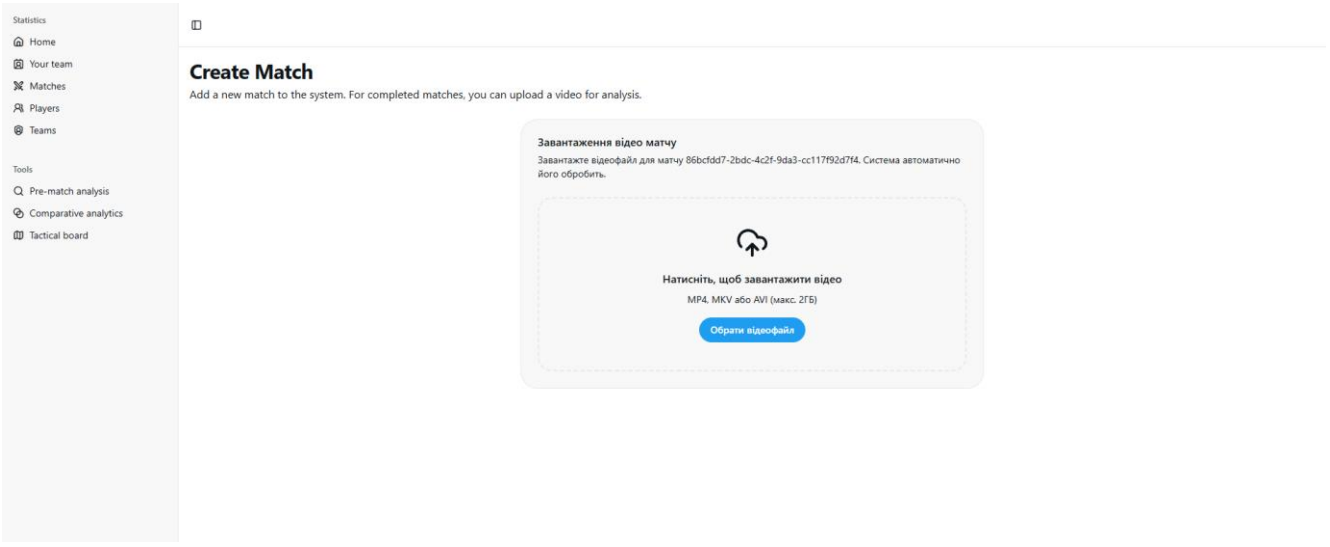


Рисунок 4.20 – Сторінка завантаження відео матча

4.6.3 Генерація рекомендацій за допомогою LLM

Процес генерації рекомендацій використовує модель, що була виявлена як раціональна для цієї дії згідно з виконаного дослідження у вигляді GPT-4o-mini яка використовує інформацію про матч, що надає компонент метаданих відповідно контексту який визначається на базі відкритої сторінки. Надалі інформація з контексту використовується для генерації інтелектуальних порад за допомогою вище названої LLM. Даний процес відображено за допомогою sequence діаграми яка зображена на рисунку 4.21.

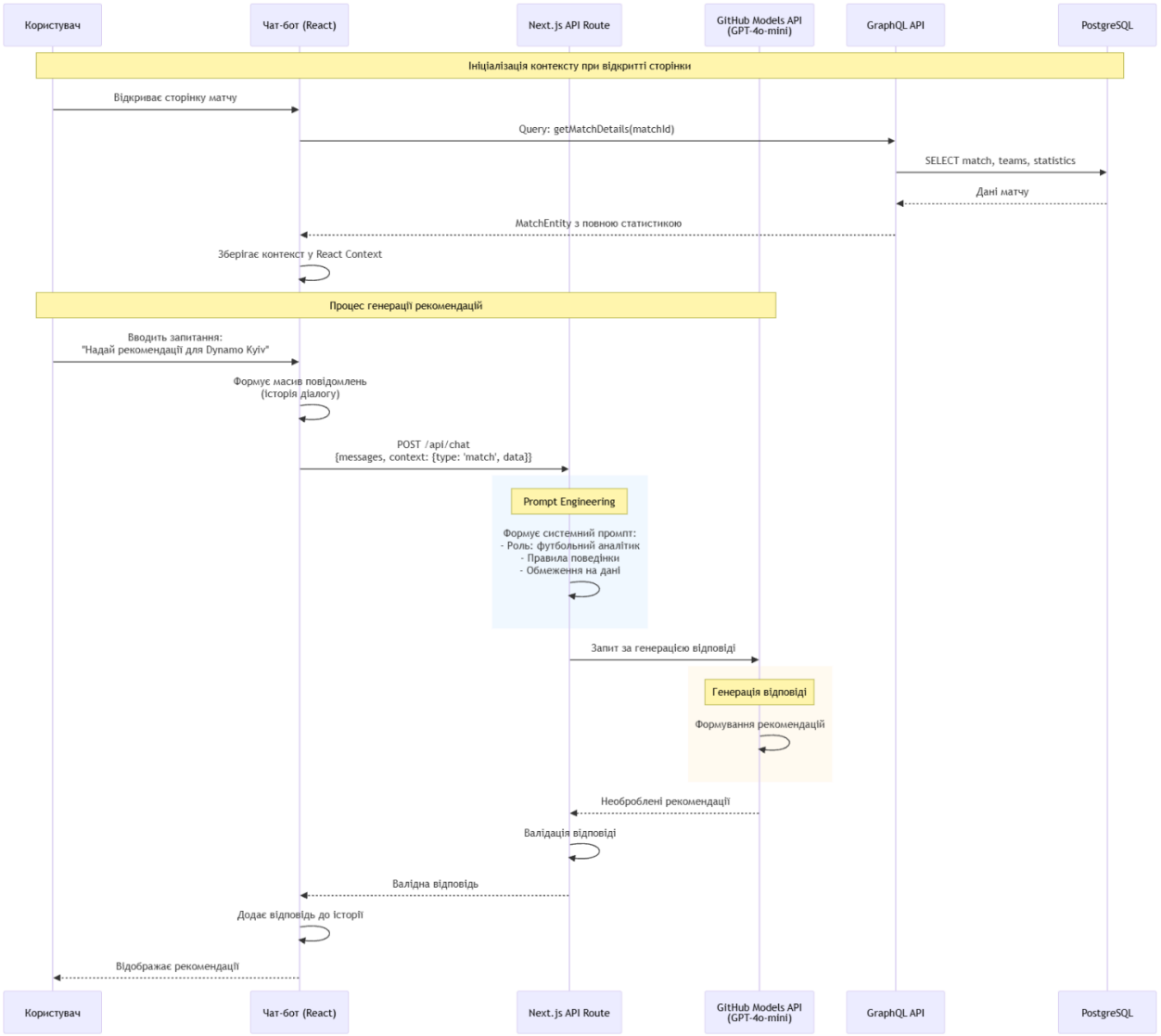


Рисунок 4.21 – Sequence діаграма процесу генерації рекомендацій за допомогою LLM

Згідно діаграми, процес генерації рекомендацій складається з двох основних етапів: ініціалізація контексту та безпосередня генерація рекомендацій. При відкритті сторінки матчу веб-додаток автоматично завантажує повну статистику матчу через GraphQL та зберігає її у React Context, що робить ці дані доступними для чат-бота без додаткових запитів.

Для забезпечення правильної відповіді від LLM згідно з дослідженням застосовано prompt engineering за допомогою принципів якого описана інструкція і очікувана поведінку LLM під час відповіді, що дозволить обмежити її лише на інформації, що надається йому при запиті. Створену інструкцію наведено у лістингу 4.12.

Лістинг 4.12 – Код інструкції для LLM для створення рекомендацій

```
const basePrompt = `You are a professional football tactical analyst assistant for coaches and technical staff.

CRITICAL RULES:
1. ONLY use data provided in the context below. Do NOT invent, assume, or reference any statistics, players, or events not explicitly present in the provided data.
2. If asked about something not in the context, clearly state: "This information is not available in the provided match data."
3. Always respond in Ukrainian language.
4. Your audience is professional coaches and technical staff - use appropriate tactical terminology.
5. Be concise and actionable. Focus on tactical insights, not general football knowledge.
6. Structure responses with clear sections but keep them focused.

When analyzing:
- Identify specific tactical patterns from the data
- Highlight statistical anomalies that indicate problems or strengths
- Provide actionable recommendations based ONLY on provided metrics
- Reference specific player names and their exact statistics from the context

Do NOT:
- Make up statistics or player performances
- Speculate about events not in the data
- Provide generic football advice not tied to specific data points
- Suggest watching video or analyzing moments not provided`
```

Надалі в кінець цієї інструкції додається інформація відповідно контексту сторінки на якій сторінці користувач виконує запит до LLM. Відповідно якщо користувач знаходиться на сторінці певного матчу, то в кінець цієї інструкції буде додана вся інформація про результат цього матчу. На рисунку 4.22 зображено контекст, що передається до інструкції при знаходженні користувача на сторінці матча.

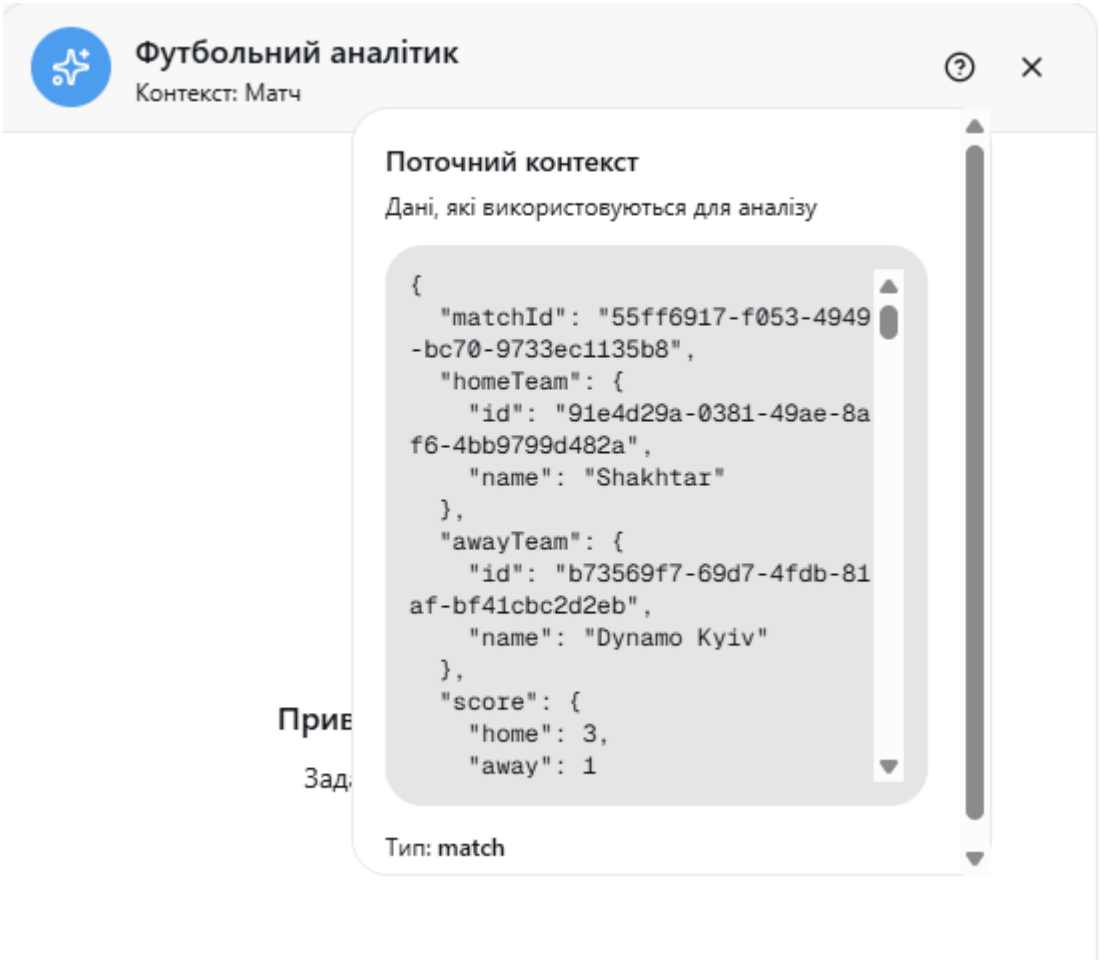


Рисунок 4.22 – Контекст, що передається до LLM на сторінці матча

Комунікація між користувачем і LLM відбувається за допомогою чату який знаходиться на кожній сторінці знизу зліва у вигляді чарівної палиці і відкривається по кліку на неї. Чат зберігає інформацію лише на поточний час сесії без збереження історії між сесіями. На рисунку 5.12 зображено приклад отримання рекомендацій до матчу між Shakhtar та Dynamo Kyiv, де Shakhtar виграв з рахунком три-один і як тренер Dynamo Kyiv було запитано рекомендації. Аналізуючи надану

рекомендацію від LLM можна побачити, що вона врахувала всю статистику, що зберігається про цей матч, навела кілька ключових етапів свого аналізу і в кінці навела рекомендації для тренера Dynamo Kyiv на що потрібно обернути увагу (див. рисунок 4.23).

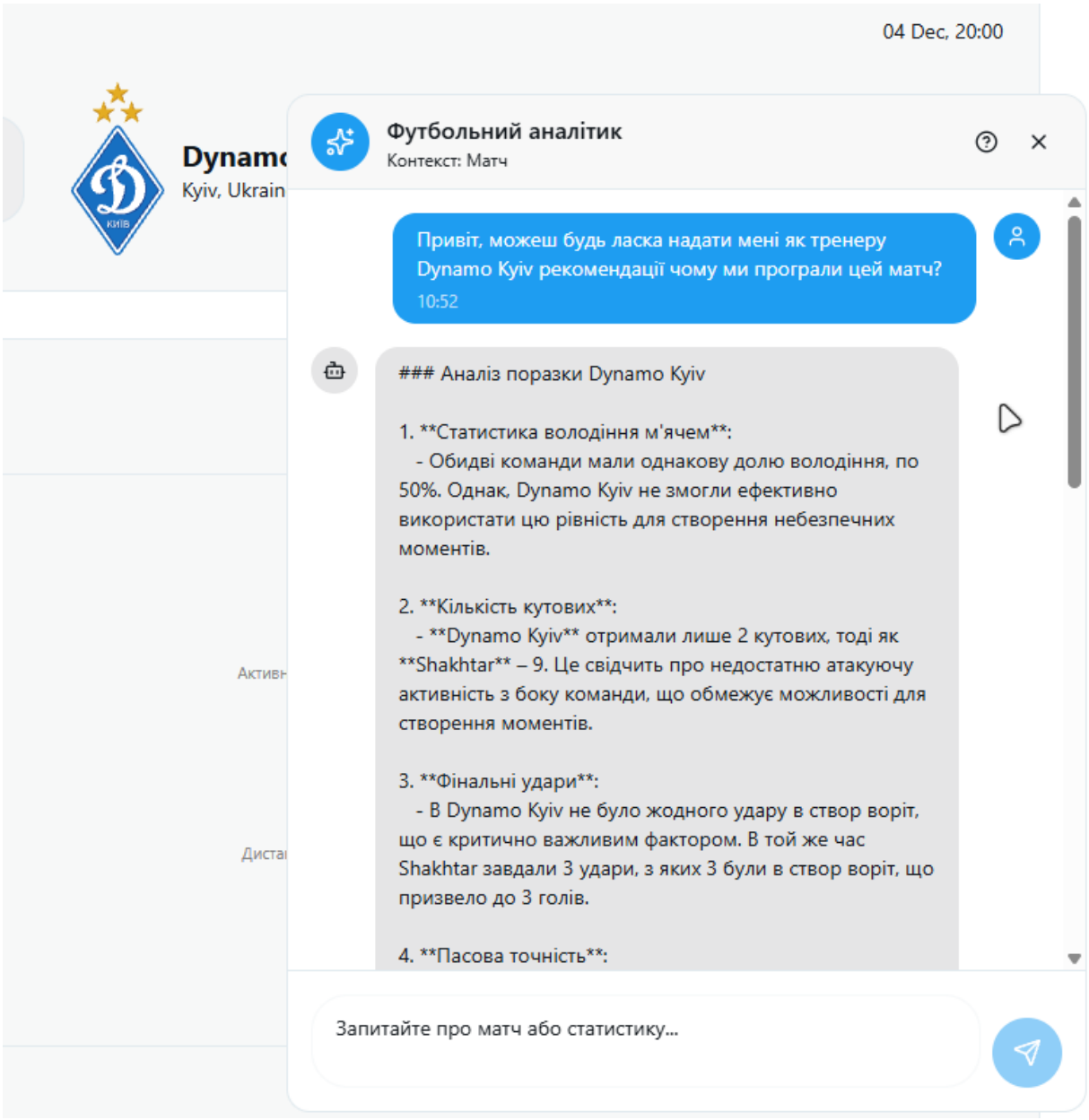


Рисунок 4.23 – Процес комунікації користувача з LLM за допомогою чата

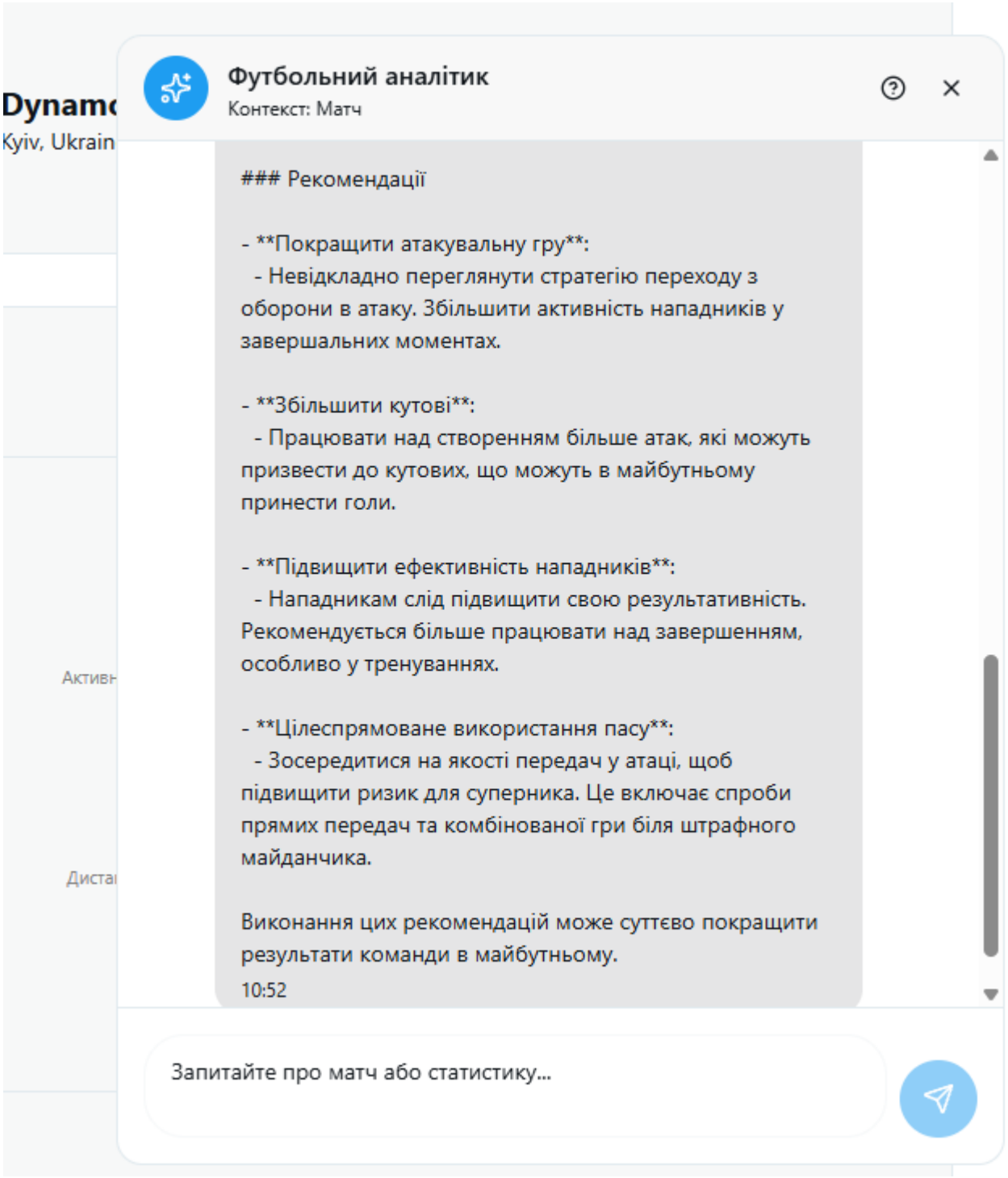


Рисунок 4.24 – Рекомендації, що були надані користувачеві LLM після аналізу результатів матча

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було виконано дослідження методів та технологій автоматизованого аналізу футбольних матчів на основі комп'ютерного зору та машинного навчання. У рамках цієї роботи було проаналізовано сучасні підходи до оцінки ефективності гравців, досліджено архітектури нейронних мереж для детекції об'єктів (YOLO, EfficientDet, Detectron2), алгоритм трекінгу (ByteTrack) та методи класифікації гравців за командною належністю (K-means кластеризація, глибоке навчання).

На основі проведеного дослідження було спроєктовано та реалізовано інформаційну систему аналізу футбольних матчів, яка складається з чотирьох основних компонентів:

– компонент обробки відео – реалізовано на базі Python 3.10 та FastAPI з використанням бібліотеки Ultralytics для роботи з YOLOv8, що виконує розпізнавання об’єктів на полі та 32 ключових точок розмітки поля. За допомогою класу ViewTransformer виконується перетворення координат з пікселів у метричну систему через гомографію, що дозволяє генерувати тактичні діаграми з видом зверху на поле та розраховувати метрики, що використовують дистанцію;

– компонент розрахунку метрик – реалізовано на TypeScript 5 з високопродуктивним runtime Bun 1.2, що обчислює фізичні метрики гравців (дистанція, швидкість) та командні показники (володіння м'ячем);

– компонент метаданих – побудовано на NestJS 10 з GraphQL API та Prisma ORM, що управляє метаданими команд, гравців та матчів у PostgreSQL;

– веб-додаток, що спрощує використання інформаційної системи користувачем та яке інтегрується з LLM API для генерації тактичних рекомендацій на основі аналізу метрик.

Результатом кваліфікаційної роботи є система, що здатна автоматично аналізувати матчі, розраховувати об’єктивні метрики і надавати рекомендації тренерському штабу, але має незначні похибки, що не дозволяє повноцінно замінити аналіз гри тренером і створює міцну основу для подальшої роботи в галузі спортивної аналітики та розробки інтелектуальних систем обробки відео.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Ivanov V., Latyshev O. Coaching decision support system using information technology. 1st International Scientific and Practical Conference «The Integration of Research, Innovation and Economy», 2025. с. 18-20. URL: https://doi.org/10.70286/isu-08.10.2025 (дата доступу 08.11.2025).

2. McHale, I. G., Scarf, P. On the Development of a Soccer Player Performance Rating System for the English Premier League. ResearchGate. 2012. URL: https://www.researchgate.net/publication/259927379_On_the_Development_of_a_Soccer_Player_Performance_Rating_System_for_the_English_Premier_League (дата доступу 10.11.2025).

3. Oliva-Lozano, J. M., Martín-Fuentes, I., Muyor, J. M. Summarizing Physical Performance in Professional Soccer: Development of a New Composite Index. Nature Scientific Reports. 2024. URL: https://www.nature.com/articles/s41598-024-65581-5 (дата доступу 12.11.2025).

4. Pappalardo, L., Cintia, P., Rossi, A., Massucco, E., Ferragina, P., Pedreschi, D., Giannotti, F. PlayeRank. Data-Driven Performance Evaluation and Player Ranking in Soccer via a Machine Learning Approach. 2019. URL: https://doi.org/10.1145/3343172 (дата доступу 14.11.2025).

5. Spearman, W. Beyond Expected Goals. MIT Sloan Sports Analytics Conference. 2018. URL: https://www.researchgate.net/publication/327139841_Beyond_Expected_Goals (дата доступу 14.11.2025).

6. Decroos, T., Bransen, L., Van Haaren, J., Davis, J. Actions Speak Louder Than Goals: Valuing Player Actions in Soccer. Data Mining and Knowledge Discovery. Springer, 2019. URL: https://doi.org/10.1145/3292500.3330758 (дата доступу 15.11.2025).

7. Fernandez, J., Bornn, L. Valuing On-the-Ball Actions in Soccer: A Critical Survey and New Approaches. Frontiers in Sports and Active Living. 2021. URL: https://tomdecroos.github.io/reports/xt_vs_vaep.pdf (дата доступу 16.11.2025).

8. Іванов В., Латишев О. Використання комп'ютерного зору для автоматизації аналізу футбольних матчів. І Міжнародна науково-практична конференція

«EUROPEAN SCIENCE AND INNOVATION CONGRESS», 8-10.12.2025, Барселона, Іспанія. с. 204-207. URL:

9. Law 1 The Field of Play. [Електронний ресурс] – Режим доступу до ресурсу: https://www.theifab.com/laws/latest/the-field-of-play

10. Maglo, A., Orcesi, A., Denize, J., Pham, Q. C. Individual Locating of Soccer Players from a Single Moving View. Sensors 23(18):7938, 2023. URL: https://doi.org/10.3390/s23187938 (дата доступу 17.11.2025).

11. Cioppa, A., Giancola, S., Somers, V., Magera, F., Zhou, X., et al. SoccerNet 2023 Challenges Results. Sports Engineering, 2024. URL: https://doi.org/10.48550/arXiv.2309.06006 (дата доступу 17.11.2025).

12. Azzami S. Y. A., Pramono H., Alzami F., Irawan C. Clustering and Profiling Analysis for FIFA Football Player using K-Means. Jurnal Informatika Jurnal Pengembangan IT 10(1):178-189, 2025. URL: https://doi.org/10.30591/jpit.v10i1.7897 (дата доступу 17.11.2025).

13. Falaleev, N. S., Chen, R. Enhancing Soccer Camera Calibration Through Keypoint Exploitation. Proceedings of the 7th ACM International Workshop on Multimedia Content Analysis in Sports (MMSports '24), 2024. URL: https://doi.org/10.48550/arXiv.2410.07401 (дата доступу 17.11.2025).

14. Dang X. Artificial Intelligence-Driven Tactical Analysis In Football Training. Pacific International Journal. 2025. URL: https://doi.org/10.55014/pij.v8i4.847 (дата доступу 17.11.2025).

15. Meijer, W., Trubiani, C., Aleti, A. Experimental evaluation of architectural software performance design patterns in microservices. Journal of Systems and Software 217:112182, 2024. URL: https://doi.org/10.1016/j.jss.2024.112183 (дата доступу 17.11.2025).

16. Richardson, C. Microservices Patterns: With Examples in Java, 2018. – 520 с.

17. Raschka, S., & Mirjalili, V. Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2 (3rd ed.), 2019. – 770 с.

18. Buna, S. GraphQL in Action, 2020. – 384 с.

19. Rathore, M., Bagui, S. S. MongoDB: Meeting the Dynamic Needs of Modern Applications. Encyclopedia (MDPI) 4(4):93, 2024. URL: https://www.mdpi.com/2673-8392/4/4/93 (дата доступу 17.11.2025).

20. Ultralytics YOLO. [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/ultralytics/ultralytics

21. Football players detection dataset. [Електронний ресурс] – Режим доступу до ресурсу: https://universe.roboflow.com/roboflow-jvuqo/football-players-detection-3zvbc

22. Football field detection dataset. [Електронний ресурс] – Режим доступу до ресурсу: https://universe.roboflow.com/roboflow-jvuqo/football-field-detection-f07vi